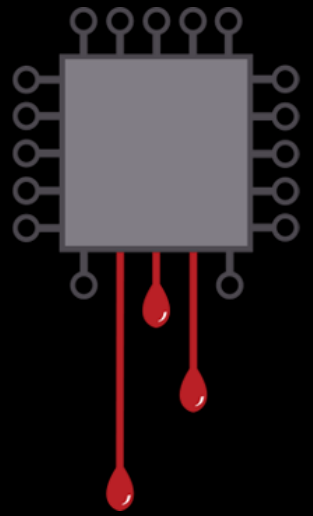
BLEEDING  BIT

(All) Your APs (Are) Belong to Us

Ben Seri, VP Research
Dor Zusman, Researcher

ARMIS

black hat WEBCASTS

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

BLE in recent news

WIRED SUBSCRIBE

ANDY GREENBERG SECURITY 08.05.19 02:14 PM

THE CLEVER CRYPTOGRAPHY BEHIND APPLE'S 'FIND MY' FEATURE

NGUYEN HUU DUC DUANG/GETTY IMAGES

SHARE

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—to the sufficiently paranoid, at least—like both a physical security innovation and a potential privacy disaster. But while security experts immediately wondered whether

WIRED SUBSCRIBE

LILY HAY NEWMAN SECURITY 08.15.19 02:17 PM

GOOGLE WILL REPLACE TITAN SECURITY KEY OVER A BLUETOOTH FLAW

GOOGLE

SHARE

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

Why do APs support BLE?

INDOOR NAVIGATION

MEDICAL ASSET TRACKING

RETAIL CUSTOMER TRACKING

SMART SENSORS

But Why????

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)

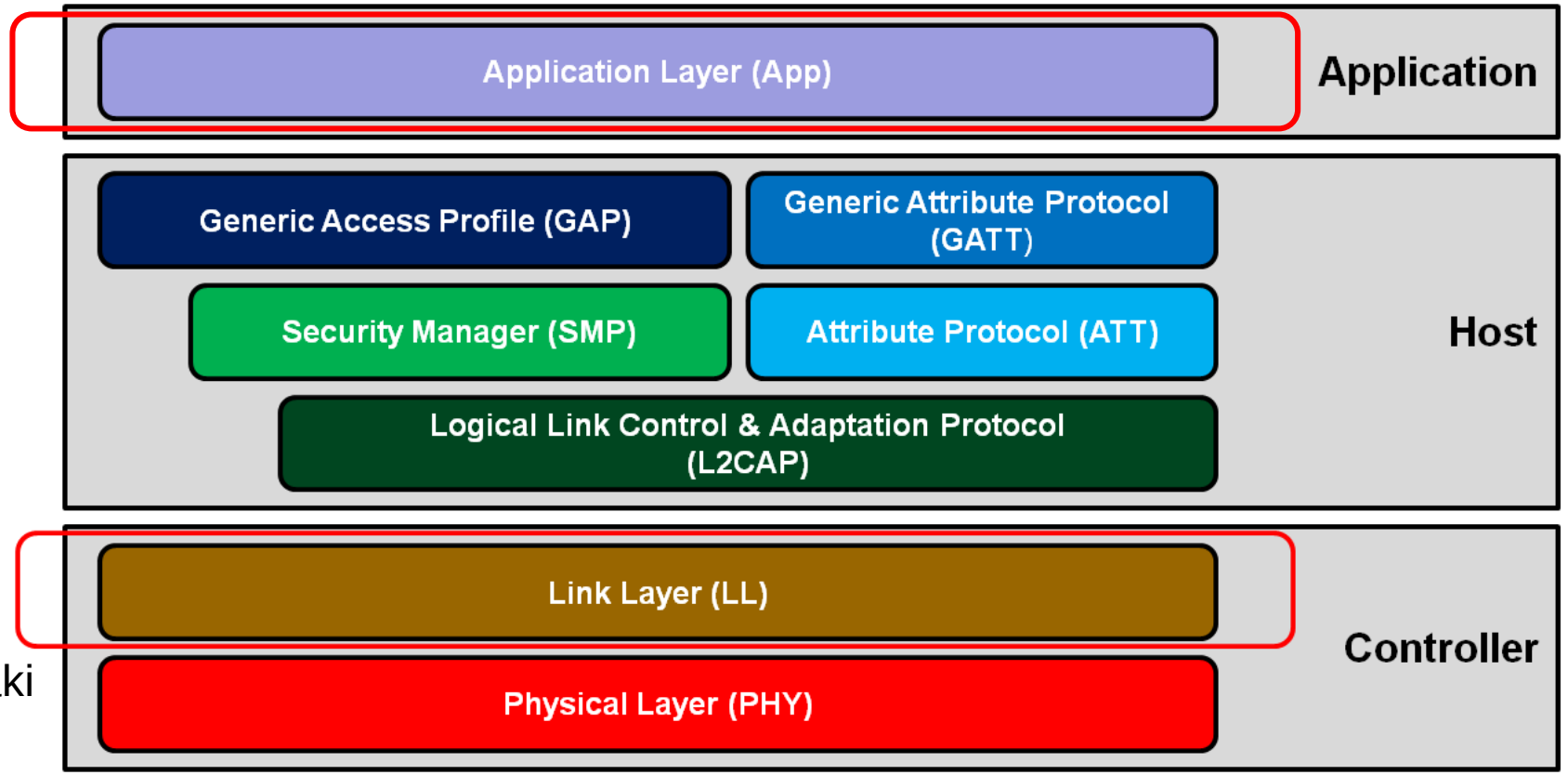
Obviously it's there to increase the attack area. Duh.

Parent Share

BLE Attack surface

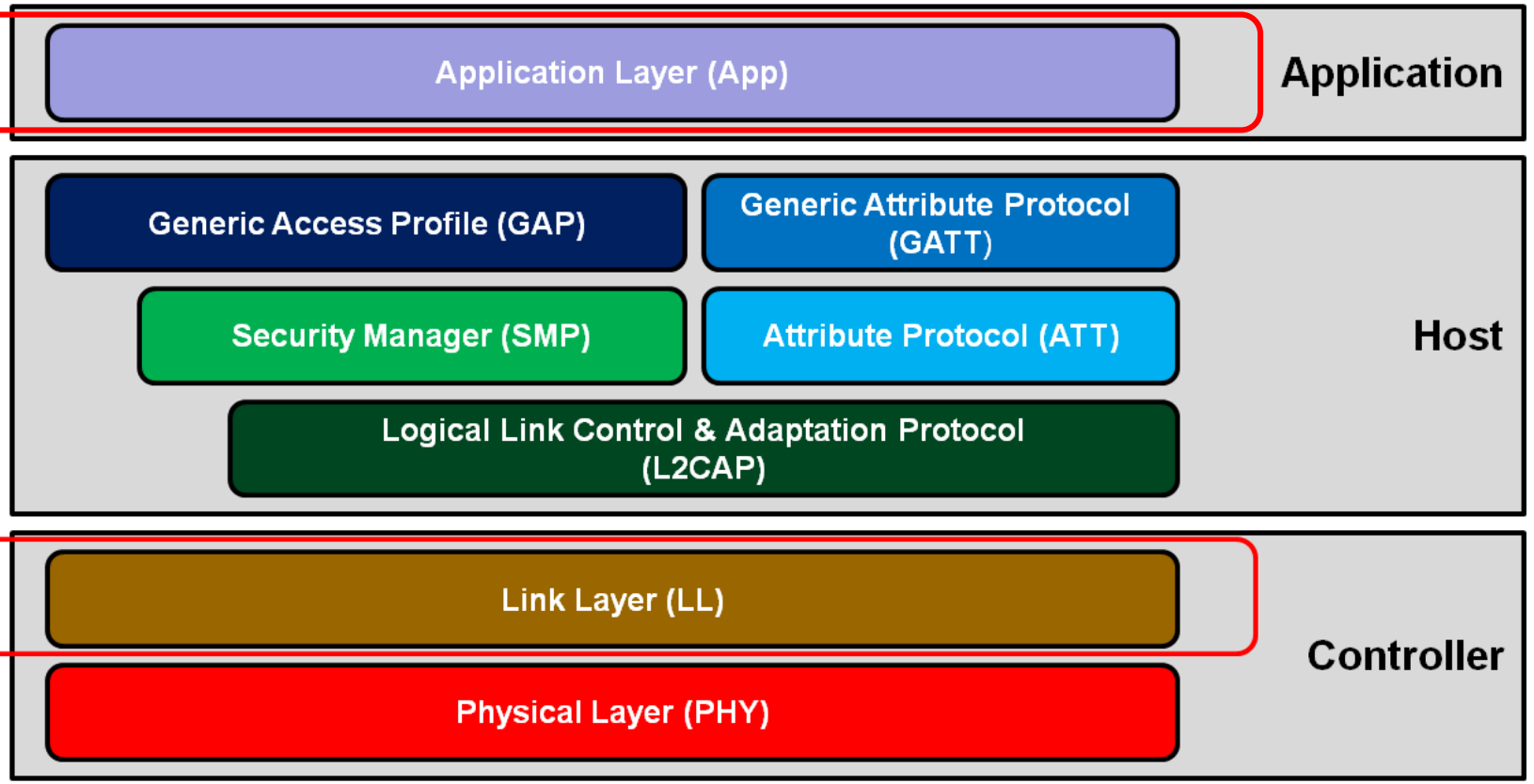
CVE-2018-7080
Affecting Aruba

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



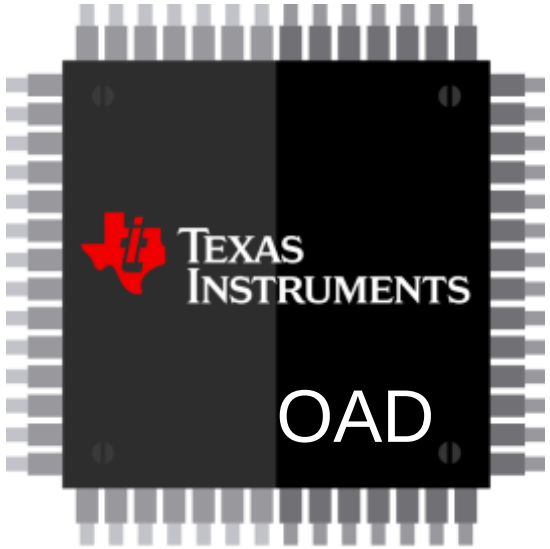
BLE Attack surface

CVE-2018-7080
Affecting Aruba



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

OTA solutions over BLE - The challenges

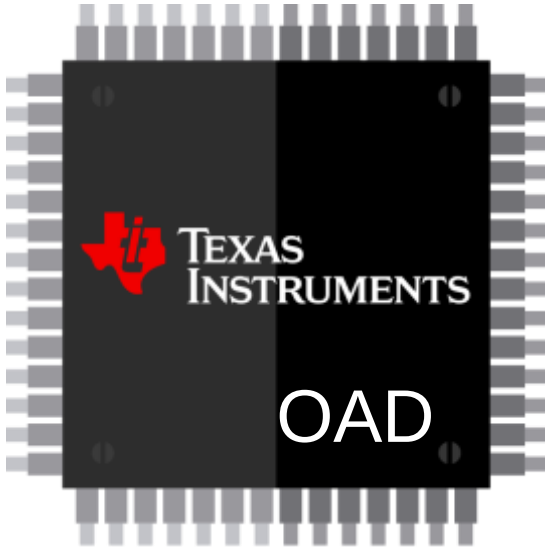


Capturing firmware over the air?

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

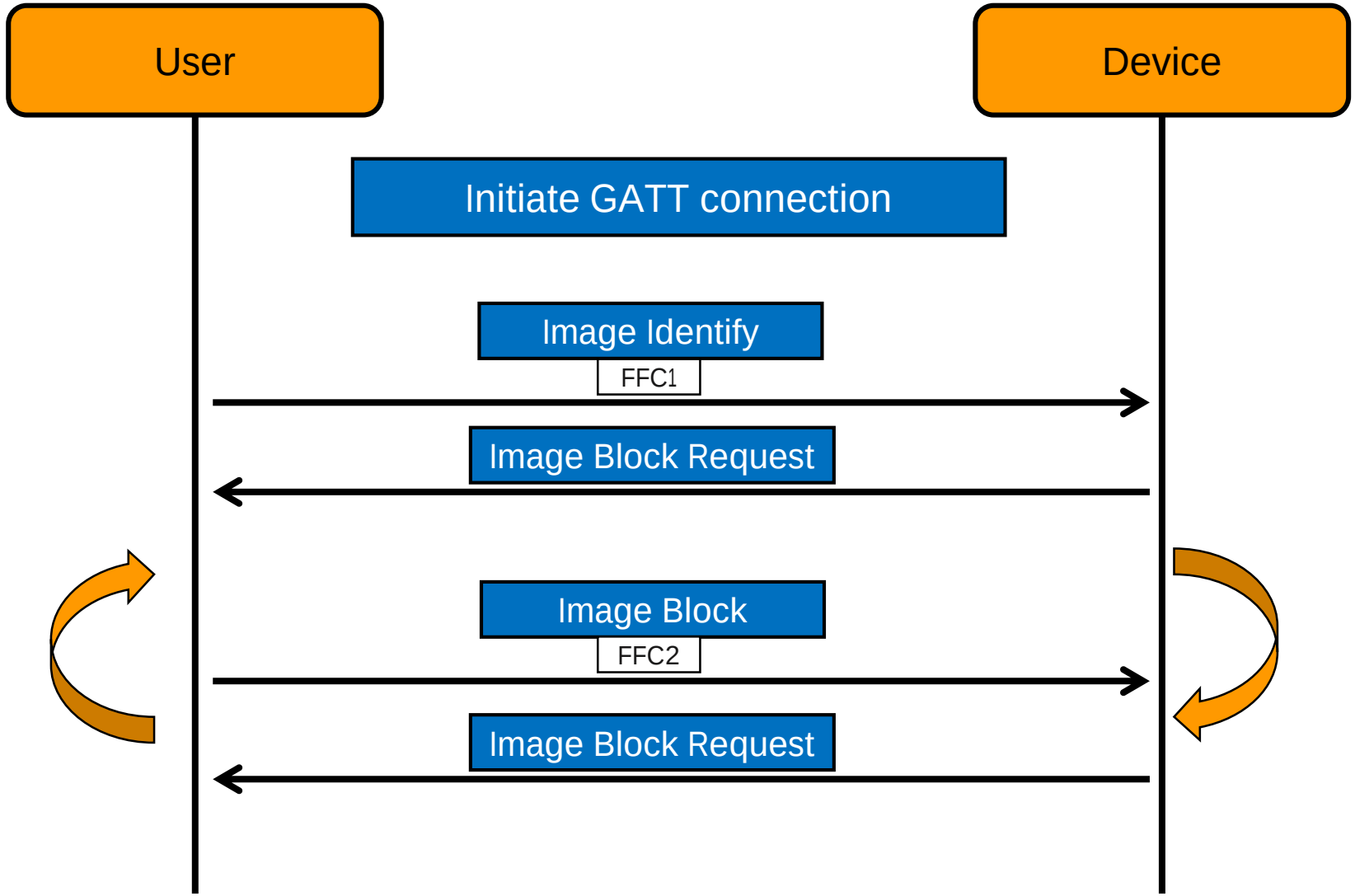
GATT connection is unauthenticated

Firmware integrity is not validated, or uses weak cryptographic signature

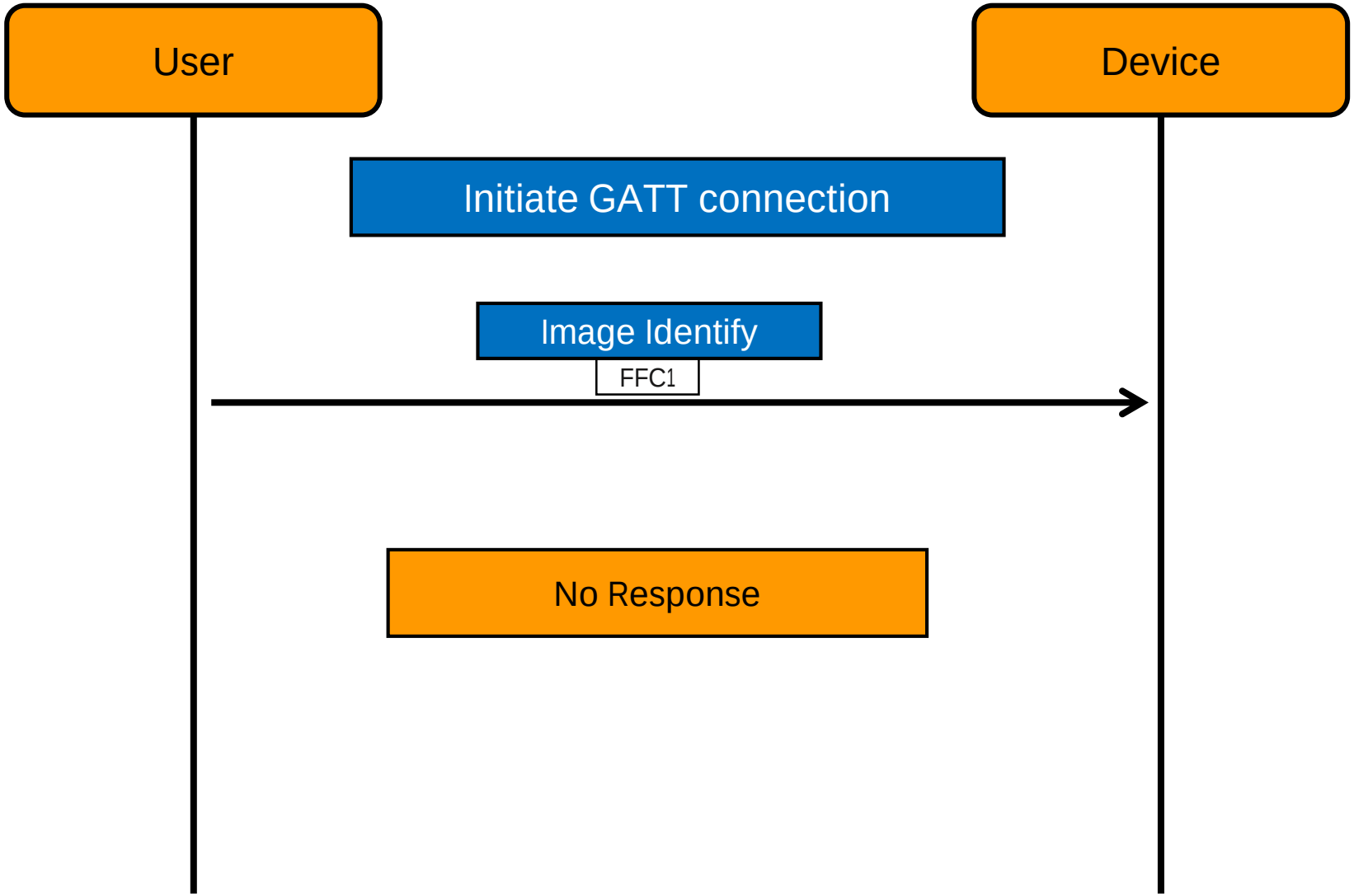
BLE in Aruba Access Points

```
$ 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: f000ffc0-0451-4000-b000-000000000000

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

attr handle = 0x0032, end grp handle = 0xffff
uuid: 272fe150-6c6c-4718-a3d4-6de8a3735cff
```

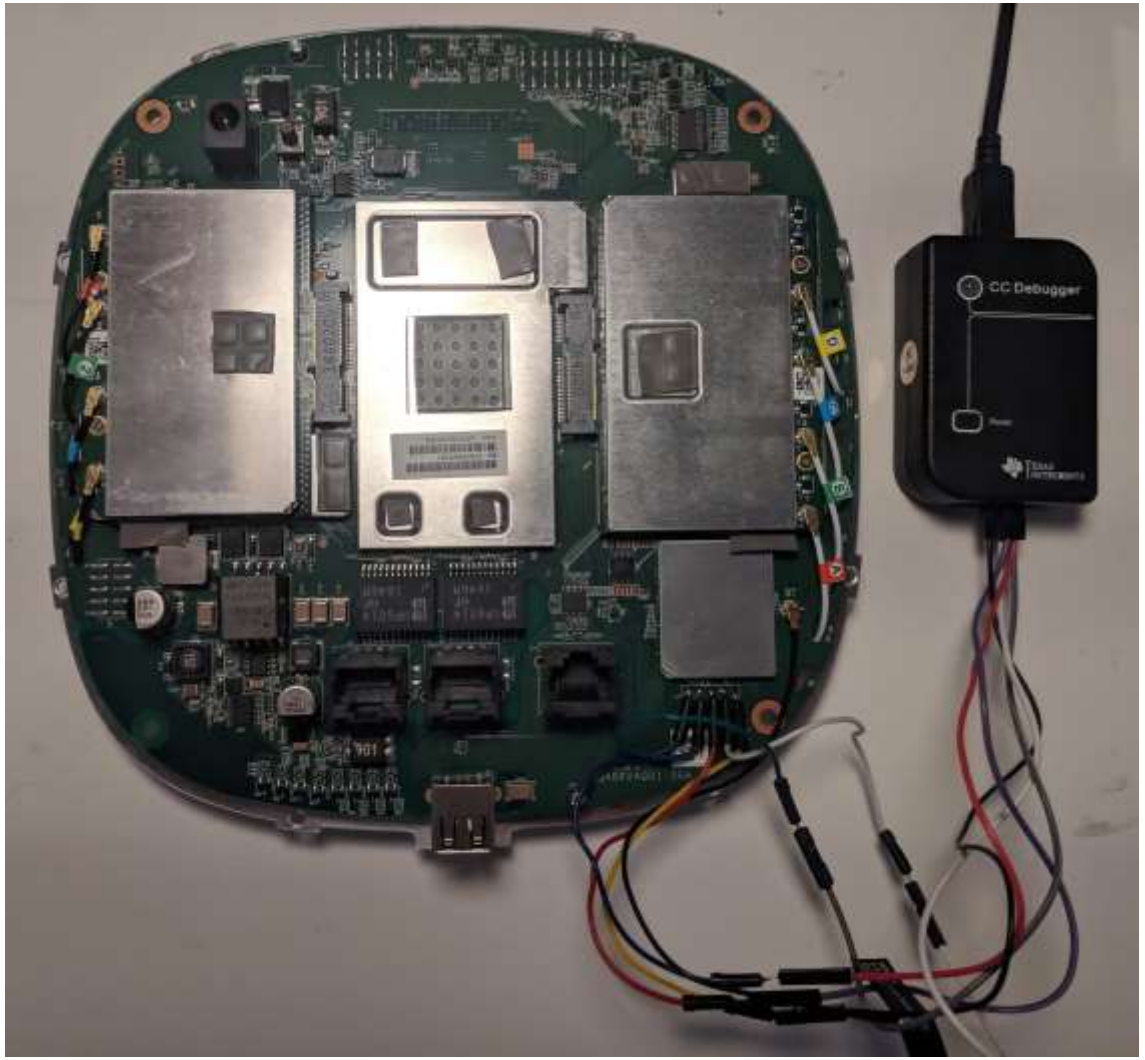
OAD in General



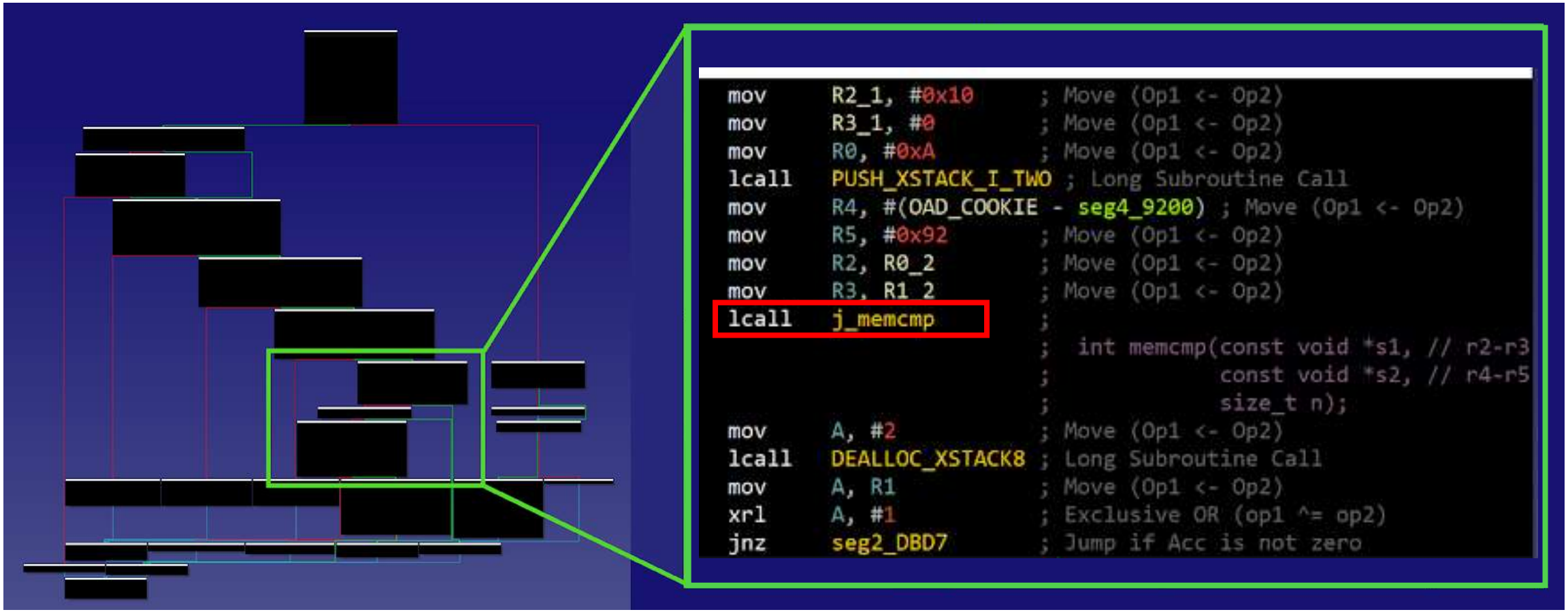
OAD in Aruba Access Points



Extracting BLE firmware



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_DBD7         ; Jump if Acc is not zero
```

Analyzing custom OAD

```
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);
 }
...
}
```

TI's OAD

Analyzing custom OAD

```
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);
 }
...
}
```

TI's OAD

```
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;
   }
 }
 ...
}
```

Aruba's OAD

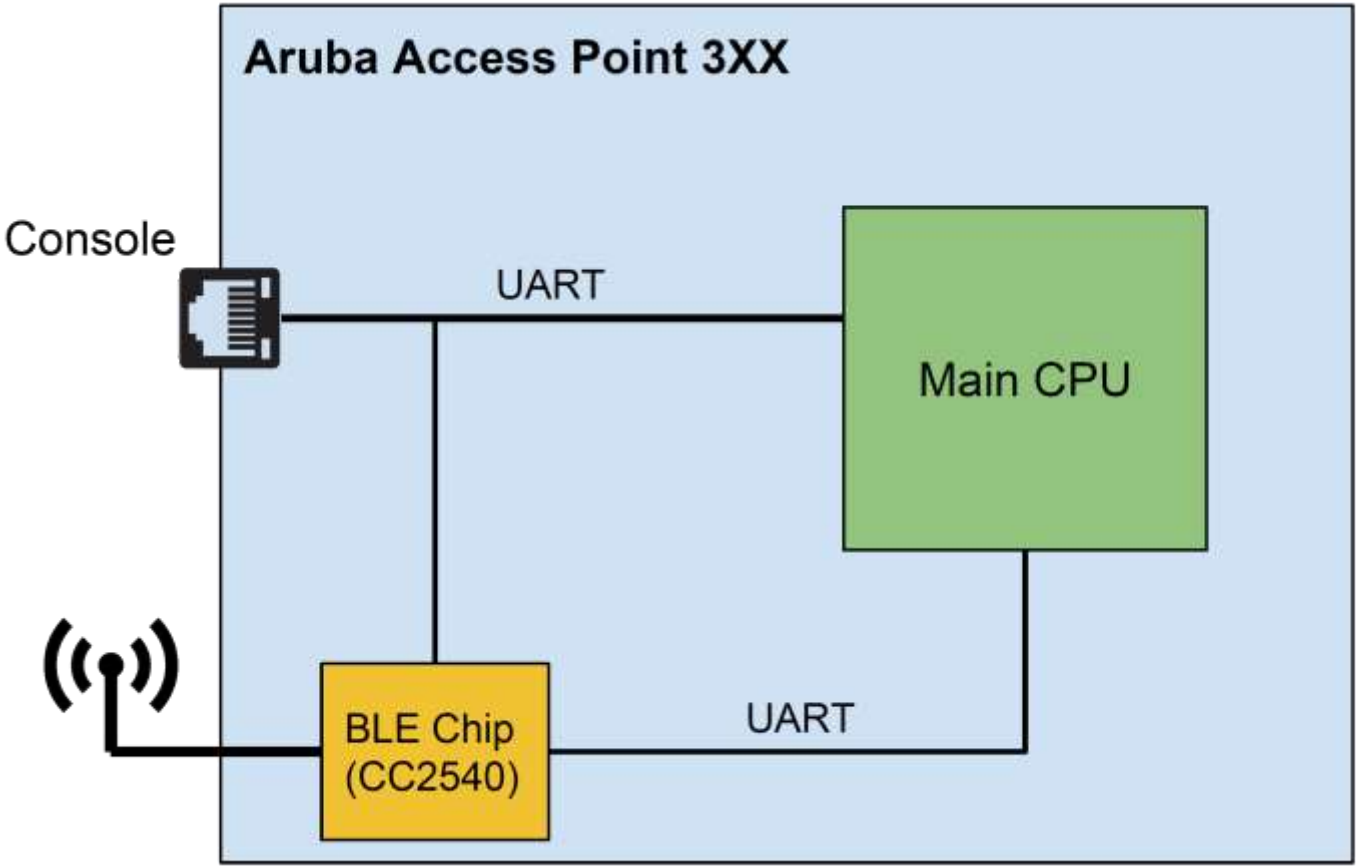
Analyzing custom OAD

```
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;
   }
 }
 ...
}
```

Aruba's OAD


SHHHH...
MY SECRET PASSWORD IS
MEOW1234

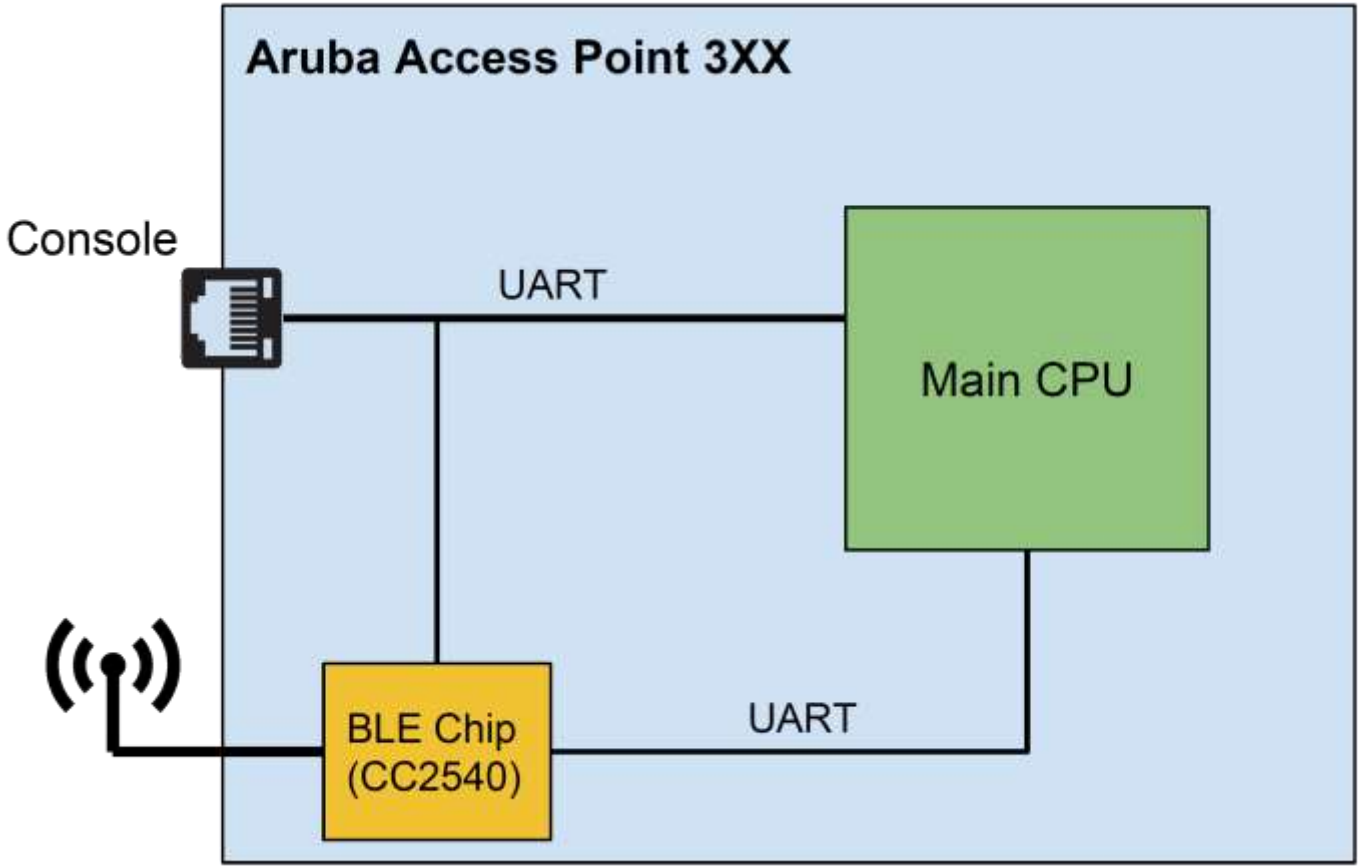
OTA OAD OMG


Aruba Access Point 3XX
Console
UART
Main CPU
BLE Chip
(CC2540)
UART

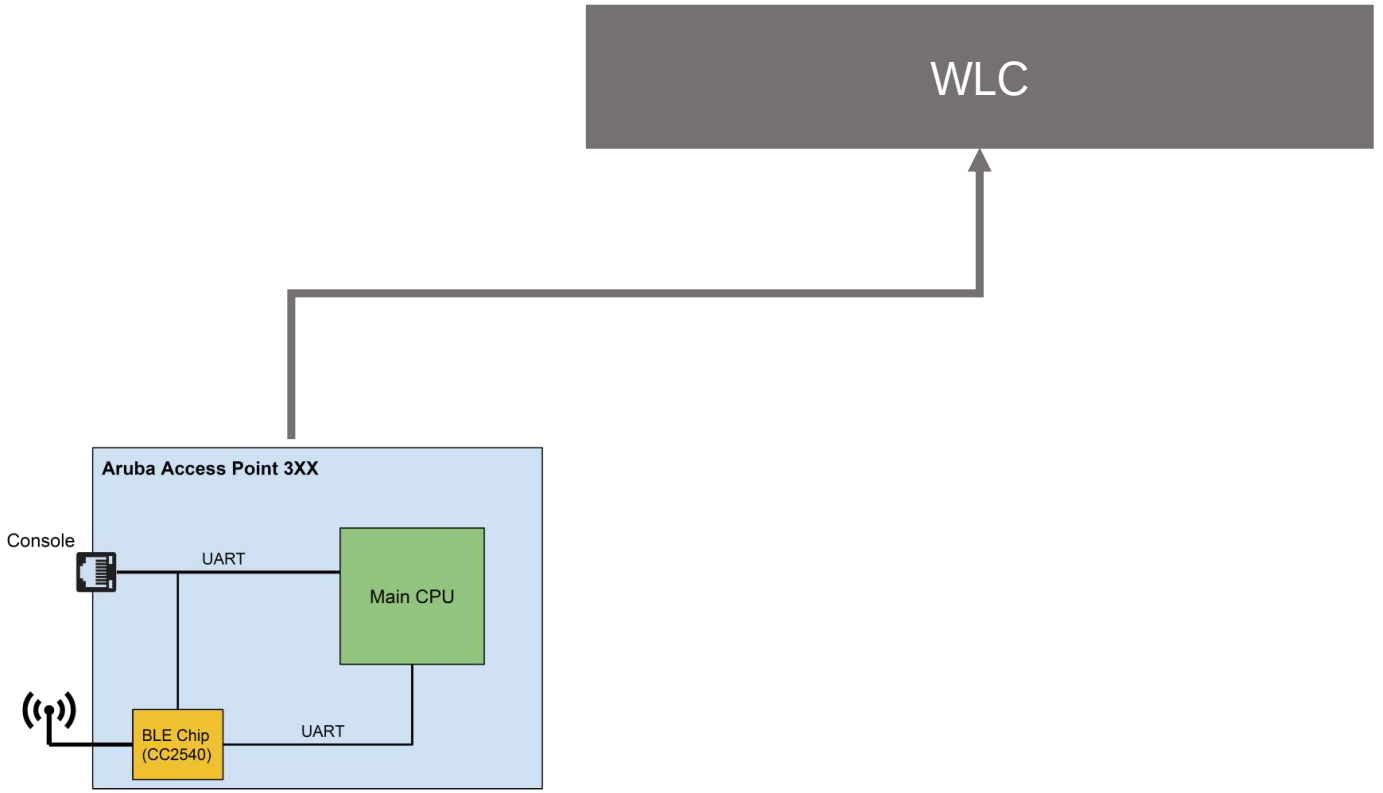
OTA OAD OMG

```
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
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         0.0.0.0         0.0.0.0         U     0      0        0 tun0
0.0.0.0         192.168.1.1     0.0.0.0         UG    -3     0        0 br0
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 br0
192.168.1.5     0.0.0.0         255.255.255.255 UH    0      0        0 tun0
192.168.11.0    0.0.0.0         255.255.255.0   U     0      0        0 br0
~ #

~ #
```

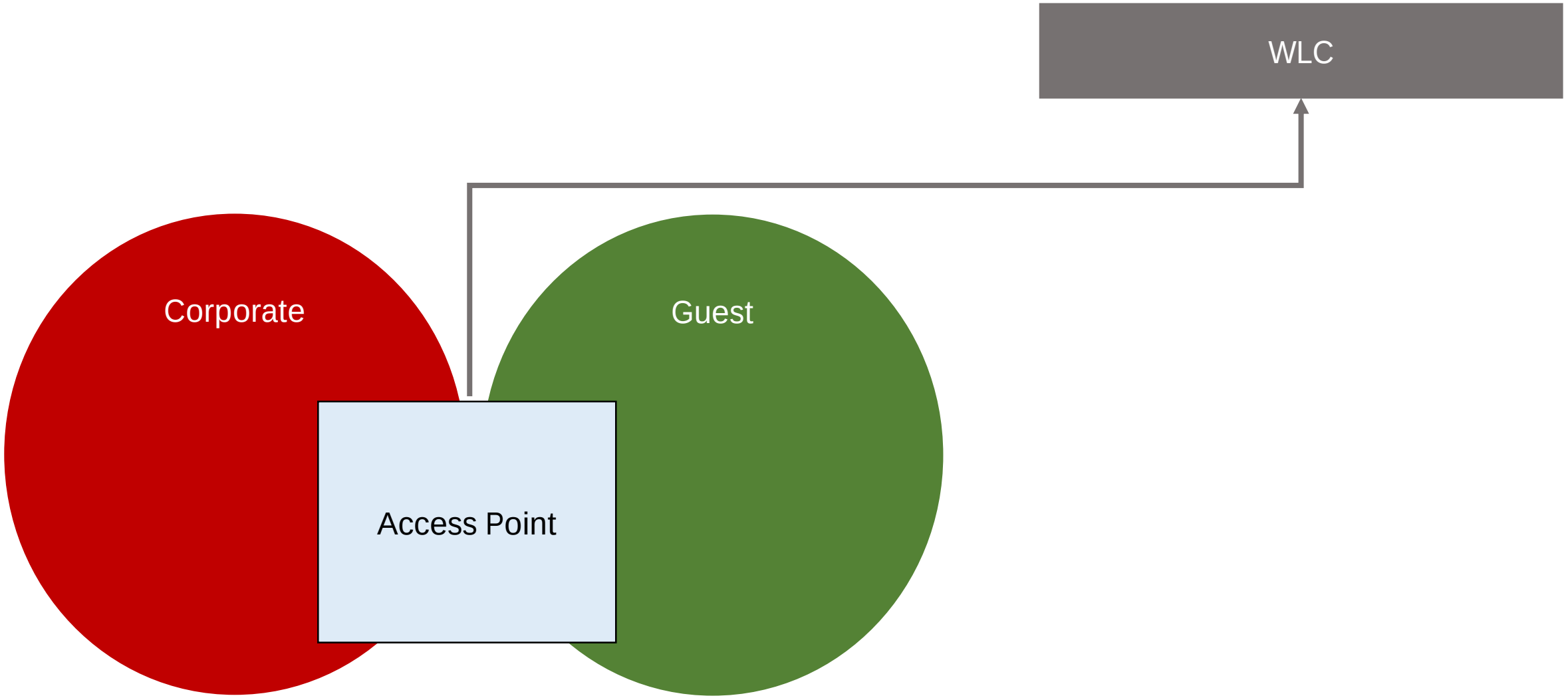
What would a BLEEDINGBIT attack look like?



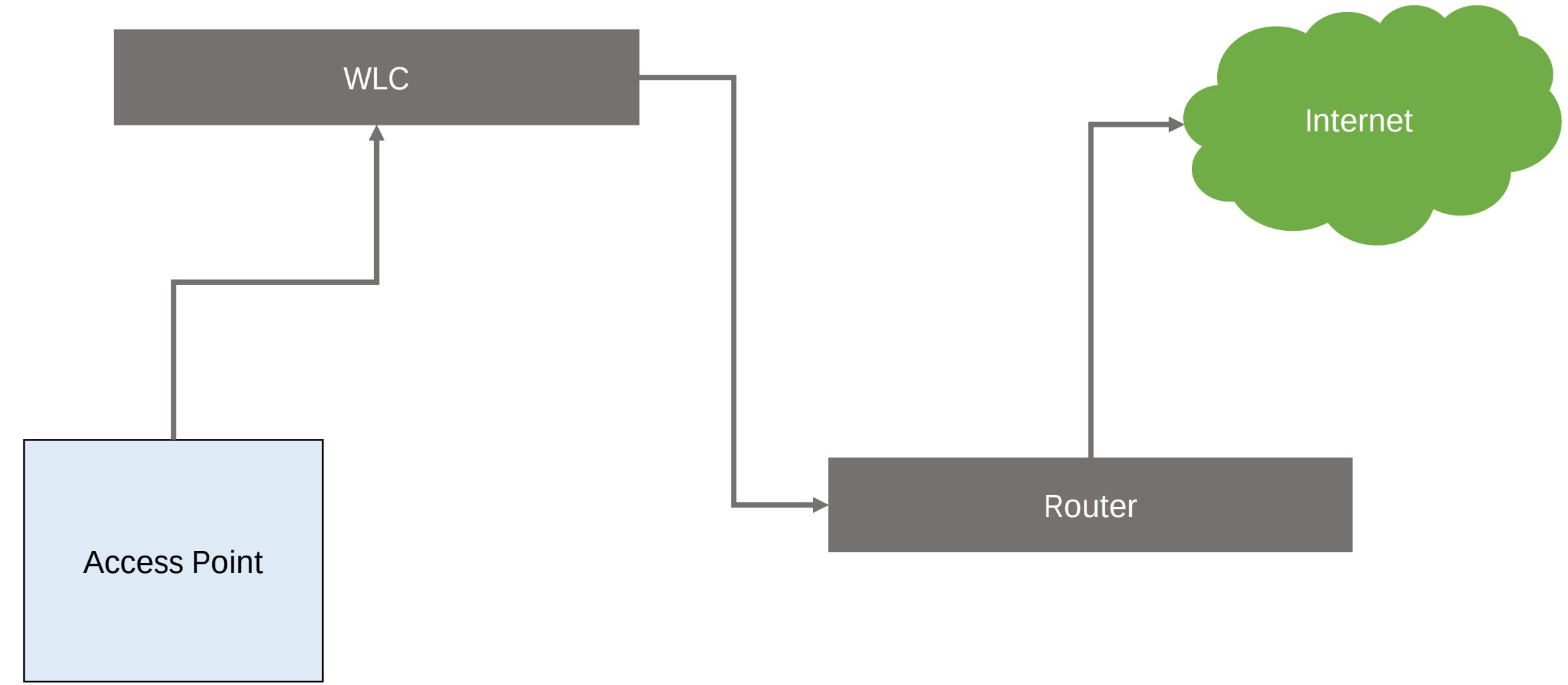
What would a BLEEDINGBIT attack look like?



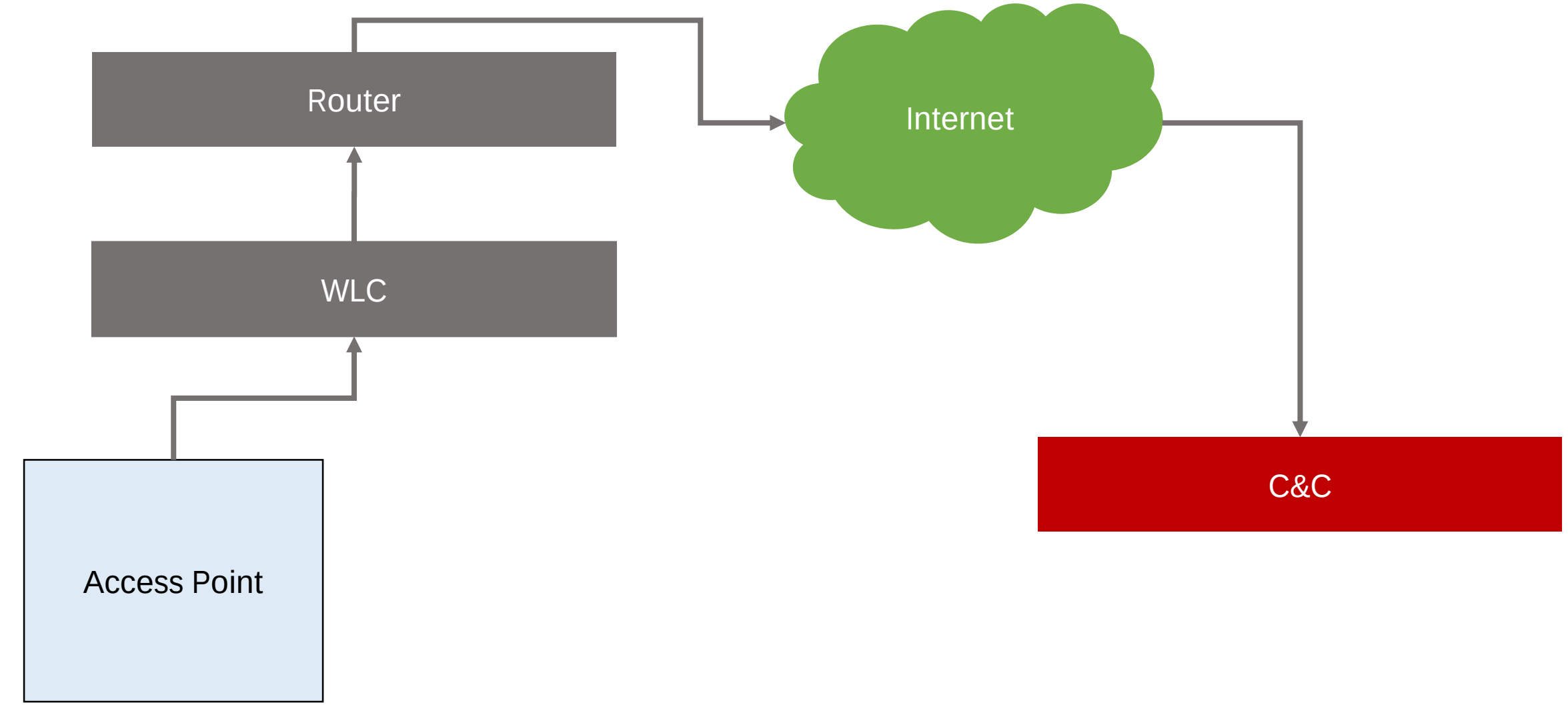
What would a BLEEDINGBIT attack look like?



What would a BLEEDINGBIT attack look like?



What would a BLEEDINGBIT attack look like?



What would a BLEEDINGBIT attack look like?

Router

Internet

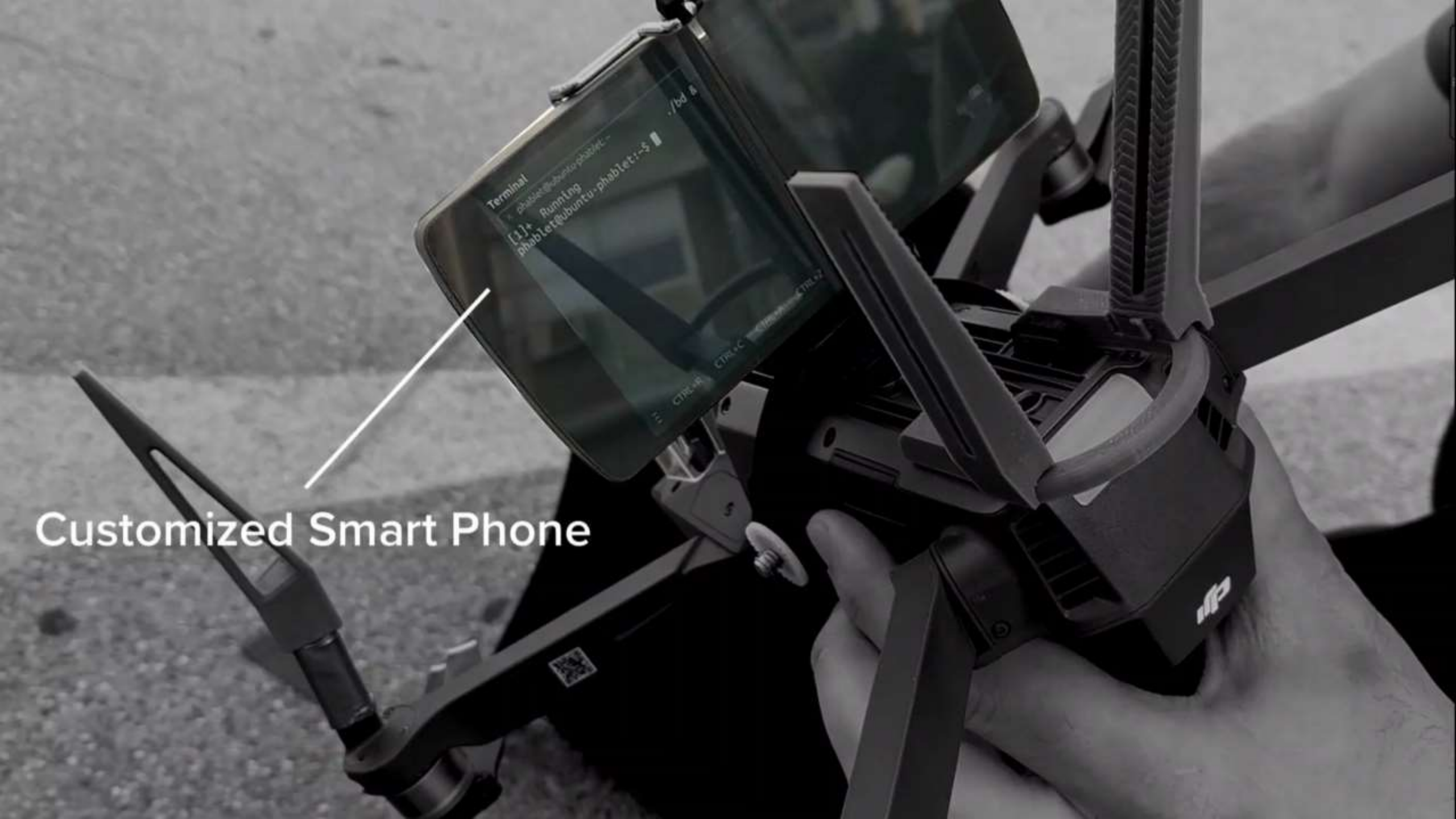
Video

C&C

Access Point

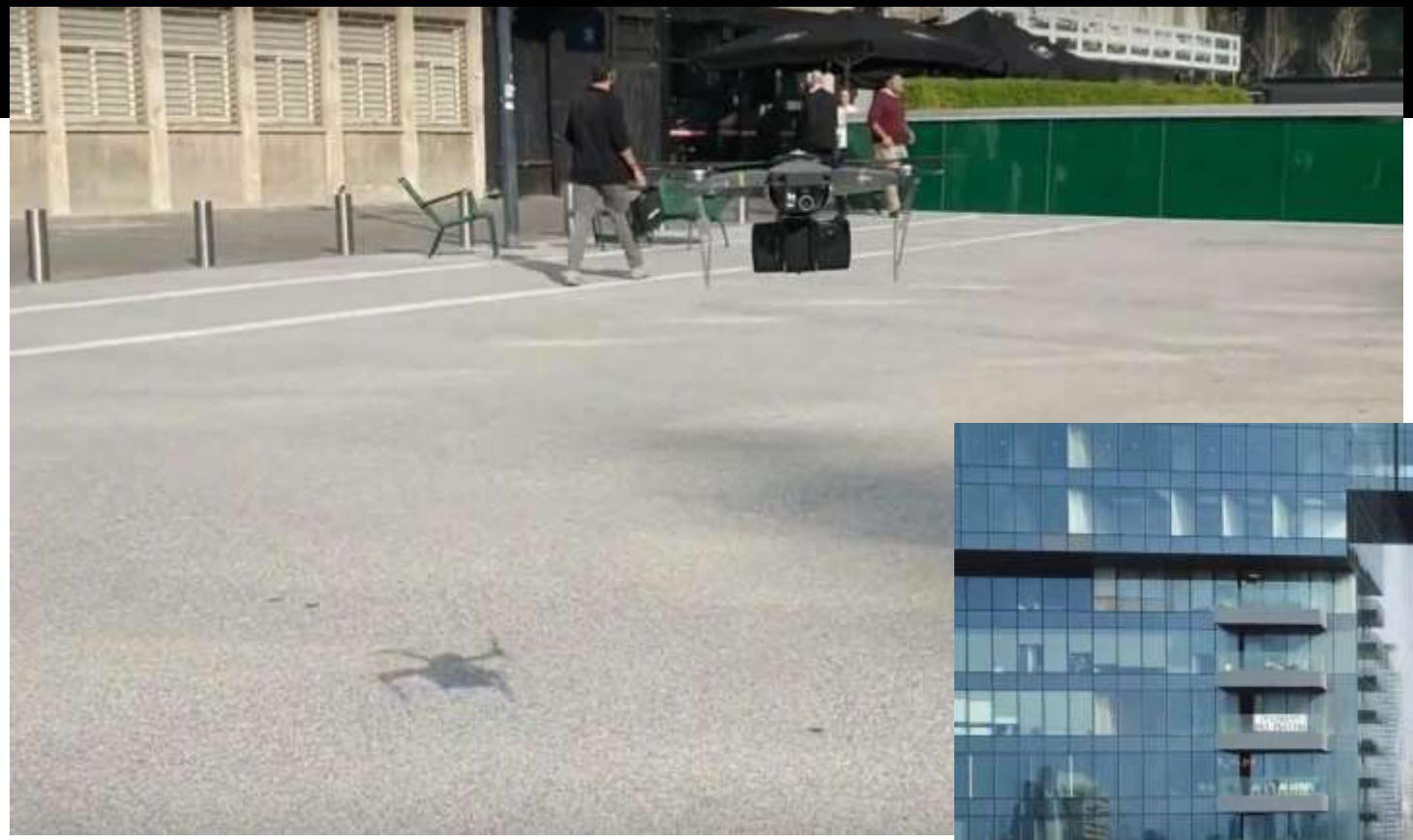
ARMIS | BLEEDINGBIT

The Drone



Customized Smart Phone


The Hacker



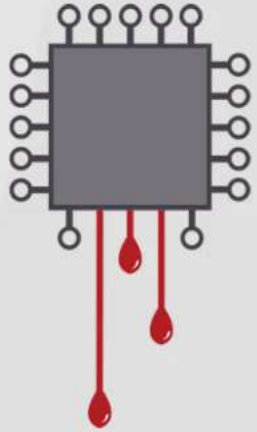

להשכרה
שטחי משרד
אחרונים


NITSBA TOWER


Drone With
Smart Phone

Targeting an
Aruba AP 325
TI cc2540 BLE Chip

BLEEDING BIT



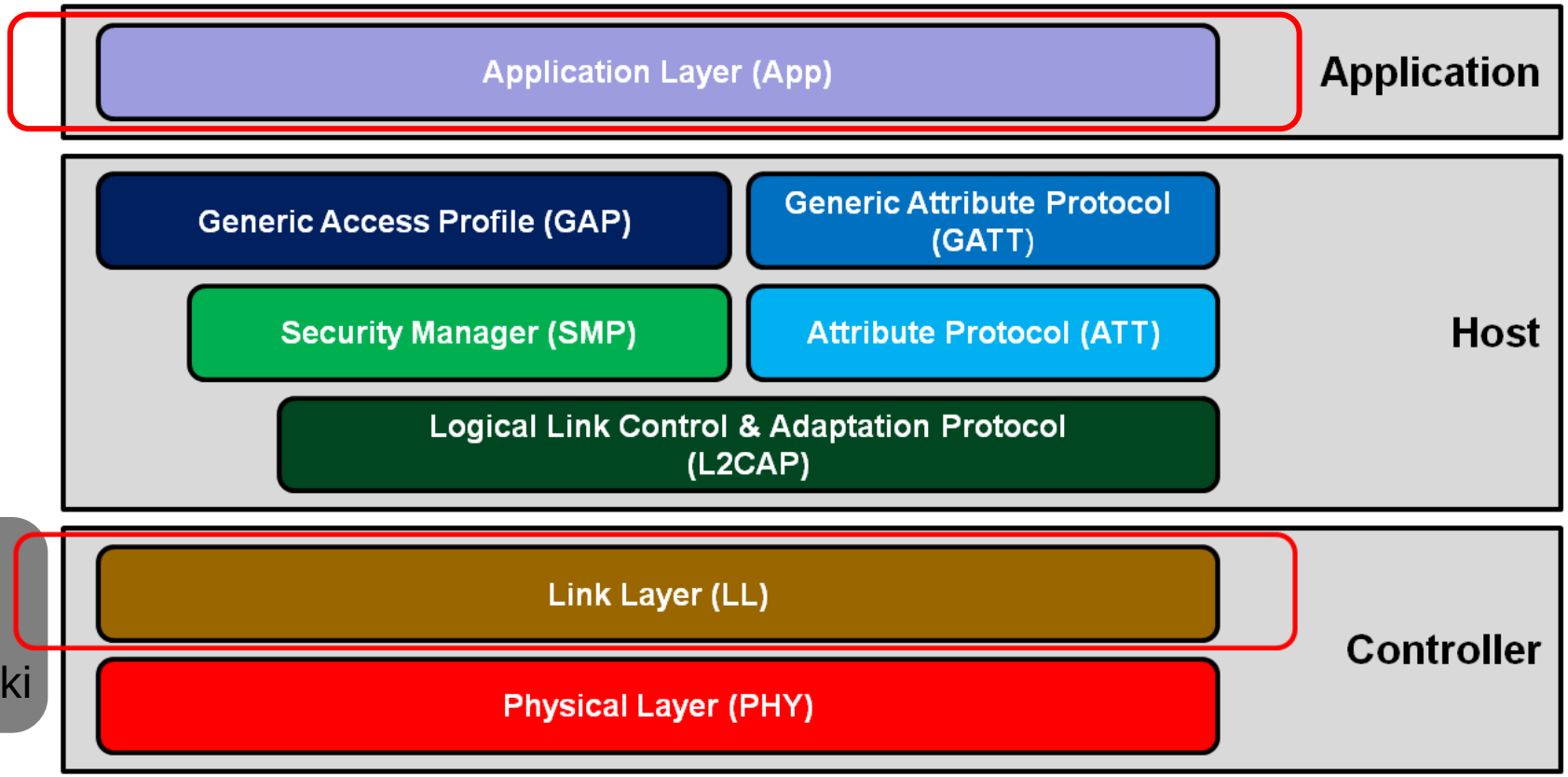
Demonstration

Takeover of Aruba Access Point

armis

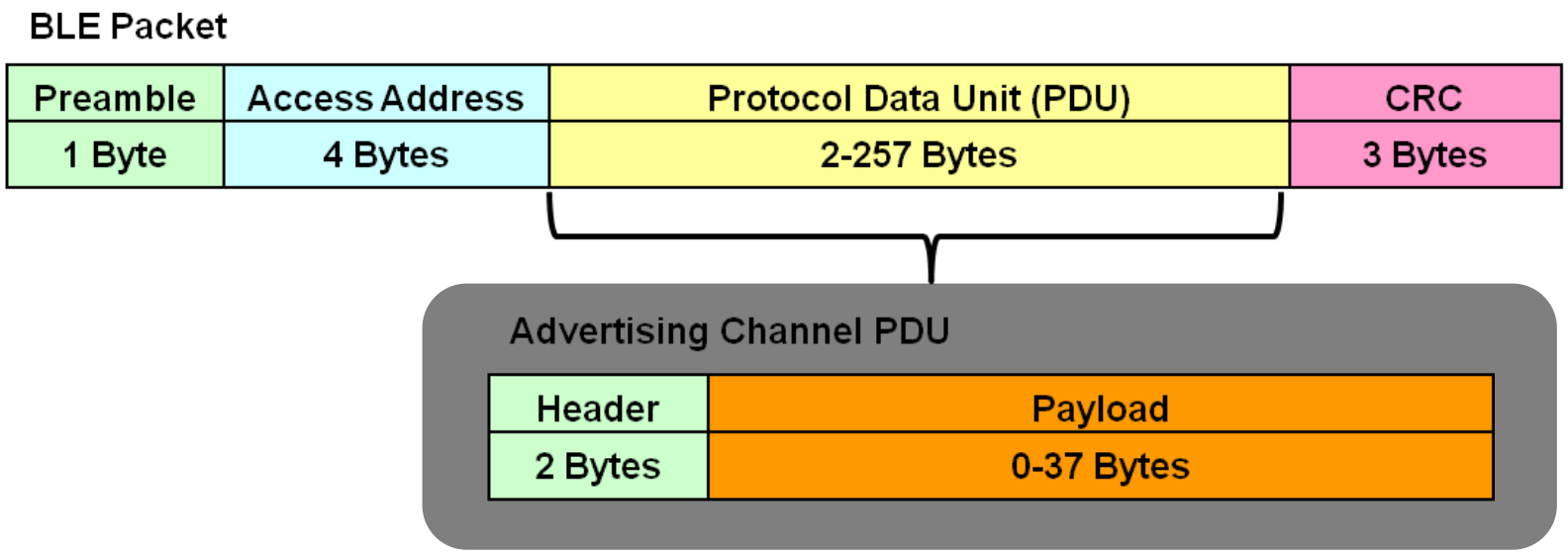
BLE Attack surface

CVE-2018-7080
Affecting Aruba

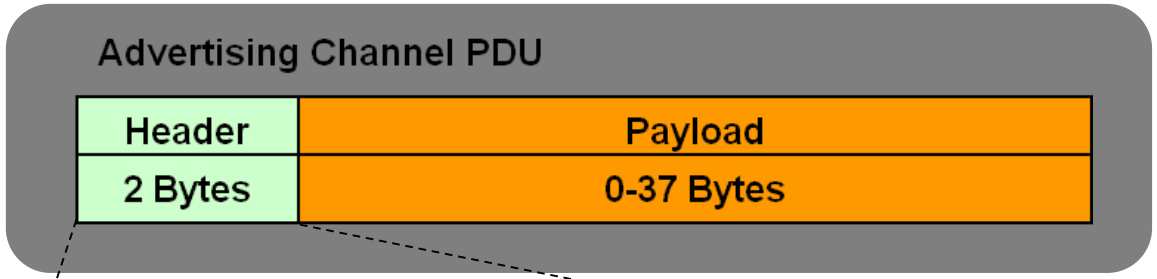


CVE-2018-16986
TI BLE STACK
Affecting Cisco, Meraki

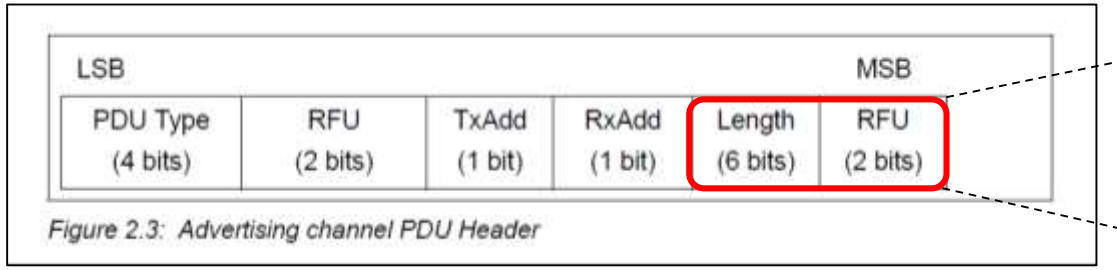
BLE link layer

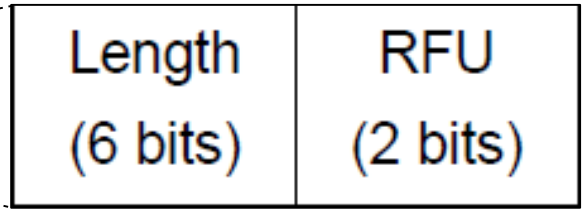


BLE link layer

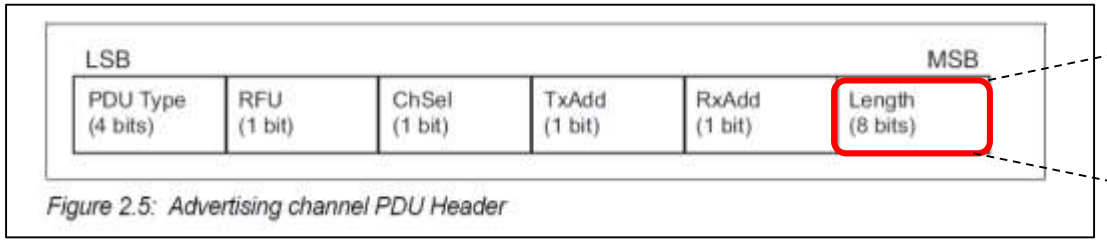


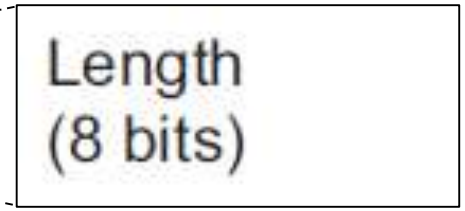
Bluetooth® Core Specification version 4.2



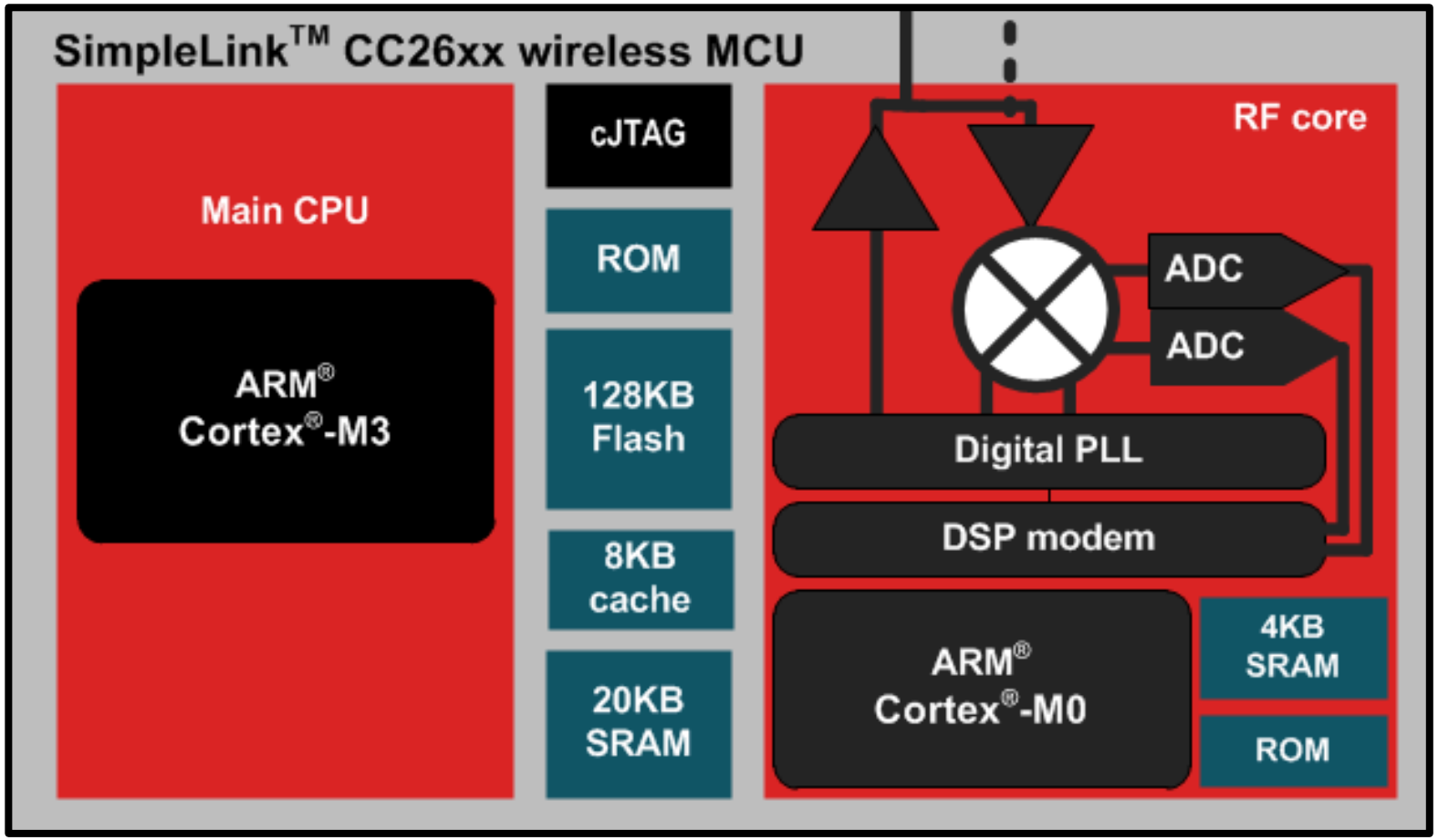


Bluetooth® Core Specification version 5.0

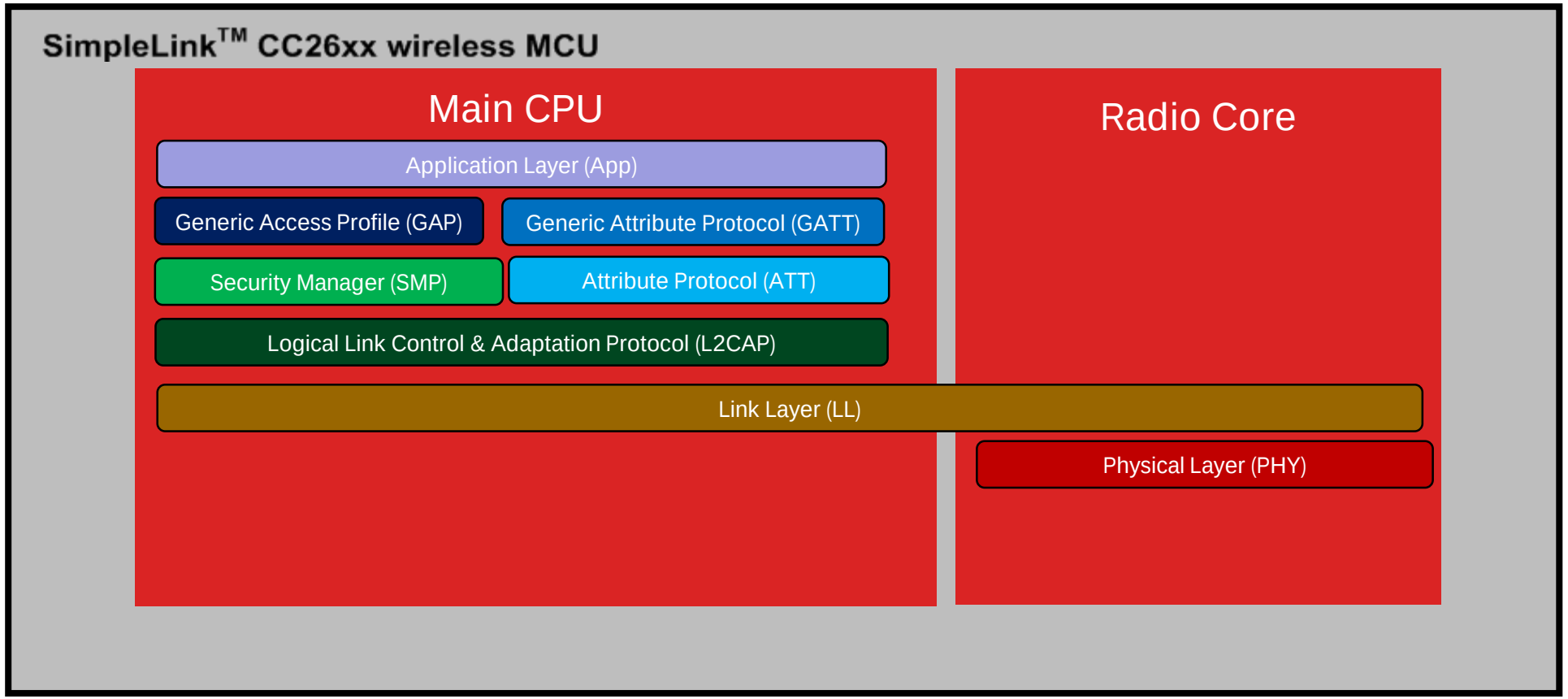




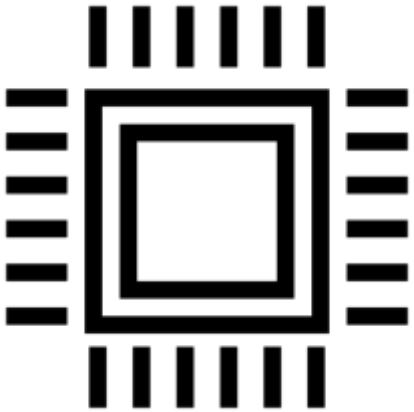
TI CC2640 Architecture



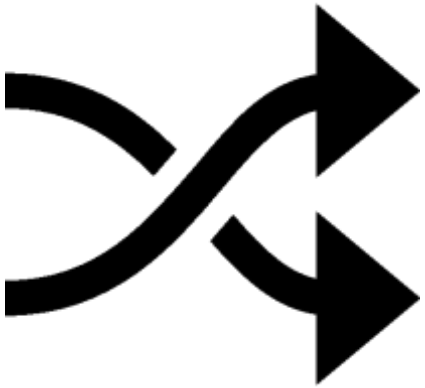
TI CC2640 Architecture



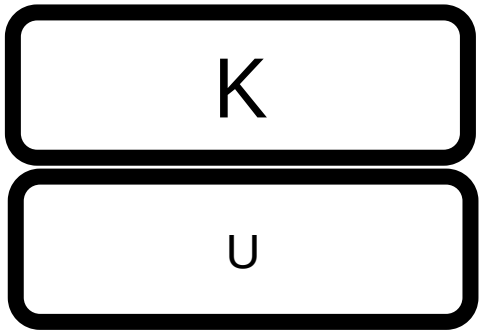
CC2640 (lack of) Security



NO DEP
(NX-BIT)



NO ASLR



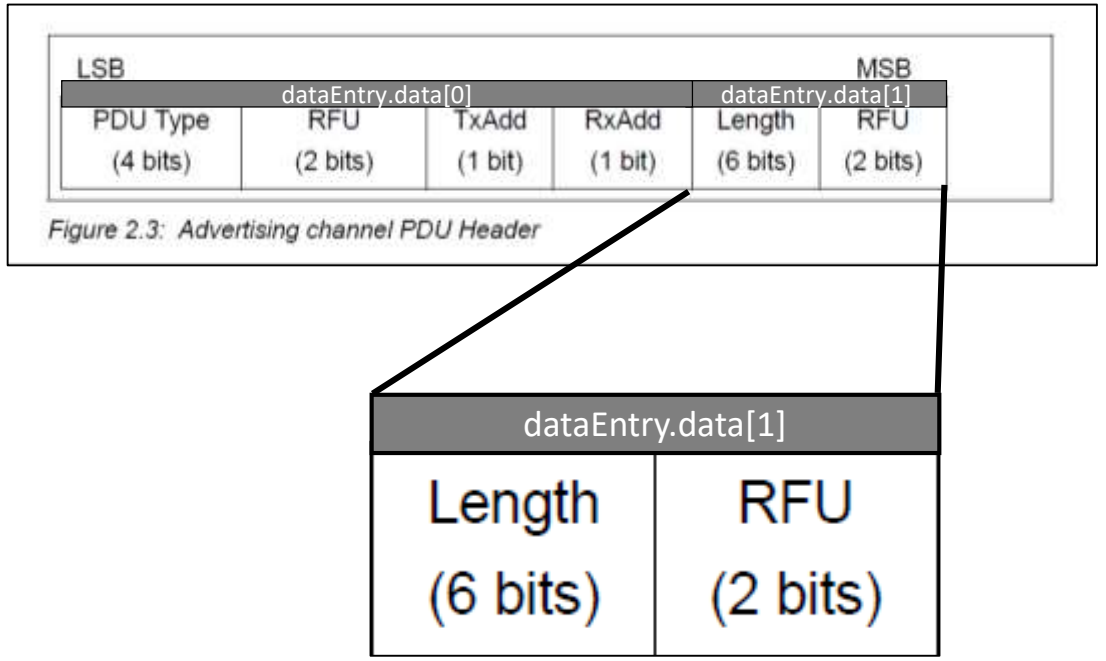
NO MEMORY
MANAGMENT

CC2640 Memory Corruption


```
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;
```

Main core

CC2640 Memory Corruption




```
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;
```

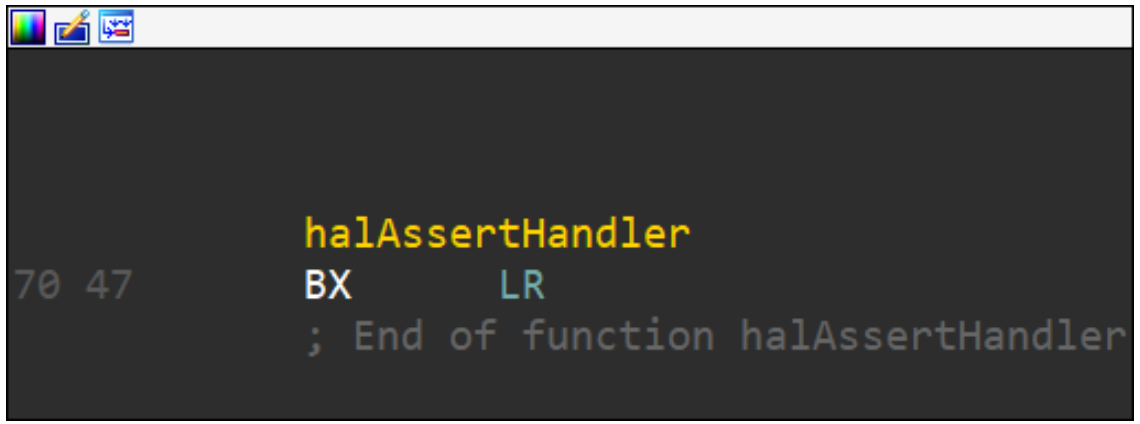
Main core

CC2640 Memory Corruption


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

Main core

CC2640 Memory Corruption



```
halAssertHandler
70 47       BX      LR
            ; End of function halAssertHandler
```

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

Main core

CC2640 Memory Corruption


```
// 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++;
}
...
}
```

Main core

Lets try and crash it

RFU	Length	Actual payload size	Crash?
11	111111 (255)	255	

Lets try and crash it

RFU	Length	Actual payload size	Crash?
11	111111 (255)	255	
00	000001 (1)	1	

Lets try and crash it

RFU	Length	Actual payload size	Crash?
11	111111 (255)	255	
00	000001 (1)	1	
00	111111 (63)	63	

Lets try and crash it

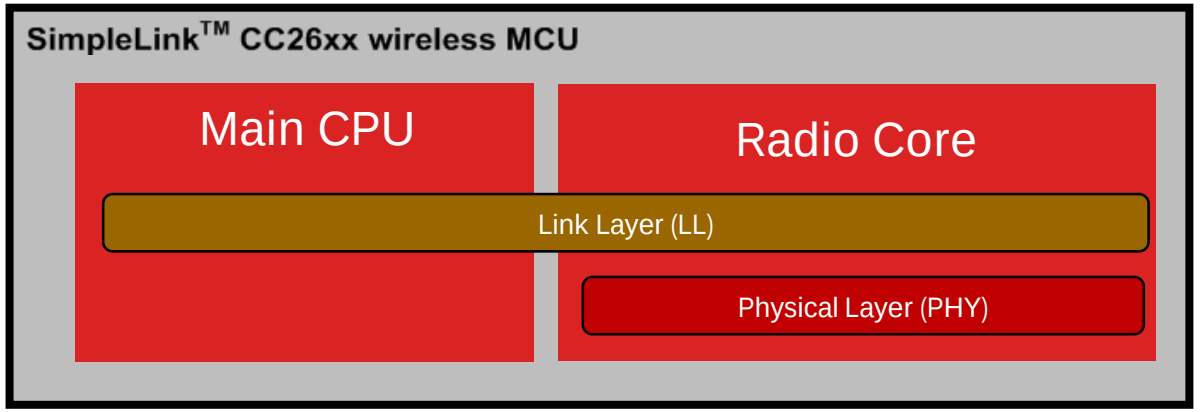
RFU	Length	Actual payload size	Crash?
11	111111 (255)	255	
00	000001 (1)	1	
00	111111 (63)	63	
00	100101 (37)	37	

Lets try and crash it

RFU	Length	Actual payload size	Crash?
11	111111 (255)	255	
00	000001 (1)	1	
00	111111 (63)	63	
00	100101 (37)	37	
10	100101 (165)	**37**	

CC2640 - RTFM

*“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.”*



Packet Length: Main Core vs Radio Core

```
void llGetAdvChanPDU(...)
{
 dataEntry_t *dataEntry;
 uint8 pktLength;
 uint8 *pktData;
 ...
 dataEntry = RFHAL_GetNextDataEntry(RxQ);
 ...
 pktLength = dataEntry.data[1];
```

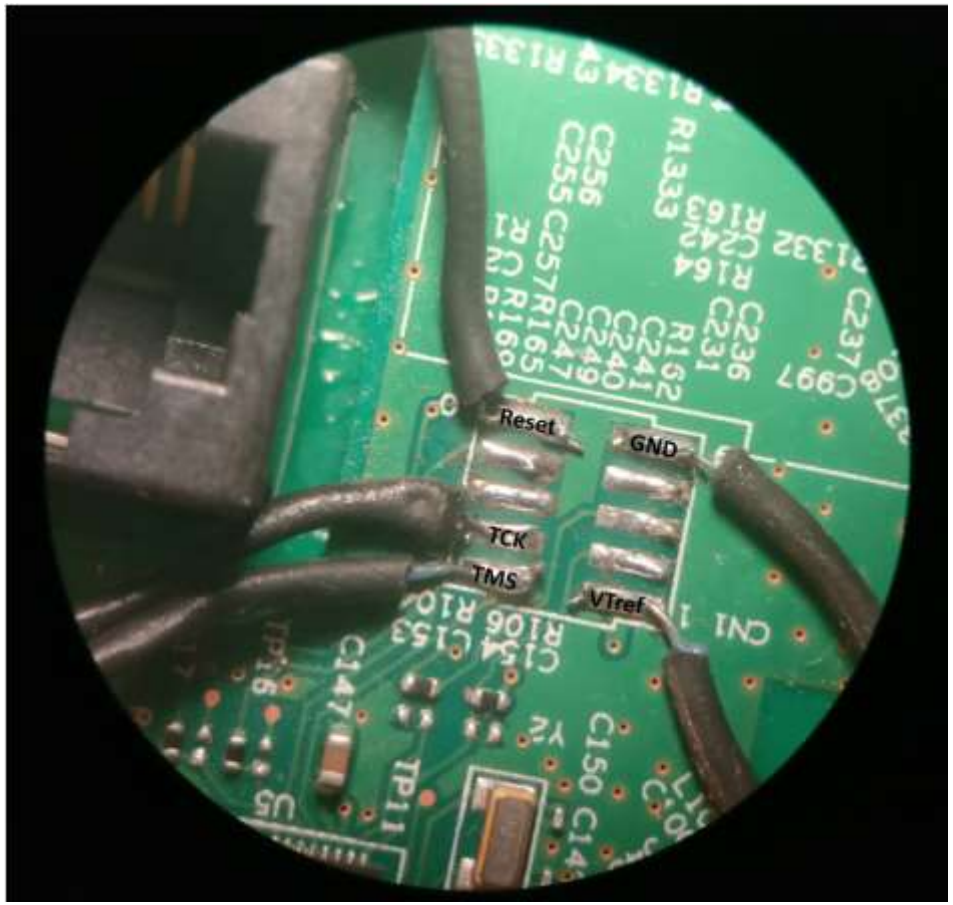
Main core

```
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
}
```

Radio core

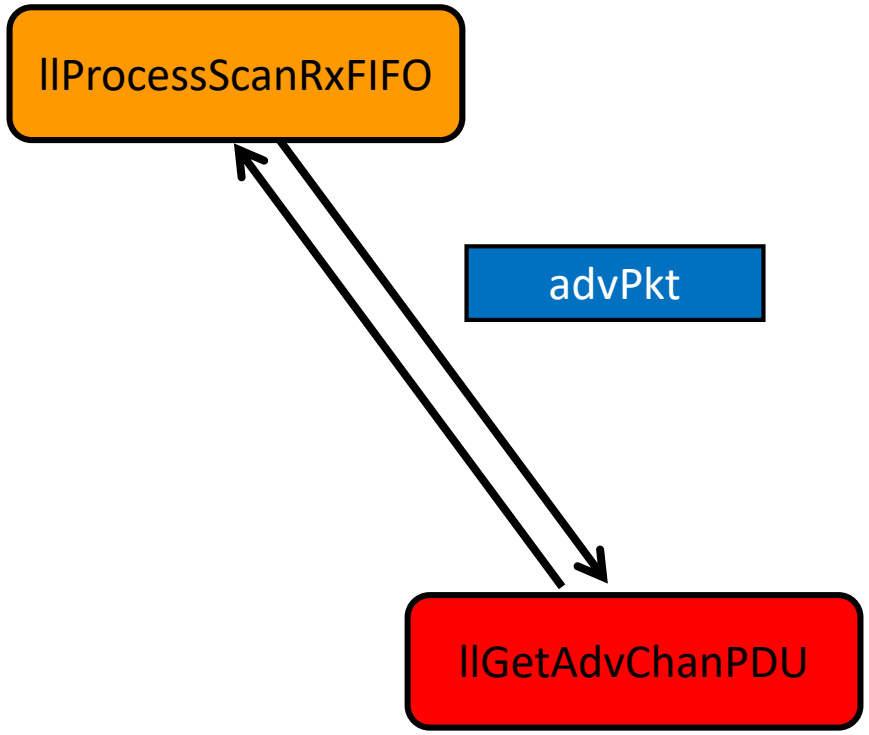
Case Study

CISCO AP1815W



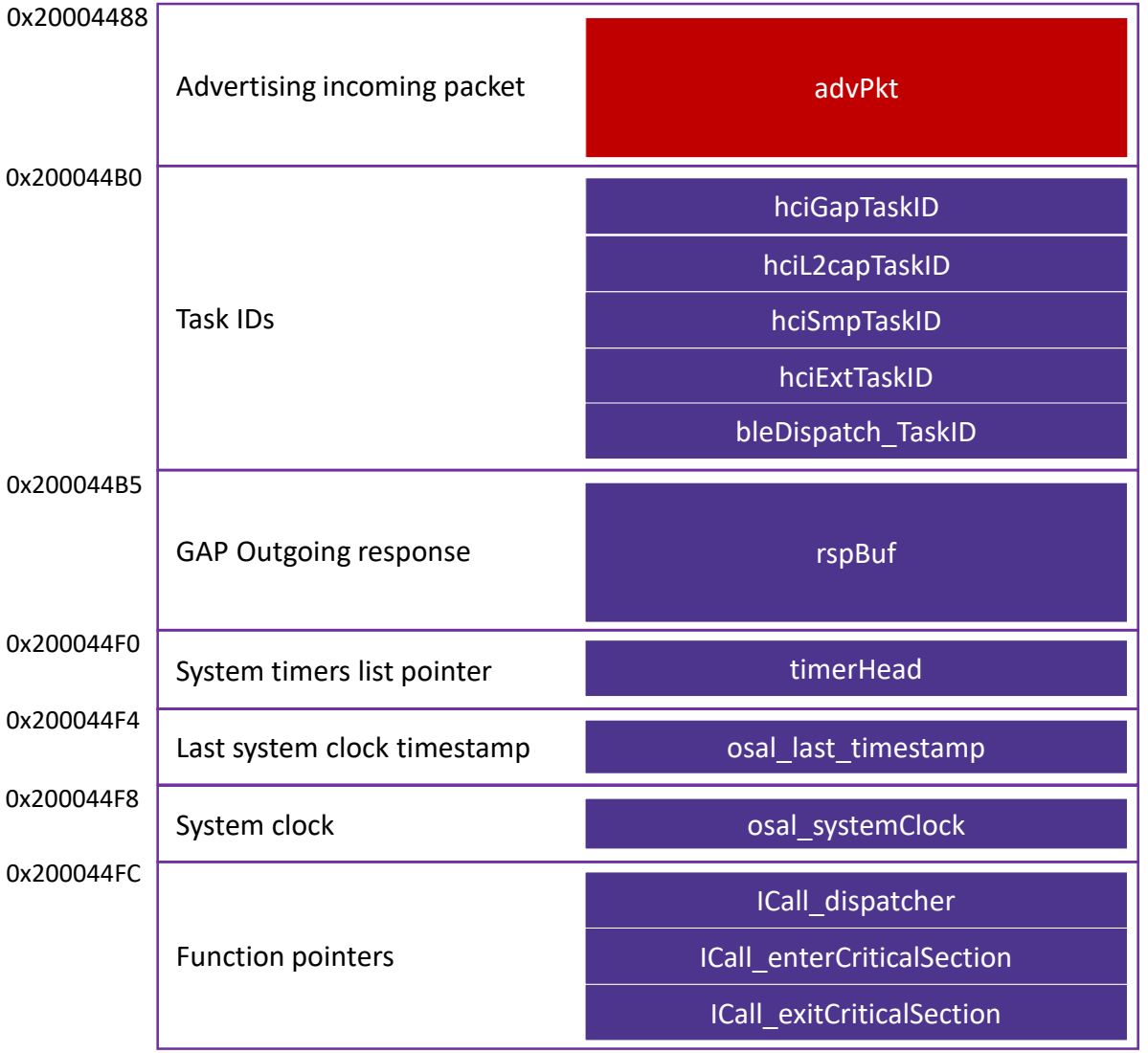
JTAG Header

CC2640 Memory Corruption

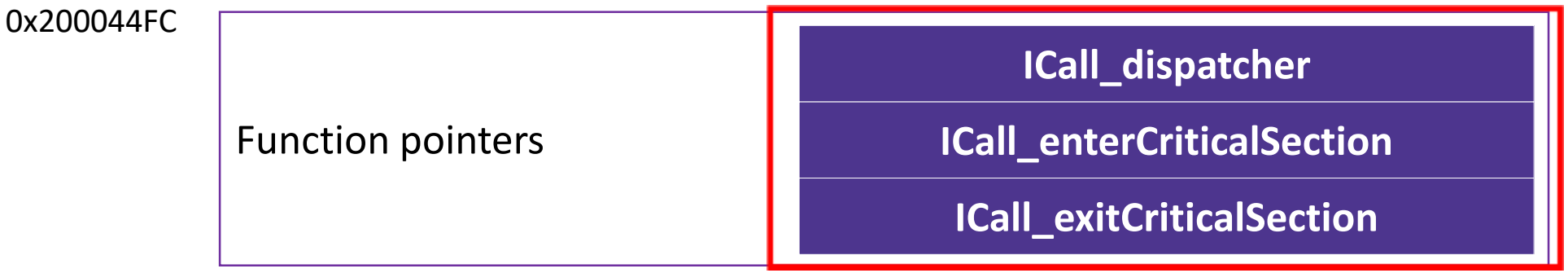



```
void llGetAdvChanPDU(...)
{
 dataEntry_t *dataEntry;
 uint8 pktLength;
 uint8 *pktData;
 ...
 // Copy the rest of the packet
 for (i = 0; i < (unsigned int)*dataLen; ++i)
 {
   *advData++ = *pktData++;
 }
 ...
}
```

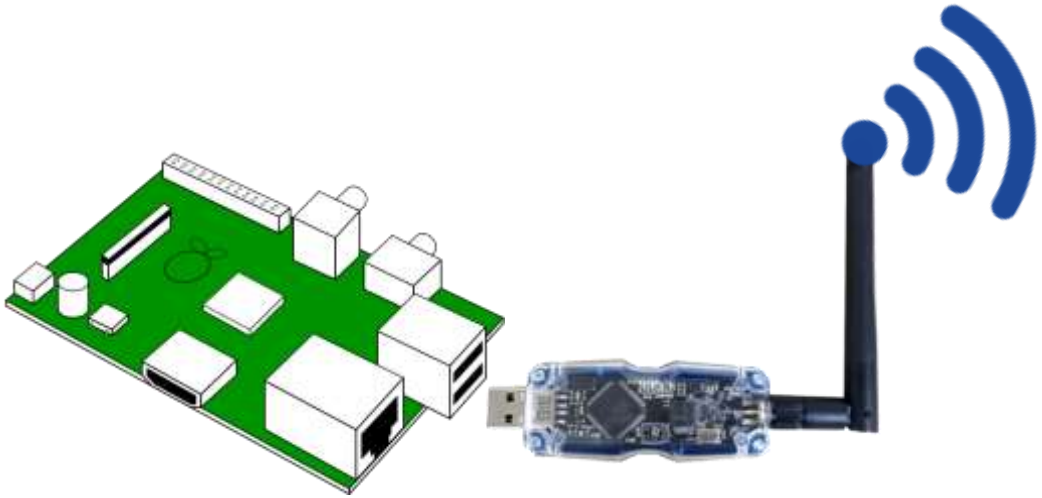
What is being overwritten?



What is being overwritten?



Attacking an Cisco AP

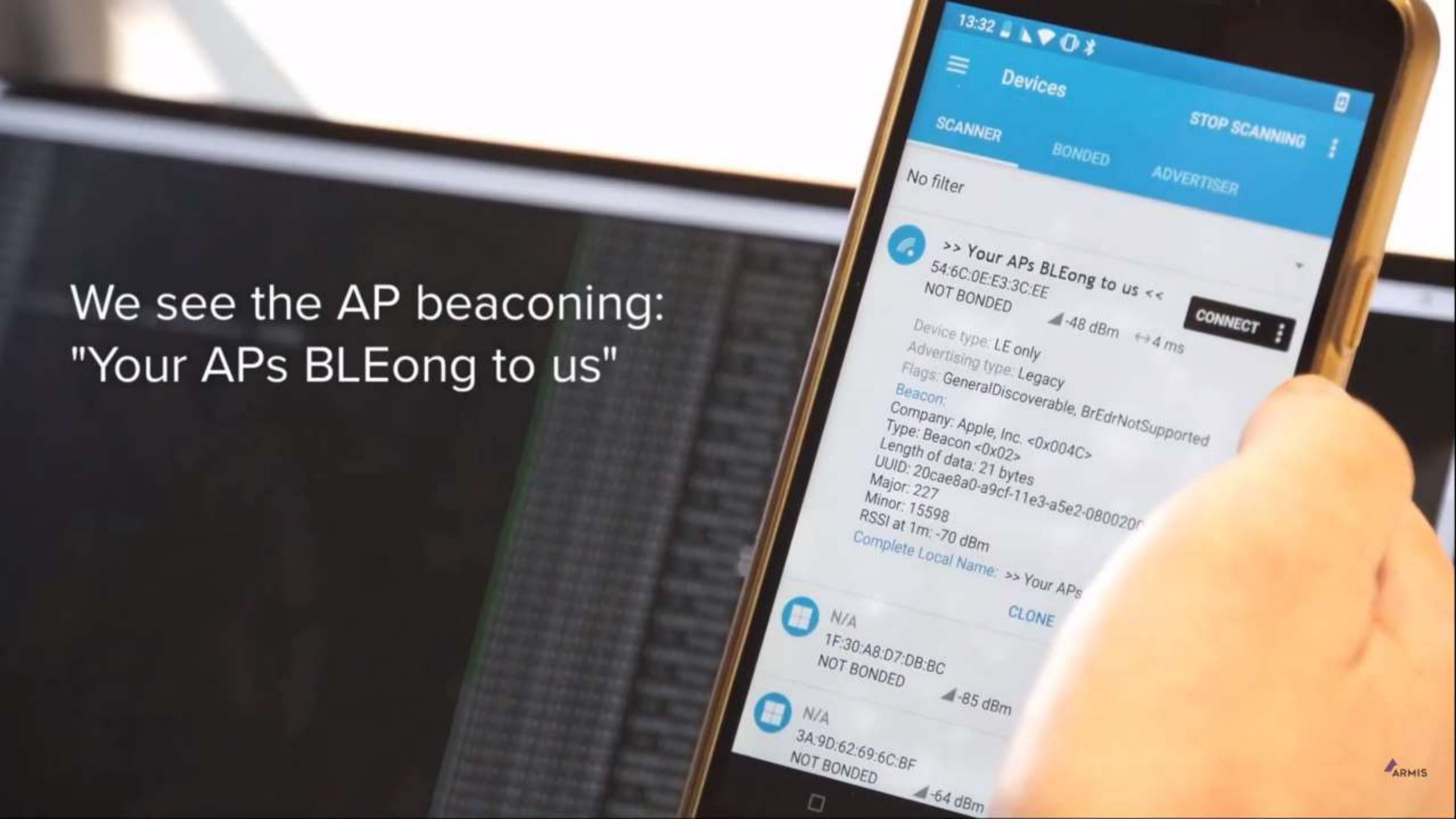


RPi

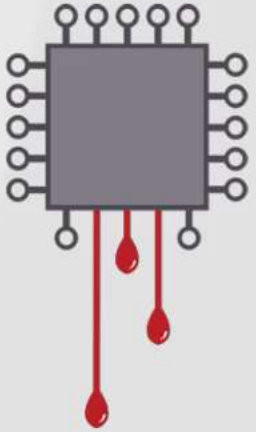
Ubertooth

AP Cisco 1815W


DELL
ARMIS


We see the AP beaconing:
"Your APs BLEong to us"
13:32
Devices
STOP SCANNING
SCANNER
BONDED
ADVERTISER
No filter
>> Your APs BLEong to us <<
54:6C:0E:E3:3C:EE
NOT BONDED
-48 dBm
4 ms
CONNECT
Device type: LE only
Advertising type: Legacy
Flags: GeneralDiscoverable, BrEdrNotSupported
Beacon:
Company: Apple, Inc. <0x004C>
Type: Beacon <0x02>
Length of data: 21 bytes
UUID: 20cae8a0-a9cf-11e3-a5e2-080020
Major: 227
Minor: 15598
RSSI at 1m: -70 dBm
Complete Local Name: >> Your APs
CLONE
N/A
1F:30:A8:D7:DB:BC
NOT BONDED
-85 dBm
N/A
3A:9D:62:69:6C:BF
NOT BONDED
-64 dBm
ARMIS

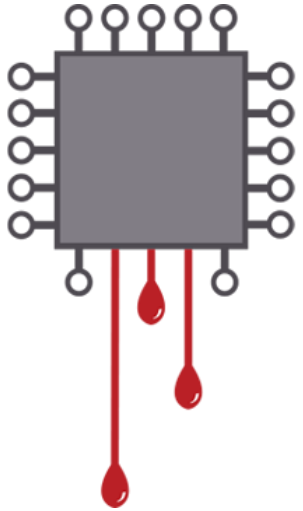
BLEEDING BIT



Demonstration

Take Over of BLE Chip in Cisco Access Point

armis

BLEEDING  BIT

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

ARMIS