# Security analysis of Bluetooth Low Energy Over-The-Air firmware updates

Huy Hung LE

Superviors:
Aurélien Francillon
Romain Cayre

Eurecom

June 29, 2023

## Presentation Outline

## Device firmware updates (DFU)

- DFU over-the-air is a process that allow to update firmware running on a device over BLE from a smartphone or a laptop

- Nordic Semiconductor provides a proprietary DFU process which implemented as a BLE Generic Attribute (GATT) Service

- nRF51 and nRF52 from Nordic Semiconductor are widely used in smartwatches, smartlocks

## Device firmware updates (DFU)

- DFU over-the-air is a process that allow to update firmware running on a device over BLE from a smartphone or a laptop

- Nordic Semiconductor provides a proprietary DFU process which implemented as a BLE Generic Attribute (GATT) Service

- nRF51 and nRF52 from Nordic Semiconductor are widely used in smartwatches, smartlocks

- Goals

  - Reverse engineer the DFU process
  - Analysis the security of this process

## Presentation Outline

## Capture packages over the air

- Using Bluetooth HCI snoop log feature of Android phone

## Capture packages over the air

- Using Bluetooth HCI snoop log feature of Android phone

- Using another DK and Wireshark

## Results

Open bootloader

1. Trigger the DFU process
2. Start DFU process
3. Send length of sd, bl, app
4. Init DFU parameter
5. Send init packet
6. Init DFU parameter complete
7. Set PRN value to 10
8. Prepare to send firmware
9. Send the firmware

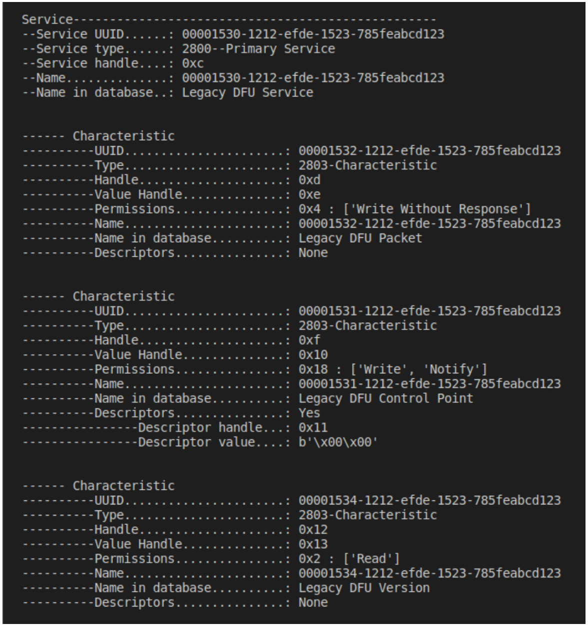
Secure bootloader

1. Trigger the DFU process
2. Select Command Object
3. Set PRN value to 0
4. Create Command Object
5. Send init package
6. Calculate CRC
7. Execute the sent file
8. Set PRN value to 10
9. Send the firmware (More complicated than open bootloader)

## Python program

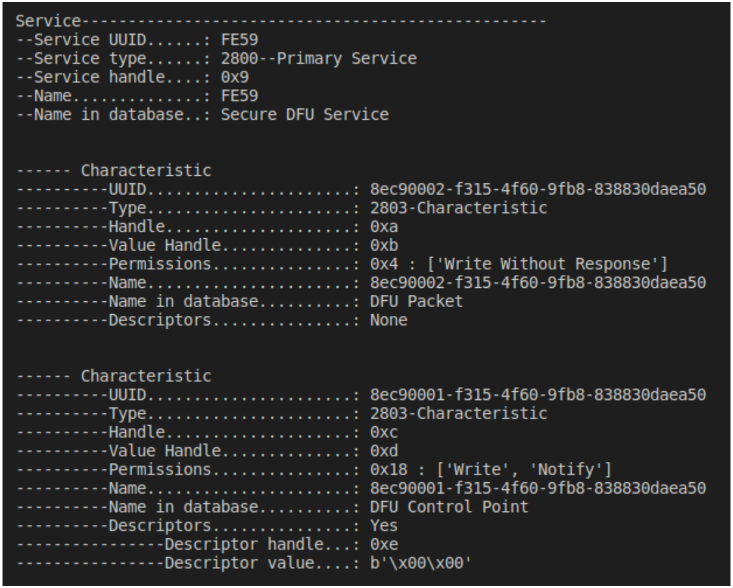
- Using Whad-client framework by Romain Cayre
- Scan, connect, discovery services and characteristics
- Search UUID in a database to find the corresponding name
- With a firmware zip file, automatic unzip, read the data, process with the file then do the DFU process

## Screenshots - Open bootloader



```
Service--------------------------------------------------
--Service UUID......: 00001530-1212-efde-1523-785feabcd123
--Service type......: 2800--Primary Service
--Service handle....: 0xc
--Name..............: 00001530-1212-efde-1523-785feabcd123
--Name in database..: Legacy DFU Service


------ Characteristic
----------UUID.....................: 00001532-1212-efde-1523-785feabcd123
----------Type.....................: 2803-Characteristic
----------Handle...................: 0xd
----------Value Handle.............: 0xe
----------Permissions..............: 0x4 : ['Write Without Response']
----------Name.....................: 00001532-1212-efde-1523-785feabcd123
----------Name in database.........: Legacy DFU Packet
----------Descriptors..............: None


------ Characteristic
----------UUID.....................: 00001531-1212-efde-1523-785feabcd123
----------Type.....................: 2803-Characteristic
----------Handle...................: 0xf
----------Value Handle.............: 0x10
----------Permissions..............: 0x18 : ['Write', 'Notify']
----------Name.....................: 00001531-1212-efde-1523-785feabcd123
----------Name in database.........: Legacy DFU Control Point
----------Descriptors..............: Yes
----------------Descriptor handle...: 0x11
----------------Descriptor value....: b'\x00\x00'


------ Characteristic
----------UUID.....................: 00001534-1212-efde-1523-785feabcd123
----------Type.....................: 2803-Characteristic
----------Handle...................: 0x12
----------Value Handle.............: 0x13
----------Permissions..............: 0x2 : ['Read']
----------Name.....................: 00001534-1212-efde-1523-785feabcd123
----------Name in database.........: Legacy DFU Version
----------Descriptors..............: None
```

## Screenshots - Secure bootloader

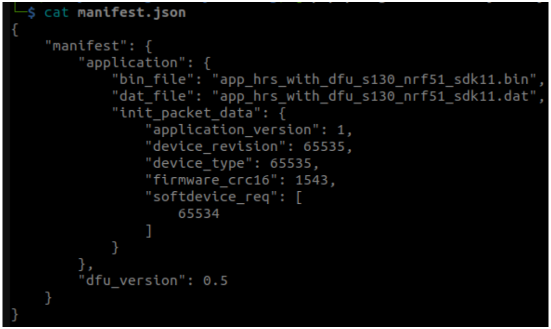


```
Service--------------------------------------------------
--Service UUID......: FE59
--Service type......: 2800--Primary Service
--Service handle....: 0x9
--Name..............: FE59
--Name in database..: Secure DFU Service


------ Characteristic
----------UUID.......................: 8ec90002-f315-4f60-9fb8-838830daea50
----------Type.......................: 2803-Characteristic
----------Handle.....................: 0xa
----------Value Handle...............: 0xb
----------Permissions................: 0x4 : ['Write Without Response']
----------Name.......................: 8ec90002-f315-4f60-9fb8-838830daea50
----------Name in database...........: DFU Packet
----------Descriptors................: None


------ Characteristic
----------UUID.......................: 8ec90001-f315-4f60-9fb8-838830daea50
----------Type.......................: 2803-Characteristic
----------Handle.....................: 0xc
----------Value Handle...............: 0xd
----------Permissions................: 0x18 : ['Write', 'Notify']
----------Name.......................: 8ec90001-f315-4f60-9fb8-838830daea50
----------Name in database...........: DFU Control Point
----------Descriptors................: Yes
----------------Descriptor handle...: 0xe
----------------Descriptor value....: b'\x00\x00'
```

## Presentation Outline

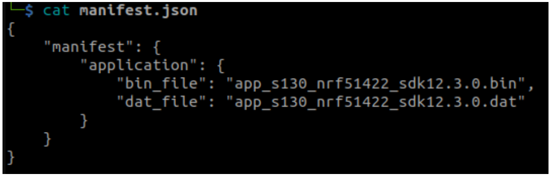
## File manifest.json

Open bootloader:



```
└─$ cat manifest.json
{
    "manifest": {
        "application": {
            "bin_file": "app_hrs_with_dfu_s130_nrf51_sdk11.bin",
            "dat_file": "app_hrs_with_dfu_s130_nrf51_sdk11.dat",
            "init_packet_data": {
                "application_version": 1,
                "device_revision": 65535,
                "device_type": 65535,
                "firmware_crc16": 1543,
                "softdevice_req": [
                    65534
                ]
            }
        },
        "dfu_version": 0.5
    }
}
```

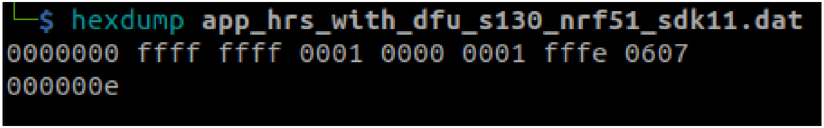
Secure bootloader:



```
└─$ cat manifest.json
{
    "manifest": {
        "application": {
            "bin_file": "app_s130_nrf51422_sdk12.3.0.bin",
            "dat_file": "app_s130_nrf51422_sdk12.3.0.dat"
        }
    }
}
```

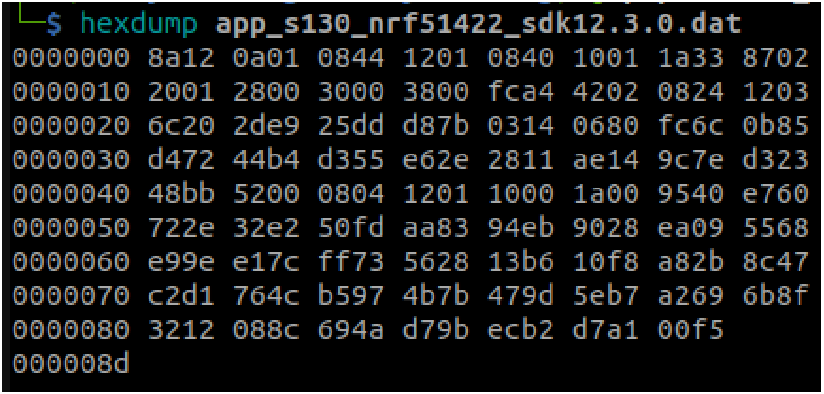
## File .dat

Open bootloader:



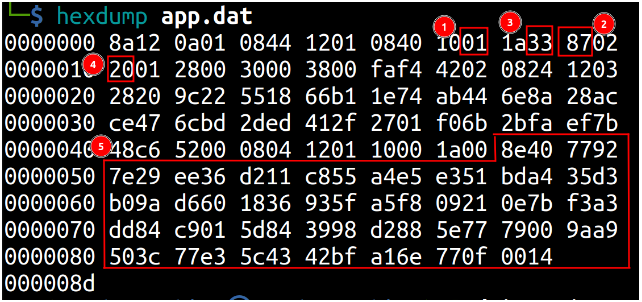
```
└─$ hexdump app_hrs_with_dfu_s130_nrf51_sdk11.dat
0000000 ffff ffff 0001 0000 0001 fffe 0607
000000e
```

Secure bootloader:



```
└─$ hexdump app_s130_nrf51422_sdk12.3.0.dat
0000000 8a12 0a01 0844 1201 0840 1001 1a33 8702
0000010 2001 2800 3000 3800 fca4 4202 0824 1203
0000020 6c20 2de9 25dd d87b 0314 0680 fc6c 0b85
0000030 d472 44b4 d355 e62e 2811 ae14 9c7e d323
0000040 48bb 5200 0804 1201 1000 1a00 9540 e760
0000050 722e 32e2 50fd aa83 94eb 9028 ea09 5568
0000060 e99e e17c ff73 5628 13b6 10f8 a82b 8c47
0000070 c2d1 764c b597 4b7b 479d 5eb7 a269 6b8f
0000080 3212 088c 694a d79b ecb2 d7a1 00f5
000008d
```

## File .dat - secure bootloader



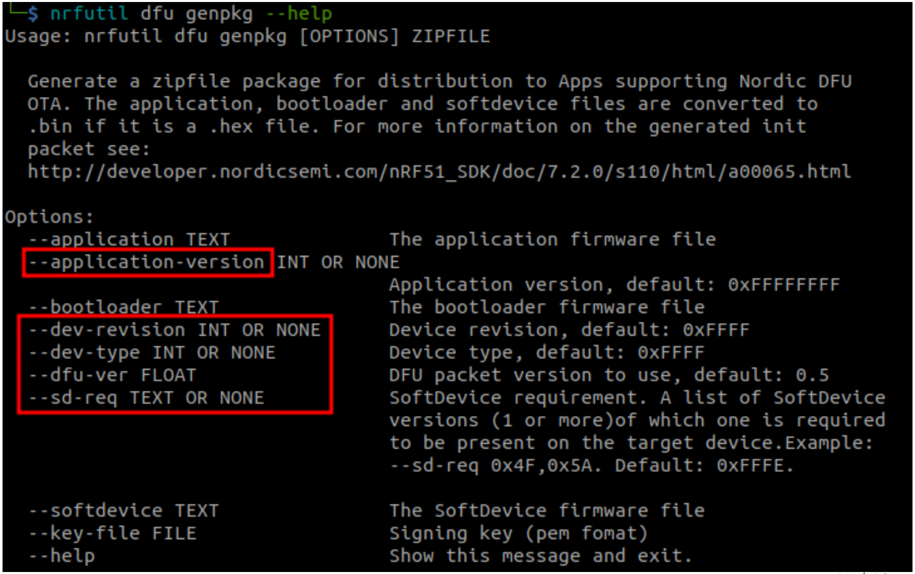
```
└─$ hexdump app.dat
0000000 8a12 0a01 0844 1201 0840 1001 1a33 8702
0000010 2001 2800 3000 3800 faf4 4202 0824 1203
0000020 2820 9c22 5518 66b1 1e74 ab44 6e8a 28ac
0000030 ce47 6cbd 2ded 412f 2701 f06b 2bfa ef7b
0000040 48c6 5200 0804 1201 1000 1a00 8e40 7792
0000050 7e29 ee36 d211 c855 a4e5 e351 bda4 35d3
0000060 b09a d660 1836 935f a5f8 0921 0e7b f3a3
0000070 dd84 c901 5d84 3998 d288 5e77 7900 9aa9
0000080 503c 77e3 5c43 42bf a16e 770f 0014
000008d
```

1. application-version
2. sd-req

3+4. may related to hardware

5. signature of the hash

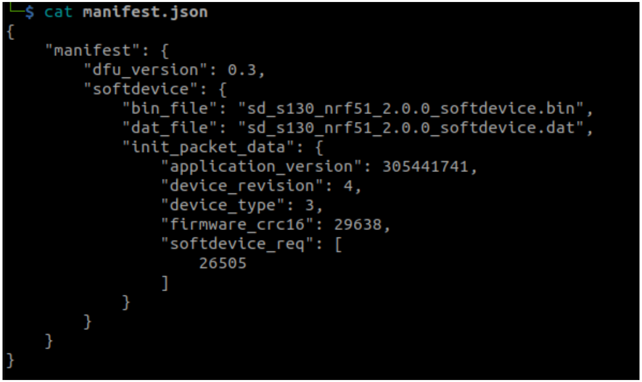
## File .dat - open bootloader



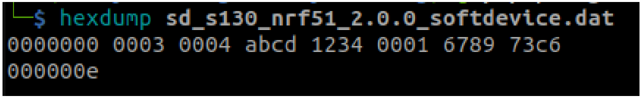
```
└─$ nrfutil dfu genpkg --help
Usage: nrfutil dfu genpkg [OPTIONS] ZIPFILE

  Generate a zipfile package for distribution to Apps supporting Nordic DFU
  OTA. The application, bootloader and softdevice files are converted to
  .bin if it is a .hex file. For more information on the generated init
  packet see:
  http://developer.nordicsemi.com/nRF51_SDK/doc/7.2.0/s110/html/a00065.html

Options:
  --application TEXT              The application firmware file
  --application-version INT OR NONE
                                  Application version, default: 0xFFFFFFFF
  --bootloader TEXT               The bootloader firmware file
  --dev-revision INT OR NONE      Device revision, default: 0xFFFF
  --dev-type INT OR NONE          Device type, default: 0xFFFF
  --dfu-ver FLOAT                 DFU packet version to use, default: 0.5
  --sd-req TEXT OR NONE           SoftDevice requirement. A list of SoftDevice
                                  versions (1 or more)of which one is required
                                  to be present on the target device.Example:
                                  --sd-req 0x4F,0x5A. Default: 0xFFFE.

  --softdevice TEXT               The SoftDevice firmware file
  --key-file FILE                 Signing key (pem fomat)
  --help                          Show this message and exit.
```

## File .dat

dev-type = 3; dev-revision = 4;
application-version = 305441741; sd-req = 0x6789;
0x1234abcd = 305441741; 0x73c6 = 29638; 0x6789 = 26505



```
└─$ cat manifest.json
{
    "manifest": {
        "dfu_version": 0.3,
        "softdevice": {
            "bin_file": "sd_s130_nrf51_2.0.0_softdevice.bin",
            "dat_file": "sd_s130_nrf51_2.0.0_softdevice.dat",
            "init_packet_data": {
                "application_version": 305441741,
                "device_revision": 4,
                "device_type": 3,
                "firmware_crc16": 29638,
                "softdevice_req": [
                    26505
                ]
            }
        }
    }
}
```



```
└─$ hexdump sd_s130_nrf51_2.0.0_softdevice.dat
0000000 0003 0004 abcd 1234 0001 6789 73c6
000000e
```

## Presentation Outline

## Validation of open bootloader

- The information is in available in SDKs

- The implementation in SDK 11.0.0 includes checks for Device type and revision, Supported SoftDevices, and the checksum, but not for the Application version

- We can disable the check by not specific it when create firmware packages (hence, it will use default values such as 0xFFFF for device type and revision, 0xFFFE for softdevice)

## Validation of open bootloader

- The information is in available in SDKs

- The implementation in SDK 11.0.0 includes checks for Device type and revision, Supported SoftDevices, and the checksum, but not for the Application version

- We can disable the check by not specific it when create firmware packages (hence, it will use default values such as 0xFFFF for device type and revision, 0xFFFE for softdevice)

$\longrightarrow$ No signature
CRC is optional
Validation can be disable by modifying the corresponding value to the default one (we know where is the value)

## Validation of open bootloader

- The information is in available in SDKs

- The implementation in SDK 11.0.0 includes checks for Device type and revision, Supported SoftDevices, and the checksum, but not for the Application version

- We can disable the check by not specific it when create firmware packages (hence, it will use default values such as 0xFFFF for device type and revision, 0xFFFE for softdevice)

$\longrightarrow$ No signature
CRC is optional
Validation can be disable by modifying the corresponding value to the default one (we know where is the value)

$\longrightarrow$ If an attacker has access to a legitimate firmware, he can craft a bad firmware and pass the firmware validation process

## Validation of secure bootloader

- The information for validation is available in SDKs

- In brieft, the validation process consists checking some values: Hardware version, SoftDevice version, Firmware version, the hash of the packet, Signature of the packet, Available space (or Firmware size)
The order of checking may different between SDK versions

- The acceptance rules for versions
  - Hardware version and Softdevice Firmware ID: need to match
  - Firmware version: $>$ or $\geq$

## Validation of secure bootloader

- The information for validation is available in SDKs

- In brieft, the validation process consists checking some values: Hardware version, SoftDevice version, Firmware version, the hash of the packet, Signature of the packet, Available space (or Firmware size)
The order of checking may different between SDK versions

- The acceptance rules for versions

- Hardware version and Softdevice Firmware ID: need to match
- Firmware version: $>$ or $\geq$

$\longrightarrow$ If an attacker has access to a good firmware that can do DFU, he/she will know the Hardware and Softdevice of the target
$+$ The tool nrfutil can read the information from a firmware package
$+$ The attacker can also use some tools that can send the init package and wait for responding. The order of checking is different between SDKs, they may be able to distinguish the difference and identify the SDKs version

## Presentation Outline

## Summary

## Summary

- Just by discovering, the attack may identify the type of bootloader: Secure bootloader or open bootloader

## Summary

- Just by discovering, the attack may identify the type of bootloader: Secure bootloader or open bootloader

- If the bootloader is open bootloader, the device is vulnerable to Firmware Modification Attack

## Summary

- Just by discovering, the attack may identify the type of bootloader: Secure bootloader or open bootloader

- If the bootloader is open bootloader, the device is vulnerable to Firmware Modification Attack

- If the firmware is secure bootloader
+ by using signature and CRC, the device may be safe from Firmware Modification Attack
+ by the rules of version acceptance, the device may be safe from Downgrade Attack
+ however, the device can be fingerprinted by the version acceptance rules
+ the order of checking rules is different between SDKs, the attacker may compare the time and fingerprint the device

# Demo

**Thank you for listening!**