FirmXRay: Detecting Bluetooth Link Layer Vulnerabilities From Bare-Metal Firmware

Haohuang Wen, Zhiqiang Lin, and Yinqian Zhang

CCS 2020
Bluetooth Low Energy

Bluetooth 4.0

Low Energy
Low Technical Barrier for IoT Development
Low Technical Barrier for IoT Development
Low Technical Barrier for IoT Development
Low Technical Barrier for IoT Development
Low Technical Barrier for IoT Development

Are they secure?
BLE Workflow

Peripheral

Central
BLE Workflow
BLE Workflow

1. Broadcast
2. Scan
3. Connection Request
4. Connection Established
BLE Workflow

(I) Broadcast and Connection

1. Broadcast
2. Scan
3. Connection Request
4. Connection Established

(II) Pairing and Bonding

5. Pairing Feature Exchange
BLE Workflow

1. Broadcast
2. Scan
3. Connection Request
4. Connection Established

(II) Pairing and Bonding

5. Pairing Feature Exchange
6. STK/LTK Generation (Legacy/LESC Pairing)
7. Transport Specific Key Distribution
BLE Workflow

(I) Broadcast and Connection
1. Broadcast
2. Scan
3. Connection Request
4. Connection Established

(II) Pairing and Bonding
5. Pairing Feature Exchange
6. STK/LTK Generation (Legacy/LESC Pairing)
7. Transport Specific Key Distribution

(III) Data Transmission
8. Read/Write Data
BLE Link Layer Vulnerabilities

Vulnerabilities

1. **Identity Tracking.** Configure static MAC address during broadcast [DPCM16].
BLE Link Layer Vulnerabilities

Vulnerabilities

1. **Identity Tracking.** Configure static MAC address during broadcast [DPCM16].
2. **Active MITM.** Just Works is adopted as the pairing method.
BLE Link Layer Vulnerabilities

Vulnerabilities

1. **Identity Tracking**: Configure static MAC address during broadcast [DPCM16].
2. **Active MITM**: Just Works is adopted as the pairing method.
3. **Passive MITM**: Legacy pairing is used during key exchange [ble14].
BLE Link Layer Vulnerabilities

Vulnerabilities

1. **Identity Tracking**. Configure static MAC address during broadcast [DPCM16].
2. **Active MITM**. Just Works is adopted as the pairing method.
3. **Passive MITM**. Legacy pairing is used during key exchange [ble14].

Identification

1. Traffic analysis
2. Mobile app analysis
BLE Link Layer Vulnerabilities

Vulnerabilities

1. **Identity Tracking**: Configure static MAC address during broadcast [DPCM16].
2. **Active MiTM**: Just Works is adopted as the pairing method.
3. **Passive MiTM**: Legacy pairing is used during key exchange [ble14].

Identification

1. Traffic analysis
2. Mobile app analysis
3. Firmware analysis
An Example of a Just Works Pairing Vulnerability

Read Only Memory

```
1  243a8    mov    r2, #0x0
2  243aa   orr    r2, #0x1
3  243ac   and    r2, #0xe1
4  243ae   add    r2, #0xc
5  243b0   and    r2, #0xdf
6  243b2   ldr    r1, [0x260c8]
7  243b4   str    r2, [r1, #0x0]
... 
8  25f44   ldr    r2, [0x260c8]  
9  25f46   mov    r1, #0x0
10 25f48   svc   0x7f  // SD_BLE_GAP_SEC_PARAMS_REPLY
    ... 
11 260c8   0x20003268  // ble_gap_sec_parms_t*
```

Register Values

```
r1 = 0x0
r2 = 0x0
```
An Example of a Just Works Pairing Vulnerability

Read Only Memory

<table>
<thead>
<tr>
<th>Line</th>
<th>Address</th>
<th>Instruction</th>
<th>Source Address</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>243a8</td>
<td>mov r2, #0x0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>243aa</td>
<td>orr r2, #0x1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>243ac</td>
<td>and r2, #0xe1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>243ae</td>
<td>add r2, #0xc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>243b0</td>
<td>and r2, #0xdf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>243b2</td>
<td>ldr r1, [0x260c8]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>243b4</td>
<td>str r2, [r1, #0x0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>25f44</td>
<td>ldr r2, [0x260c8]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>25f46</td>
<td>mov r1, #0x0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>25f48</td>
<td>svc 0x7f</td>
<td>SD_BLE_GAP_SEC_PARAMS_REPLY</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>260c8</td>
<td>0x20003268</td>
<td>ble_gap_sec_parms_t*</td>
<td></td>
</tr>
</tbody>
</table>

Register Values

r1 = 0x0
r2 = 0xD
An Example of a Just Works Pairing Vulnerability

Read Only Memory

1. 243a8 mov r2, #0x0
2. 243aa orr r2, #0x1
3. 243ac and r2, #0xe1
4. 243ae add r2, #0xc
5. 243b0 and r2, #0xdf
6. 243b2 ldr r1, [0x260c8]
7. 243b4 str r2, [r1, #0x0]
...  
8. 25f44 ldr r2, [0x260c8]
9. 25f46 mov r1, #0x0
10. 25f48 svc 0x7f
   // SD_BLE_GAP_SEC_PARAMS_REPLY
...  
11. 260c8 0x20003268
   // ble_gap_sec_params_t*

Random Access Memory

```
Struct ble_gap_sec_params_t

20003268 uint8 pairing_feature
20003269 uint8 min_key_size
20003270 uint8 max_key_size
20003271 ble_gap_sec_kdist_t kdist_own
20003275 ble_gap_sec_kdist_t kdist_peer
```

Register Values

- r1 = 0x20003268
- r2 = 0xD
An Example of a Just Works Pairing Vulnerability

Read Only Memory

1 243a8  mov  r2, #0x0
2 243aa  orr  r2, #0x1
3 243ac  and  r2, #0xe1
4 243ae  add  r2, #0xc
5 243b0  and  r2, #0xdf
6 243b2  ldr  r1, [0x260c8]
7 243b4  str  r2, [r1, #0x0]

... 8 25f44  ldr  r2, [0x260c8]
9 25f46  mov  r1, #0x0
10 25f48  svc  0x7f

// SD_BLE_GAP_SEC_PARAMS_REPLY
...
11 260c8  0x20003268
// ble_gap_sec_params_t*

Random Access Memory

Struct ble_gap_sec_params_t

20003268  uint8 pairing_feature = 0xD

20003269  uint8 min_key_size
20003270  uint8 max_key_size
20003271  ble_gap_sec_kdist_t kdist_own
20003275  ble_gap_sec_kdist_t kdist_peer

Register Values

r1 = 0x20003268
r2 = 0xD
An Example of a Just Works Pairing Vulnerability

**Read Only Memory**

1. `243a8` mov r2, #0x0
2. `243aa` orr r2, #0x1
3. `243ac` and r2, #0xe1
4. `243ae` add r2, #0xc
5. `243b0` and r2, #0xdf
6. `243b2` ldr r1, [0x260c8]
7. `243b4` str r2, [r1, #0x0]

**Random Access Memory**

`Struct ble_gap_sec_params_t`

- `20003268` uint8 pairing_feature = 0xD
- `20003269` uint8 min_key_size
- `20003270` uint8 max_key_size
- `20003271` ble_gap_sec_kdist_t kdist_own
- `20003275` ble_gap_sec_kdist_t kdist_peer

**Register Values**

- `r1 = 0x0`
- `r2 = 0x20003268`

// ble_gap_sec_params_t*
An Example of a Just Works Pairing Vulnerability

Read Only Memory

1. 243a8  mov r2, #0x0
2. 243aa  orr r2, #0x1
3. 243ac  and r2, #0xe1
4. 243ae  add r2, #0xc
5. 243b0  and r2, #0xdf
6. 243b2  ldr r1, [0x260c8]
7. 243b4  str r2, [r1, #0x0]
...
8. 25f44  ldr r2, [0x260c8]
9. 25f46  mov r1, #0x0
10. 25f48  svc 0x7f
    // SD_BLE_GAP_SEC_PARAMS_REPLY
    ...
11. 260c8  0x20003268
    // ble_gap_sec_params_t*

Random Access Memory

Struct ble_gap_sec_params_t

20003268  uint8 pairing_feature = 0xD

<table>
<thead>
<tr>
<th>BOND</th>
<th>MITM</th>
<th>IO</th>
<th>OOB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

20003269  uint8 min_key_size
20003270  uint8 max_key_size
20003271  ble_gap_sec_kdist_t kdist_own
20003275  ble_gap_sec_kdist_t kdist_peer

Register Values

r1 = 0x0
r2 = 0x20003268
An Example of a Just Works Pairing Vulnerability

Correct Firmware Disassembling

Read Only Memory

1  243a8   mov    r2, #0x0
2  243aa  orr    r2, #0x1
3  243ac and   r2, #0xe1
4  243ae add r2, #0xc
5  243b0  and r2, #0xdf
6  243b2  ldr r1, [0x260c8]
7  243b4 str r2, [r1, #0x0]
... 
8  25f44  ldr r2, [0x260c8]
9  25f46  mov r1, #0x0
10  25f48  svc 0x7f
// SD_BLE_GAP_SEC_PARAMS_REPLY
... 
11  260c8  0x20003268
    // ble_gap_secParms_t*

Random Access Memory

Struct ble_gap_sec_params_t

20003268  uint8  pairing_feature = 0xD

<table>
<thead>
<tr>
<th>BOND</th>
<th>MITM</th>
<th>IO</th>
<th>OOB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

Register Values

r1 = 0x0
r2 = 0x20003268

Correct Firmware Disassembling

Read Only Memory

1  243a8   mov    r2, #0x0
2  243aa  orr    r2, #0x1
3  243ac and   r2, #0xe1
4  243ae add r2, #0xc
5  243b0  and r2, #0xdf
6  243b2  ldr r1, [0x260c8]
7  243b4 str r2, [r1, #0x0]
... 
8  25f44  ldr r2, [0x260c8]
9  25f46  mov r1, #0x0
10  25f48  svc 0x7f
// SD_BLE_GAP_SEC_PARAMS_REPLY
... 
11  260c8  0x20003268
    // ble_gap_secParms_t*
An Example of a Just Works Pairing Vulnerability

Correct Firmware Disassembling

Read Only Memory

1. 243a8  mov  r2, #0x0
2. 243aa  orr  r2, #0x1
3. 243ac  and  r2, #0xe1
4. 243ae  add  r2, #0xc
5. 243b0  and  r2, #0xdf
6. 243b2  ldr  r1, [0x260c8]
7. 243b4  str  r2, [r1, #0x0]
...
8. 25f44  ldr  r2, [0x260c8]
9. 25f46  mov  r1, #0x0
10. 25f48  svc  0x7f
// SD_BLE_GAP_SEC_PARAMS_REPLY
...
11. 260c8  0x20003268
  // ble_gap_sec_parms_t*

Recognize data structures

Random Access Memory

Struct ble_gap_sec_params_t

- uint8 pairing_feature = 0xD
  - BOND 1, MITM 0
  - IO 3, OOB 0
- uint8 min_key_size
- uint8 max_key_size
- ble_gap_sec_kdist_t kdist_own
- ble_gap_sec_kdist_t kdist_peer

Register Values

- r1 = 0x0
- r2 = 0x20003268
An Example of a Just Works Pairing Vulnerability

Correct Firmware Disassembling

Read Only Memory

1. 243a8 mov r2, #0x0
2. 243aa orr r2, #0x1
3. 243ac and r2, #0xe1
4. 243ae add r2, #0xc
5. 243b0 and r2, #0xdf
6. 243b2 ldr r1, [0x260c8]
7. 243b4 str r2, [r1, #0x0]
...
8. 25f44 ldr r2, [0x260c8]
9. 25f46 mov r1, #0x0
10. 25f48 svc 0x7f
    // SD_BLE_GAP_SEC_PARAMS_REPLY
...
11. 260c8 0x20003268
    // ble_gap_sec_params_t*

Recognize data structures

Random Access Memory

Struct ble_gap_sec_params_t

20003268 uint8 pairing_feature = 0xD

<table>
<thead>
<tr>
<th>BOND</th>
<th>MITM</th>
<th>IO</th>
<th>OOB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
// BOND = 1, MITM = 0
// IO   = 3, OOB  = 0

20003269 uint8 min_key_size
20003270 uint8 max_key_size
20003271 ble_gap_sec_kdist_t kdist_own
20003275 ble_gap_sec_kdist_t kdist_peer

Value computation

Register Values

r1 = 0x0
r2 = 0x20003268
FirmXRay Overview

Robust Firmware Disassembling

Precise Data Structure Recognition

Configuration Value Resolution

Disassembler

Constraints

X = argmax_N(x)_{x \in \mathbb{R}}

Vulnerabilities

Detection Policies

Bare-metal Firmware
Robust Firmware Disassembling
Robust Firmware Disassembling

Incorrect Base 0x0

05452  ldr  r0, pc+0x72
05454  blx  r0=>0x22A90
...
054c4  0x22A90
...
Function Foo()
07a90  push  {r3, r4, r5, lr}
...
1fe52  ldr  r0, pc+0x146
1fe54  ldma  r0=>0x23058, {r4, r5, r6}
...
04f98  0x23058
...
08058 "KinsaHealth"

Correct Base 0x1B000

20452  ldr  r0, pc+0x72
20454  blx  r0=>0x22A90
...
204c4  0x22A90
...
Function Foo()
22a90  push  {r3, r4, r5, lr}
...
1fe52  ldr  r0, pc+0x146
1fe54  ldma  r0, {r4, r5, r6}
...
1ff98  0x23058
...
23058 "KinsaHealth"

(1) Absolute Function Pointer

(2) Absolute String Pointer
Robust Firmware Disassembling

Base
0x0

05452  ldr    r0, pc+0x72
05454  blx    r0=>0x22A90

... 054c4  0x22A90

Function Foo()
07a90  push  {r3, r4, r5, lr}

04e52  ldr    r0, pc+0x146
04e54  ldmia  r0=>0x23058, {r4, r5, r6}

... 04f98  0x23058

08058  "KinsaHealth"
Robust Firmware Disassembling

**Introduction**

**Motivating Example**

**FIRMXRAY**

**Evaluation**

**Discussion**

**Takeaway**

**References**

---

**Base 0x0**

<table>
<thead>
<tr>
<th>05452</th>
<th>ldr</th>
<th>r0, pc+0x72</th>
</tr>
</thead>
<tbody>
<tr>
<td>05454</td>
<td>blx</td>
<td>r0=&gt;0x22A90</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>054c4</td>
<td>0x22A90</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Function Foo()**

<table>
<thead>
<tr>
<th>07a90</th>
<th>push</th>
<th>{r3, r4, r5, lr}</th>
</tr>
</thead>
</table>

---

**Absolute Pointers:** 0x22A90, 0x23058

**Gadgets:** 0x07A90, 0x08058
Robust Firmware Disassembling

Absolute Pointers: 0x22A90, 0x23058

Gadgets: 0x07A90, 0x08058

N(0x1B000) = 2
Robust Firmware Disassembling

Base 0x0

05452  ldr  r0, pc+0x72
05454  blx  r0=>0x22A90
...
054c4  0x22A90
...
Function Foo()
07a90  push  {r3, r4, r5, lr}

Absolute Pointers: 0x22A90, 0x23058

Gadgets: 0x07A90, 0x08058

0x1B000 = 0x22A90 - 0x07A90
= 0x23058 - 0x08058

N(0x1B000) = 2
Precise Data Structure Recognition

Read Only Memory

1 243a8  mov    r2, #0x0
2 243aa  orr    r2, #0x1
3 243ac  and    r2, #0xe1
4 243ae  add    r2, #0xc
5 243b0  and    r2, #0xdf
6 243b2  ldr    r1, [0x260c8]
7 243b4  str    r2, [r1,#0x0]
...
8 25f44  ldr    r2, [0x260c8]
9 25f46  mov    r1, #0x0
10 25f48  svc    0x7f
// SD_BLE_GAP_SEC_PARAMS_REPLY(r0, r1, r2)
...
11 260c8  0x20003268
// ble_gap_sec_Parms_t*
Configuration Value Resolution

**Robust Firmware Disassembling**

**Precise Data Structure Recognition**

**Configuration Value Resolution**

---

### Program Path

1. `243a8` mov r2, #0x0
2. `243aa` orr r2, #0x1
3. `243ac` and r2, #0xe1
4. `243ae` add r2, #0xc
5. `243b0` and r2, #0xdf
6. `243b2` ldr r1, [0x260c8]
7. `243b4` str r2, [r1, #0x0]
8. `25f44` ldr r2, [0x260c8]
9. `25f46` mov r1, #0x0
10. `25f48` svc 0x7f
   // SD_BLE_GAP_SEC_PARAMS_REPLY
   ...
11. `260c8` 0x20003268
   // ble_gap_sec_parms_t*

### Read Only Memory

- `1 243a8` mov r2, #0x0
- `2 243aa` orr r2, #0x1
- `3 243ac` and r2, #0xe1
- `4 243ae` add r2, #0xc
- `5 243b0` and r2, #0xdf
- `6 243b2` ldr r1, [0x260c8]
- `7 243b4` str r2, [r1, #0x0]
- `8 25f44` ldr r2, [0x260c8]
- `9 25f46` mov r1, #0x0
- `10 25f48` svc 0x7f
  // SD_BLE_GAP_SEC_PARAMS_REPLY
  ...
- `11 260c8` 0x20003268
  // ble_gap_sec_parms_t*
Program Path

```
l dr r2, [0x260c8]
str r2, [r1, #0x0]
```

Read Only Memory

```
l dr r2, [0x260c8]
mov r2, #0x0
orr r2, #0x1
and r2, #0xe1
add r2, #0xc
and r2, #0xdf
```

// ble_gap_sec_parms_t*

```
ldr r2, [0x260c8]
str r2, [r1, #0x0]
```
Configuration Value Resolution

Robust Firmware Disassembling

Precise Data Structure Recognition

Configuration Value Resolution

Read Only Memory

1 243a8  mov    r2, #0x0
2 243aa  orr    r2, #0x1
3 243ac  and    r2, #0xe1
4 243ae  add    r2, #0xc
5 243b0  and    r2, #0xdf
6 243b2  ldr    r1, [0x260c8]
7 243b4  str    r2, [r1, #0x0]
...
8 25f44  ldr    r2, [0x260c8]
9 25f46  mov    r1, #0x0
10 25f48  svc 0x7f
   // SD_BLE_GAP_SEC_PARAMS_REPLY
   ...
11 260c8  0x20003268
   // ble_gap_secParms_t*

Program Path

ldr r2, [0x260c8]
str r2, [r1, #0x0]
ldr r1, [0x260c8]
and r2, #0x0
add r2, #0xc
and r2, #0xdf
orr r2, #0xe1
mov r2, #0x0
Configuration Value Resolution

Robust Firmware Disassembling

Precise Data Structure Recognition

Configuration Value Resolution

Read Only Memory

```
1 243a8 mov r2, #0x0
2 243aa orr r2, #0x1
3  243ac and r2, #0xe1
4  243ae add r2, #0xc
5 243b0 and r2, #0xdf
6 243b2 ldr r1, [0x260c8]
7  243b4 str r2, [r1, #0x0]
...
8 25f44 ldr r2, [0x260c8]
9 25f46 mov r1, #0x0
10 25f48 svc 0x7f
  // SD_BLE_GAP_SEC_PARAMS_REPLY
...
11 260c8 0x20003268
  // ble_gap_secParms_t*
```

Program Path

```
ldr r2, [0x260c8]
str r2, [r1, #0x0]
ldr r1, [0x260c8]
and r2, #0xdf
add r2, #0xc
and r2, #0xe1
orr r2, #0x1
mov r2, #0x0
r2 = 0x20003268
```
## Configuration Value Resolution

<table>
<thead>
<tr>
<th>Policy</th>
<th>SDK Function Name</th>
<th>Reg. Index</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SD_BLE_GAP_ADDR_SET</td>
<td>0</td>
<td>Configure the MAC address</td>
</tr>
<tr>
<td></td>
<td>SD_BLE_GAP_APPEARANCE_SET</td>
<td>0</td>
<td>Set device description</td>
</tr>
<tr>
<td></td>
<td>SD_BLE_GATTS_SERVICE_ADD</td>
<td>0, 1</td>
<td>Add a BLE GATT service</td>
</tr>
<tr>
<td></td>
<td>SD_BLE_GATTS_CHARACTERISTIC_ADD</td>
<td>2</td>
<td>Add a BLE GATT characteristic</td>
</tr>
<tr>
<td></td>
<td>SD_BLE_UUID_VS_ADD</td>
<td>0</td>
<td>Specify the UUID base</td>
</tr>
<tr>
<td></td>
<td>GAP_ConfigDeviceAddr*</td>
<td>0</td>
<td>Setup the address type</td>
</tr>
<tr>
<td></td>
<td>GATTServApp_RegisterService*</td>
<td>0</td>
<td>Register BLE GATT service</td>
</tr>
<tr>
<td>(i)</td>
<td>SD_BLE_GAP_SEC_PARAMS_REPLY</td>
<td>2</td>
<td>Reply peripheral pairing features</td>
</tr>
<tr>
<td></td>
<td>SD_BLE_GAP_AUTH</td>
<td>1</td>
<td>Reply central pairing features</td>
</tr>
<tr>
<td></td>
<td>SD_BLE_GAP_AUTH_KEY_REPLY</td>
<td>1, 2</td>
<td>Reply with an authentication key</td>
</tr>
<tr>
<td></td>
<td>SD_BLE_GATTS_CHARACTERISTIC_ADD</td>
<td>2</td>
<td>Add a BLE GATT characteristic</td>
</tr>
<tr>
<td></td>
<td>GAPBondMgr_SetParameter*</td>
<td>2</td>
<td>Setup pairing parameters</td>
</tr>
<tr>
<td></td>
<td>GATTServApp_RegisterService*</td>
<td>0</td>
<td>Register BLE GATT service</td>
</tr>
<tr>
<td>(ii)</td>
<td>SD_BLE_GAP_LESC_DHKEY_REPLY</td>
<td>0</td>
<td>Reply with a DH key</td>
</tr>
<tr>
<td></td>
<td>GAPBondMgr_SetParameter*</td>
<td>2</td>
<td>Setup pairing parameters</td>
</tr>
</tbody>
</table>
Firmware Collection

2M Free Apps
Firmware Collection

2M Free Apps

Filter

13K BLE Apps
Firmware Collection

2M Free Apps 13K BLE Apps 793 Firmware
Filter Unpack Extract
Firmware Collection

2M Free Apps

Filter

13K BLE Apps

Unpack

Extract

793 Firmware

768 Nordic

25 TI
Firmware Categorization

- Firmware categorization

<table>
<thead>
<tr>
<th>Category</th>
<th>Nordic-based Firmware</th>
<th>Others</th>
<th>Medical Devices</th>
<th>Others</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wearable</td>
<td>204</td>
<td>76</td>
<td>67</td>
<td>58</td>
</tr>
<tr>
<td>Sensor</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>Tag (Tracker)</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>Smart Lock</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Smart Toy</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Table: Top categories of firmware.
Firmware Categorization

- Firmware categorization
  - Descriptive APIs (e.g., SD_BLE_GAP_APPEARANCE_SET)
Firmware Categorization

- Firmware categorization
  - Descriptive APIs (e.g., `SD_BLE_GAP_APPEARANCE_SET`)
  - Mobile app descriptions
Firmware Categorization

- Firmware categorization
  - Descriptive APIs (e.g., `SD_BLE_GAP_APPEARANCE_SET`)
  - Mobile app descriptions

<table>
<thead>
<tr>
<th>Category</th>
<th># Firmware</th>
<th># Device</th>
<th>Avg. Size (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Nordic-based Firmware</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wearable</td>
<td>204</td>
<td>138</td>
<td>98.2</td>
</tr>
<tr>
<td>Others</td>
<td>76</td>
<td>22</td>
<td>223.5</td>
</tr>
<tr>
<td>Sensor</td>
<td>67</td>
<td>51</td>
<td>80.9</td>
</tr>
<tr>
<td>Tag (Tracker)</td>
<td>58</td>
<td>41</td>
<td>84.2</td>
</tr>
<tr>
<td>Robot</td>
<td>41</td>
<td>21</td>
<td>117.7</td>
</tr>
<tr>
<td>Medical Devices</td>
<td>41</td>
<td>21</td>
<td>138.6</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>793</td>
<td>538</td>
<td>102.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Category</th>
<th># Firmware</th>
<th># Device</th>
<th>Avg. Size (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TI-based Firmware</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor</td>
<td>19</td>
<td>19</td>
<td>132.9</td>
</tr>
<tr>
<td>Smart Lock</td>
<td>2</td>
<td>2</td>
<td>46.3</td>
</tr>
<tr>
<td>Smart Toy</td>
<td>2</td>
<td>2</td>
<td>47.8</td>
</tr>
<tr>
<td>Medical Devices</td>
<td>1</td>
<td>1</td>
<td>70.2</td>
</tr>
<tr>
<td>Others</td>
<td>1</td>
<td>1</td>
<td>76.7</td>
</tr>
</tbody>
</table>

Table: Top categories of firmware.
Firmware Categorization

- **Firmware categorization**
  - Descriptive APIs (e.g., `SD_BLE_GAP_APPEARANCE_SET`)
  - Mobile app descriptions

- **Firmware aggregation**
  - Aggregate different versions of firmware of the same device
  - The 793 firmware represent 538 real devices

<table>
<thead>
<tr>
<th>Category</th>
<th># Firmware</th>
<th># Device</th>
<th>Avg. Size (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Nordic-based Firmware</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wearable</td>
<td>204</td>
<td>138</td>
<td>98.2</td>
</tr>
<tr>
<td>Others</td>
<td>76</td>
<td>22</td>
<td>223.5</td>
</tr>
<tr>
<td>Sensor</td>
<td>67</td>
<td>51</td>
<td>80.9</td>
</tr>
<tr>
<td>Tag (Tracker)</td>
<td>58</td>
<td>41</td>
<td>84.2</td>
</tr>
<tr>
<td>Robot</td>
<td>41</td>
<td>21</td>
<td>117.7</td>
</tr>
<tr>
<td>Medical Devices</td>
<td>41</td>
<td>21</td>
<td>138.6</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>793</td>
<td>538</td>
<td>102.7</td>
</tr>
<tr>
<td><strong>TI-based Firmware</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor</td>
<td>19</td>
<td>19</td>
<td>132.9</td>
</tr>
<tr>
<td>Smart Lock</td>
<td>2</td>
<td>2</td>
<td>46.3</td>
</tr>
<tr>
<td>Smart Toy</td>
<td>2</td>
<td>2</td>
<td>47.8</td>
</tr>
<tr>
<td>Medical Devices</td>
<td>1</td>
<td>1</td>
<td>70.2</td>
</tr>
<tr>
<td>Others</td>
<td>1</td>
<td>1</td>
<td>76.7</td>
</tr>
</tbody>
</table>

Table: Top categories of firmware.
Identity Tracking Vulnerability Identification

Among the 538 devices, nearly all of them (98.1%) have configured random static addresses that do not change periodically.
Experiment Results

Identity Tracking Vulnerability Identification

Among the 538 devices, nearly all of them (98.1%) have configured random static addresses that do not change periodically.

<table>
<thead>
<tr>
<th>Firmware Name</th>
<th>Mobile App</th>
<th>Category</th>
<th># Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>cogobeacon</td>
<td>com.aegismobility.guardian</td>
<td>Car Accessory</td>
<td>4</td>
</tr>
<tr>
<td>sd_bl</td>
<td>fr.solem.solemwf</td>
<td>Agricultural Equip.</td>
<td>2</td>
</tr>
<tr>
<td>LRFL_nRF52</td>
<td>fr.solem.solemwf</td>
<td>Agricultural Equip.</td>
<td>2</td>
</tr>
<tr>
<td>orb</td>
<td>one.shade.app</td>
<td>Smart Light</td>
<td>1</td>
</tr>
<tr>
<td>sd_bl</td>
<td>com.rainbird</td>
<td>Agricultural Equip.</td>
<td>1</td>
</tr>
</tbody>
</table>

Table: Firmware using private MAC address.
Active MITM Vulnerability Identification

385 (71.5%) devices use Just Works pairing, which essentially does not provide any protection against active MITM attacks at the BLE link layer.
### Experiment Results

#### Active MITM Vulnerability Identification

385 (71.5%) devices use Just Works pairing, which essentially does not provide any protection against active MITM attacks at the BLE link layer.

<table>
<thead>
<tr>
<th>Item</th>
<th>N</th>
<th>T</th>
<th>Total</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td># Total Device</td>
<td>513</td>
<td>25</td>
<td>538</td>
<td>100</td>
</tr>
<tr>
<td># Device w/ active MITM vulnerability</td>
<td>384</td>
<td>1</td>
<td>385</td>
<td>71.5</td>
</tr>
<tr>
<td># Device w/ Just Works pairing only</td>
<td>317</td>
<td>1</td>
<td>318</td>
<td>59.1</td>
</tr>
<tr>
<td># Device w/ flawed Passkey implementation</td>
<td>37</td>
<td>0</td>
<td>37</td>
<td>6.9</td>
</tr>
<tr>
<td># Device w/ flawed OOB implementation</td>
<td>30</td>
<td>0</td>
<td>30</td>
<td>5.6</td>
</tr>
<tr>
<td># Device w/ secure pairing</td>
<td>6</td>
<td>24</td>
<td>30</td>
<td>3.8</td>
</tr>
<tr>
<td># Device w/ correct Passkey implementation</td>
<td>3</td>
<td>24</td>
<td>27</td>
<td>3.4</td>
</tr>
<tr>
<td># Device w/ correct OOB implementation</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Table: Pairing configurations of devices (N:Nordic, T:TI).
Passive MITM Vulnerability Identification

98.5% of the devices fail to enforce LESC pairing, and thus they can be vulnerable to passive MITM attacks if there is no application-layer encryption.
Passive MITM Vulnerability Identification

98.5% of the devices fail to enforce LESC pairing, and thus they can be vulnerable to passive MITM attacks if there is no application-layer encryption.

<table>
<thead>
<tr>
<th>Firmware Name</th>
<th>Mobile App</th>
<th>Category</th>
<th>#</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>DogBodyBoard</td>
<td>com.wowwee.chip</td>
<td>Robot</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>BW_Pro</td>
<td>com.ecomm.smart_panel</td>
<td>Tag</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Smart_Handle</td>
<td>com.exitec.smartlock</td>
<td>Smart Lock</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Sma05</td>
<td>com.smalife.watch</td>
<td>Wearable</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>CPRmeter</td>
<td>com.laerdal.cprmeter2</td>
<td>Medical Device</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>WiJumpLE</td>
<td>com.wesssrl.wijumple</td>
<td>Sensor</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>nRF Beacon</td>
<td>no.nordicsemi.android.nrfbeacon</td>
<td>Beacon</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Hoot Bank</td>
<td>com.qvivr.hoot</td>
<td>Debit Card</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Table: Firmware that enforce LESC pairing.
Attack Case Studies

nRF52840 DK

Vulnerable BLE Devices
## Attack Case Studies

<table>
<thead>
<tr>
<th>Device Name</th>
<th>Category</th>
<th>Attacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nuband Activ+</td>
<td>Wearable</td>
<td>✓</td>
</tr>
<tr>
<td>Kinsa Smart</td>
<td>Thermometer</td>
<td></td>
</tr>
<tr>
<td>Chipolo ONE</td>
<td>Tag</td>
<td>✓</td>
</tr>
<tr>
<td>SwitchBot Button Pusher</td>
<td>Smart Home</td>
<td></td>
</tr>
<tr>
<td>XOSS Cycling Computer</td>
<td>Sensor</td>
<td>✓</td>
</tr>
</tbody>
</table>

A1: User Tracking
## Attack Case Studies

<table>
<thead>
<tr>
<th>Device Name</th>
<th>Category</th>
<th>Attacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nuband Activ+</td>
<td>Wearable</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>Kinsa Smart Thermometer</td>
<td>Thermometer</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>Chipolo ONE Tag</td>
<td>Tag</td>
<td>✓</td>
</tr>
<tr>
<td><strong>SwitchBot Button Pusher</strong></td>
<td><strong>Smart Home</strong></td>
<td>✓</td>
</tr>
<tr>
<td>XOSS Cycling Computer</td>
<td>Sensor</td>
<td>✓ ✓</td>
</tr>
</tbody>
</table>

A2: Unauthorized Control
## Attack Case Studies

<table>
<thead>
<tr>
<th>Device Name</th>
<th>Category</th>
<th>Attacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nuband Activ+</td>
<td>Wearable</td>
<td>✔️</td>
</tr>
<tr>
<td>Kinsa Smart</td>
<td>Thermometer</td>
<td>✔️</td>
</tr>
<tr>
<td>Chipolo ONE</td>
<td>Tag</td>
<td>✔️</td>
</tr>
<tr>
<td>SwitchBot Button Pusher</td>
<td>Smart Home</td>
<td>✔️</td>
</tr>
<tr>
<td>XOSS Cycling Computer</td>
<td>Sensor</td>
<td>✔️</td>
</tr>
</tbody>
</table>

**A3: Sensitive Data Eavesdropping**
Discussion

► **Effectiveness.** Source of FP/FN: incorrect base address and fundamental limitations of program analysis.
**Effectiveness.** Source of FP/FN: incorrect base address and fundamental limitations of program analysis.

**Exploitation.** Not all the vulnerabilities can be exploited in practice.
Discussion

- **Effectiveness.** Source of FP/FN: incorrect base address and fundamental limitations of program analysis.

- **Exploitation.** Not all the vulnerabilities can be exploited in practice.

- **Root Cause.** Lack of hardware capabilities and misconfiguration by the developers are the two major root causes.
Discussion

- **Effectiveness.** Source of FP/FN: incorrect base address and fundamental limitations of program analysis.

- **Exploitation.** Not all the vulnerabilities can be exploited in practice.

- **Root Cause.** Lack of hardware capabilities and misconfiguration by the developers are the two major root causes.

- **Future Work.**
  - Extract more embedded firmware from apps (e.g., those downloaded from server).
  - Adapt FirmXRay to other SDKs and architectures.
  - Enable dynamic analysis and firmware emulation [CGS+20] [CWBE16] [FML20].
**Takeaway**

**FirmXRay**

- A static analysis tool based on Ghidra for detecting BLE link layer vulnerabilities from bare-metal firmware.
- A scalable approach to efficiently collect bare-metal firmware images from only mobile apps.
- Vulnerability discovery and attack case studies.

**FirmXRay**

▶ A static analysis tool based on Ghidra for detecting BLE link layer vulnerabilities from bare-metal firmware.
▶ A scalable approach to efficiently collect bare-metal firmware images from only mobile apps.
▶ Vulnerability discovery and attack case studies.

The source code is available at [https://github.com/OSUSecLab/FirmXRay](https://github.com/OSUSecLab/FirmXRay).
**Takeaway**

**FirmXRay**
- A static analysis tool based on Ghidra for detecting BLE link layer vulnerabilities from bare-metal firmware.
- A scalable approach to efficiently collect bare-metal firmware images from only mobile apps.
- Vulnerability discovery and attack case studies.

The source code is available at [https://github.com/OSUSecLab/FirmXRay](https://github.com/OSUSecLab/FirmXRay).
References


