(All) Your APs (Are) Belong to Us

Ben Seri, VP Research
Dor Zusman, Researcher
Agenda

• Bluetooth Low Energy (BLE)
• BLE in Access Points (?!)
• Over-the-air firmware upgrades – Is it secure?
• Aruba BLE vulnerability - CVE-2018-7080
• TI BLE stack RCE vulnerability - CVE-2018-16986
• Exploitation and Impact
Why Bluetooth Low Energy?

- Healthcare
- Manufacturing
- Retail
- Offices
THE CLEVER CRYPTOGRAPHY BEHIND APPLE'S 'FIND MY' FEATURE

When Apple executive Craig Federighi described a new location-tracking feature for Apple devices at the company's Worldwide Developer Conference keynote on Monday, it sounded—a sufficiently paranoid, at least—like both a physical security innovation and a potential privacy disaster. But while security experts immediately wondered whether

GOOGLE WILL REPLACE TITAN SECURITY KEY OVER A BLUETOOTH FLAW

As part of its expanded anti-phishing and account security measures, Google offers extensive support for physical authentication tokens. In a surprising setback, though, the company announced today that it has discovered a vulnerability in the Bluetooth version of its own Titan

BLE in recent news
Why do APs support BLE?

- Indoor Navigation
- Medical Asset Tracking
- Retail Customer Tracking
- Smart Sensors
Why does a wireless access point have bluetooth? (Score:0)
by Anonymous Coward on Thursday November 01, 2018 @08:05PM (#57578422)

Doesn't seem to make sense to me.

On a laptop, phone or tablet, you probably want bluetooth and wifi.

But "enterprise" wifi access points are normally wired in with a controller, and I don't see what the bluetooth would be used for.

What am I missing?

Share

Re: Why does a wireless access point have bluetooth (Score:3)
by viperidaenz (2515578) on Thursday November 01, 2018 @08:10PM (#57578442)

Obviously it's there to increase the attack area. Duh.

Parent Share

I'm reading this piece about #BLEEDINGBIT RCE in the BLE interface of enterprise wireless access points... And can't seem to get past "why do WLAN APs need Bluetooth?"
BLE Attack surface

**CVE-2018-7080**  
Affecting Aruba

**CVE-2018-16986**  
TI BLE STACK  
Affecting Cisco, Meraki

Diagram showing the attack surface with layers:
- **Application Layer (App)**
- **Generic Access Profile (GAP)**
- **Generic Attribute Protocol (GATT)**
- **Security Manager (SMP)**
- **Attribute Protocol (ATT)**
- **Logical Link Control & Adaptation Protocol (L2CAP)**
- **Link Layer (LL)**
- **Physical Layer (PHY)**
BLE Attack surface

CVE-2018-7080
Affecting Aruba

CVE-2018-16986
TI BLE STACK
Affecting Cisco, Meraki
Capturing firmware over the air?

Authentication of GATT? Based on BLE Bonding?

How to validate the firmware’s integrity? (digital signature)
OTA solutions over BLE - The problems

Firmware passed unencrypted over the air
GATT connection is unauthenticated
Firmware integrity is not validated, or uses weak cryptographic signature
$ gatttool -i hci1 --primary -b f4:5e:ab:e7:ff:5d
attr handle = 0x0001, end grp handle = 0x000b
uuid: 00001800-0000-1000-8000-00805f9b34fb

attr handle = 0x000c, end grp handle = 0x000f
uuid: 00001801-0000-1000-8000-00805f9b34fb

attr handle = 0x0010, end grp handle = 0x001c
uuid: 0000180a-0000-1000-8000-00805f9b34fb

attr handle = 0x001d, end grp handle = 0x0029
uuid: f000ffec0-0451-4000-b000-000000000000

attr handle = 0x002a, end grp handle = 0x0031
uuid: faafea00-b67b-6ee7-3d4c-424fb2f14a66

attr handle = 0x0032, end grp handle = 0xffffffff
uuid: 272fe150-6c6c-4718-a3d4-6de8a3735eff

BLE in Aruba Access Points
OAD in General

User

Device

Initiate GATT connection

Image Identify

Image Block Request

Image Block

Image Block Request

FFC1

FFC2
OAD in Aruba Access Points

User

Device

Initiate GATT connection

Image Identify

FFC1

No Response
Extracting BLE firmware

```
mov R2_1, #0x10 ; Move (Op1 <- Op2)
mov R3_1, #0 ; Move (Op1 <- Op2)
mov R0, #0xA ; Move (Op1 <- Op2)
lcall PUSH_XSTACK_I_TWO ; Long Subroutine Call
mov R4, #(
OAD_COOKIE - seg4_9200 ; Move (Op1 <- Op2)
mov R5, #0x92 ; Move (Op1 <- Op2)
mov R2, R0_2 ; Move (Op1 <- Op2)
mov R3, R1_2 ; Move (Op1 <- Op2)
lcall j_memcmp

int memcmp(const void *s1, // r2-r3
         const void *s2, // r4-r5
         size_t n);

mov A, #2 ; Move (Op1 <- Op2)
lcall DEALLOC_XSTACK8 ; Long Subroutine Call
mov A, R1 ; Move (Op1 <- Op2)
xrl A, #1 ; Exclusive OR (op1 ^ op2)
jnz seg2_DB07 ; Jump if Acc is not zero
```
static bStatus_t oadWriteAttrCB(...)
{
    ...
    if (osal_memcmp(pAttr->type.uuid,
                    oadCharUUID[OAD_CHAR_IMG_IDENTIFY],
                    ATT_UUID_SIZE)) {
        status = oadImgIdentifyWrite(connHandle, pValue);
    }
    } else if (osal_memcmp(pAttr->type.uuid,
                           oadCharUUID[OAD_CHAR_IMG_BLOCK],
                           ATT_UUID_SIZE)) {
        status = oadImgBlockWrite(connHandle, pValue);
    }
    ...
}
static bStatus_t oadWriteAttrCB(...)
{
    ...
    if (osal_memcmp(pAttr->type.uuid,
                    oadCharUUID[OAD_CHAR_IMG_IDENTIFY],
                    ATT_UUID_SIZE)) {
        status = oadImgIdentifyWrite(connHandle, pValue);
    } else if (osal_memcmp(pAttr->type.uuid,
                           oadCharUUID[OAD_CHAR_IMG_BLOCK],
                           ATT_UUID_SIZE)) {
        status = oadImgBlockWrite(connHandle, pValue);
    } ...
}

static bStatus_t ARUBA_oadWriteAttrCB(...)
{
    ...
    if (is_oad_unlocked) {
        // 128-bit UUID
        if (is_img_write_unlocked &&
            osal_memcmp(pAttr->type.uuid,
                        oadCharUUID[OAD_CHAR_IMG_IDENTIFY],
                        ATT_UUID_SIZE)) {
            status = oadImgIdentifyWrite(connHandle, pValue);
        } else if (osal_memcmp(pAttr->type.uuid,
                               oadCharUUID[OAD_CHAR_IMG_BLOCK],
                               ATT_UUID_SIZE)) {
            status = oadImgBlockWrite(connHandle, pValue);
        } else {
            status = ATT_ERR_ATTR_NOT_FOUND;
        }
    } else if (osal_memcmp(pAttr->type.uuid,
                           OAD_UNLOCK_UUID, ATT_UUID_SIZE) {
        if (osal_memcmp(pAttr->pValue, OAD_COOKIE, ATT_UUID_SIZE)) {
            is_oad_unlocked = true;
        } else if (osal_memcmp(pAttr->pValue, AB_ACCESS_COOKIE, ATT_UUID_SIZE)) {
            is_img_write_unlocked = true;
        }
    }
    ...
}
Analyzing custom OAD

```c
static bStatus_t ARUBA_oadWriteAttrCB(...)
{
    ...
    if (osal_memcmp(pAttr->type.uuid,
                      OAD_UNLOCK_UUID, ATT_UUID_SIZE)) {
        if (osal_memcmp(pAttr->pValue, OAD_COOKIE, ATT_UUID_SIZE)) {
            is_oad_unlocked = true;
        } else if (osal_memcmp(pAttr->pValue, AB_ACCESS_COOKIE, ATT_UUID_SIZE)) {
            is_img_write_unlocked = true;
        }
    }
    ...
}
```
SHHHH...

MY SECRET PASSWORD IS MEOW1234
Aruba Access Point 3XX

Console

UART

Main CPU

BLE Chip (CC2540)

UART
OTA OAD OMG

greg@greg-XPS ~/repos/research/aruba-ble
$ python3 shell.py shell 00:1A:7D:DA:71:13 f4:5e:ab:e7:ff:5d
Got handles 64 for uuid b'ff5c73a3e86dd4a318476c6c58e12f27'
Got handles 44 for uuid b'664af1b24f424c3de76e7bb601eaaffa'
Got handles 46 for uuid b'664af1b24f424c3de76e7bb602eaaffa'
Got handles 48 for uuid b'664af1b24f424c3de76e7bb603eaaffa'

~ # route -n
route -n
Kernel IP routing table
<table>
<thead>
<tr>
<th>Destination</th>
<th>Gateway</th>
<th>Genmask</th>
<th>Flags</th>
<th>Metric</th>
<th>Ref</th>
<th>Use</th>
<th>Iface</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.0.0</td>
<td>0.0.0.0</td>
<td>0.0.0.0</td>
<td>U</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>tun0</td>
</tr>
<tr>
<td>0.0.0.0</td>
<td>192.168.1.1</td>
<td>0.0.0.0</td>
<td>UG</td>
<td>-3</td>
<td>0</td>
<td>0</td>
<td>br0</td>
</tr>
<tr>
<td>192.168.1.0</td>
<td>0.0.0.0</td>
<td>255.255.255.0</td>
<td>U</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>br0</td>
</tr>
<tr>
<td>192.168.1.5</td>
<td>0.0.0.0</td>
<td>255.255.255.255</td>
<td>UH</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>tun0</td>
</tr>
<tr>
<td>192.168.11.0</td>
<td>0.0.0.0</td>
<td>255.255.255.255</td>
<td>U</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>br0</td>
</tr>
</tbody>
</table>
What would a BLEEDINGBIT attack look like?
What would a BLEEDINGBIT attack look like?
What would a BLEEDINGBIT attack look like?
What would a BLEEDINGBIT attack look like?
What would a BLEEDINGBIT attack look like?
What would a BLEEDINGBIT attack look like?
Targeting an Aruba AP 325
TI cc2540 BLE Chip
Demonstration

Takeover of Aruba Access Point
BLE Attack surface

**CVE-2018-7080**
Affecting Aruba

**CVE-2018-16986**
TI BLE STACK
Affecting Cisco, Meraki

- Application Layer (App)
- Generic Access Profile (GAP)
- Generic Attribute Protocol (GATT)
- Security Manager (SMP)
- Attribute Protocol (ATT)
- Logical Link Control & Adaptation Protocol (L2CAP)
- Link Layer (LL)
- Physical Layer (PHY)
BLE link layer

**BLE Packet**

<table>
<thead>
<tr>
<th>Preamble</th>
<th>Access Address</th>
<th>Protocol Data Unit (PDU)</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Byte</td>
<td>4 Bytes</td>
<td>2-257 Bytes</td>
<td>3 Bytes</td>
</tr>
</tbody>
</table>

**Advertising Channel PDU**

<table>
<thead>
<tr>
<th>Header</th>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 Bytes</td>
<td>0-37 Bytes</td>
</tr>
</tbody>
</table>
BLE link layer

Advertising Channel PDU

<table>
<thead>
<tr>
<th>Header</th>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 Bytes</td>
<td>0-37 Bytes</td>
</tr>
</tbody>
</table>

Bluetooth® Core Specification version 4.2

<table>
<thead>
<tr>
<th>LSB</th>
<th>MSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDU Type (4 bits)</td>
<td>RFU (2 bits)</td>
</tr>
<tr>
<td>TxAdd (1 bit)</td>
<td>RxAdd (1 bit)</td>
</tr>
<tr>
<td>Length (6 bits)</td>
<td>RFU (2 bits)</td>
</tr>
</tbody>
</table>

Figure 2.3: Advertising channel PDU Header

Bluetooth® Core Specification version 5.0

<table>
<thead>
<tr>
<th>LSB</th>
<th>MSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDU Type (4 bits)</td>
<td>RFU (1 bit)</td>
</tr>
<tr>
<td>ChSel (1 bit)</td>
<td>TxAdd (1 bit)</td>
</tr>
<tr>
<td>RxAdd (1 bit)</td>
<td>Length (8 bits)</td>
</tr>
</tbody>
</table>

Figure 2.5: Advertising channel PDU Header

Length (8 bits)
TI CC2640 Architecture

SimpleLink™ CC26xx wireless MCU

**Main CPU**
- Application Layer (App)
- Generic Access Profile (GAP)
- Generic Attribute Protocol (GATT)
- Security Manager (SMP)
- Attribute Protocol (ATT)
- Logical Link Control & Adaptation Protocol (L2CAP)

**Radio Core**
- Link Layer (LL)
- Physical Layer (PHY)
CC2640 (lack of) Security

NO DEP (NX-BIT)

NO ASLR

NO MEMORY MANAGEMENT

K

U
void llGetAdvChanPDU(uint8 *pduType, uint8 *isTxAddress,  
  uint8 *advAddr, uint8 *dataLen,  
  uint8 *advData, int8 *rssi)
{
  dataEntry_t *dataEntry;
  uint8 pktLength;
  uint8 *pktData;
  ...
  dataEntry = RFHAL_GetNextDataEntry(scanParam.pRxQ);
  ...
  pktLength = dataEntry.data[1];
  pktData = &(dataEntry.data[2]);  //Skip the 2 byte header
  *dataLen = pktLength - 6;
```c
void llGetAdvChanPDU(uint8 *pduType, uint8 *isTxAddress,
                      uint8 *advAddr, uint8 *dataLen,
                      uint8 *advData, int8 *rssi)
{
    dataEntry_t *dataEntry;
    uint8 pktLength;
    uint8 *pktData;
    ...
    dataEntry = RFHAL_GetNextDataEntry(scanParam.pRxQ);
    ...
    pktLength = dataEntry.data[1];
    pktData = &(dataEntry.data[2]); //Skip the 2 byte header
    *dataLen = pktLength - 6;
}
```
if ((signed int)*dataLen >= 32) // Check for bad size
halAssertHandler();
CC2640 Memory Corruption

```c
if ((signed int)*dataLen >= 32) // Check for bad size
    halAssertHandler();
```

```c
halAssertHandler
BX LR
; End of function halAssertHandler
```
// Copy address from packet
for (i = 0; i < 6; ++i)
{
    *advAddr++ = *pktData++;
}
...

// Parse packet header, convert packet type to pduType
enum
...

// Copy the rest of the packet
for (i = 0; i < (unsigned int)*dataLen; ++i)
{
    *advData++ = *pktData++;
}
...
<table>
<thead>
<tr>
<th>RFU</th>
<th>Length</th>
<th>Actual payload size</th>
<th>Crash?</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>1111111</td>
<td>(255)</td>
<td>255</td>
</tr>
<tr>
<td>RFU</td>
<td>Length</td>
<td>Actual payload size</td>
<td>Crash?</td>
</tr>
<tr>
<td>-----</td>
<td>---------------</td>
<td>---------------------</td>
<td>---------</td>
</tr>
<tr>
<td>11</td>
<td>111111 (255)</td>
<td>255</td>
<td>🚫</td>
</tr>
<tr>
<td>00</td>
<td>000001 (1)</td>
<td>1</td>
<td>🚫</td>
</tr>
</tbody>
</table>
Let's try and crash it

<table>
<thead>
<tr>
<th>RFU</th>
<th>Length</th>
<th>Actual payload size</th>
<th>Crash?</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>111111 (255)</td>
<td>255</td>
<td>✗</td>
</tr>
<tr>
<td>00</td>
<td>000001 (1)</td>
<td>1</td>
<td>✗</td>
</tr>
<tr>
<td>00</td>
<td>111111 (63)</td>
<td>63</td>
<td>✗</td>
</tr>
<tr>
<td>RFU</td>
<td>Length</td>
<td>Actual payload size</td>
<td>Crash?</td>
</tr>
<tr>
<td>-----</td>
<td>---------</td>
<td>---------------------</td>
<td>--------</td>
</tr>
<tr>
<td>11</td>
<td>111111</td>
<td>(255)</td>
<td>255</td>
</tr>
<tr>
<td>00</td>
<td>000001</td>
<td>(1)</td>
<td>1</td>
</tr>
<tr>
<td>00</td>
<td>111111</td>
<td>(63)</td>
<td>63</td>
</tr>
<tr>
<td>00</td>
<td>100101</td>
<td>(37)</td>
<td>37</td>
</tr>
</tbody>
</table>
### Lets try and crash it

<table>
<thead>
<tr>
<th>RFU</th>
<th>Length</th>
<th>Actual payload size</th>
<th>Crash?</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>111111 (255)</td>
<td>255</td>
<td>✗</td>
</tr>
<tr>
<td>00</td>
<td>000001 (1)</td>
<td>1</td>
<td>✗</td>
</tr>
<tr>
<td>00</td>
<td>111111 (63)</td>
<td>63</td>
<td>✗</td>
</tr>
<tr>
<td>00</td>
<td>100101 (37)</td>
<td>37</td>
<td>✗</td>
</tr>
<tr>
<td>10</td>
<td>100101 (165)</td>
<td>37</td>
<td>✓</td>
</tr>
</tbody>
</table>
"If StrictLenFilter is 1, only length fields compliant with the Bluetooth low energy specification are considered valid. For an ADV_DIRECT_IND, valid means a length field of 12, and for other ADV*_IND messages valid means a length field in the range from 6 to 37."
void llGetAdvChanPDU(...) {
  dataEntry_t *dataEntry;
  uint8 pktLength;
  uint8 *pktData;
  ...
  dataEntry = RFHAL_GetNextDataEntry(RxQ);
  ...
  pktLength = dataEntry.data[1];
}

signed int parse_and_validate_packet_header(...) {
  int packet_len;
  int pduType;
  ...
  // Radio waits for syncword
  ...
  pkt_first_word = RF_read_word();
  ...
  pduType = pkt_first_word & 0xF;
  ...
  // advLenMask == 0x3F (0b00111111)
  // maxAdvPktLen == 0x25 (37)
  packet_len = ((uint8)pkt_first_word & advLenMask);
  ...
  if ( packet_len_extracted > maxAdvPktLen )
    return -1; // Failed
}
Case Study

CISCO AP1815W

J TAG Header
CC2640 Memory Corruption

```c
void llGetAdvChanPDU(...) {
    dataEntry_t *dataEntry;
    uint8 pktLength;
    uint8 *pktData;
    ...
    // Copy the rest of the packet
    for (i = 0; i < (unsigned int)*dataLen; ++i) {
        *advData++ = *pktData++;
    }
    ...
}
```
<table>
<thead>
<tr>
<th>Address</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x20004488</td>
<td>Advertising incoming packet</td>
</tr>
<tr>
<td>0x200044B0</td>
<td>Task IDs</td>
</tr>
<tr>
<td></td>
<td>hciGapTaskID</td>
</tr>
<tr>
<td></td>
<td>hciL2capTaskID</td>
</tr>
<tr>
<td></td>
<td>hciSmpTaskID</td>
</tr>
<tr>
<td></td>
<td>hciExtTaskID</td>
</tr>
<tr>
<td></td>
<td>bleDispatch_TaskID</td>
</tr>
<tr>
<td>0x200044B5</td>
<td>GAP Outgoing response</td>
</tr>
<tr>
<td></td>
<td>rspBuf</td>
</tr>
<tr>
<td>0x200044F0</td>
<td>System timers list pointer</td>
</tr>
<tr>
<td></td>
<td>timerHead</td>
</tr>
<tr>
<td>0x200044F4</td>
<td>Last system clock timestamp</td>
</tr>
<tr>
<td></td>
<td>osal_last_timestamp</td>
</tr>
<tr>
<td>0x200044F8</td>
<td>System clock</td>
</tr>
<tr>
<td></td>
<td>osal_systemClock</td>
</tr>
<tr>
<td>0x200044FC</td>
<td>Function pointers</td>
</tr>
<tr>
<td></td>
<td>ICall_dispatcher</td>
</tr>
<tr>
<td></td>
<td>ICall_enterCriticalSection</td>
</tr>
<tr>
<td></td>
<td>ICall_exitCriticalSection</td>
</tr>
</tbody>
</table>

What is being overwritten?
<table>
<thead>
<tr>
<th>Function pointers</th>
<th>ICall_dispatcher</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ICall_enterCriticalSection</td>
</tr>
<tr>
<td></td>
<td>ICall_exitCriticalSection</td>
</tr>
</tbody>
</table>

What is being overwritten?

0x200044FC
Attacking an Cisco AP
We see the AP beaconsing: "Your APs BLEong to us"
BLEEDING BIT

Demonstration
Take Over of BLE Chip in Cisco Access Point
Take aways

- BLE Radio chips can be vulnerable to attack
- Vulnerabilities in peripheral chips can lead to network breach
- Access points and network infra devices are also unmanaged devices
Questions?

For more info & whitepaper: https://armis.com/bleedingbit