BLE GATT Fuzzing

HardwearIO

Baptiste Boyer

October 24th, 2024

Quarkslab

WHOAMI

Baptiste Boyer 🇫🇷/🇷🇪

Junior R&D Engineer at Quarkslab

Embedded/Wireless topics

Goals

GOALS

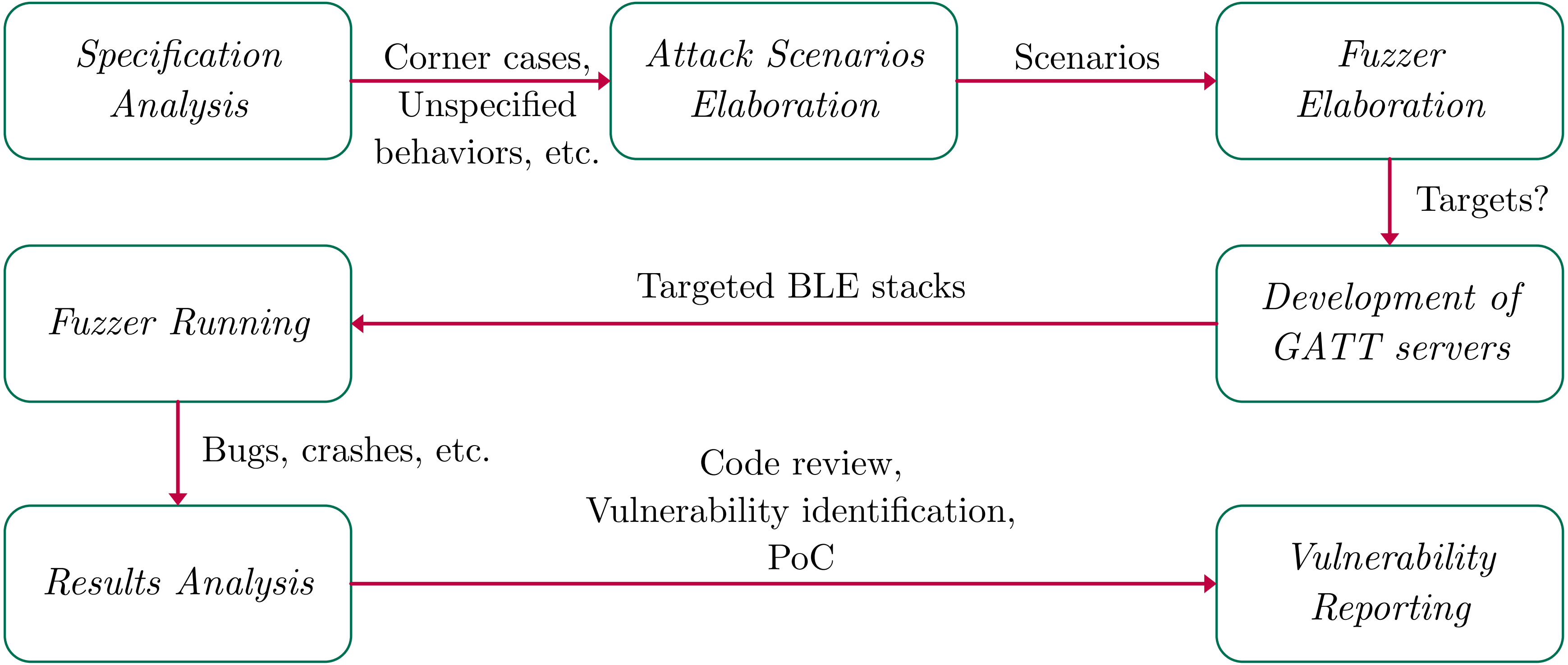
Framework Evaluation & Tool Development

Context: A new framework has been developed, *WHAD* (Wireless HAcking Devices)
Objectives: Evaluate the internal framework
Create a tool based on *WHAD* to assess its robustness, a fuzzer!

Unexplored Security Landscape

Context: A lot of security research has been done on BLE but ATT/GATT layers remain relatively unexplored
Objectives: Conduct an in-depth security assessment of these layers with our tool

METHODOLOGY



Specification Analysis
Corner cases, Unspecified behaviors, etc.
Attack Scenarios Elaboration
Scenarios
Fuzzer Elaboration
Targets?
Development of GATT servers
Targeted BLE stacks
Fuzzer Running
Bugs, crashes, etc.
Results Analysis
Code review, Vulnerability identification, PoC
Vulnerability Reporting

Fig: Adopted methodology

What is BLE?

BLE

BLE: Bluetooth Low Energy

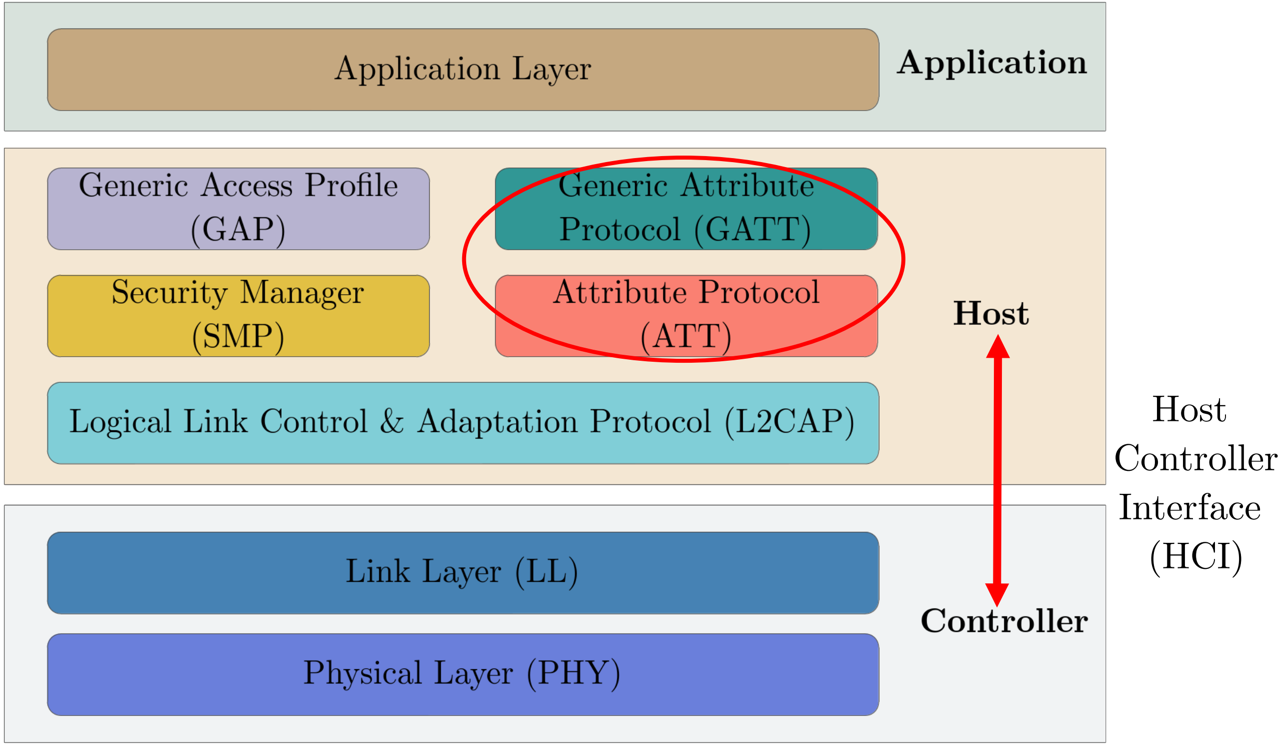
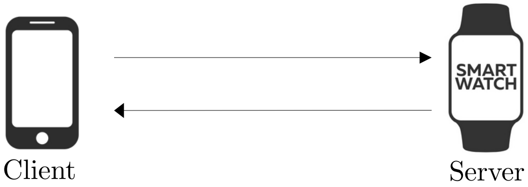

Application Layer
Application
Generic Access Profile (GAP)
Generic Attribute Protocol (GATT)
Security Manager (SMP)
Attribute Protocol (ATT)
Host
Logical Link Control & Adaptation Protocol (L2CAP)
Host Controller Interface (HCI)
Link Layer (LL)
Physical Layer (PHY)
Controller

Fig: BLE Protocol Stack

ATT LAYER

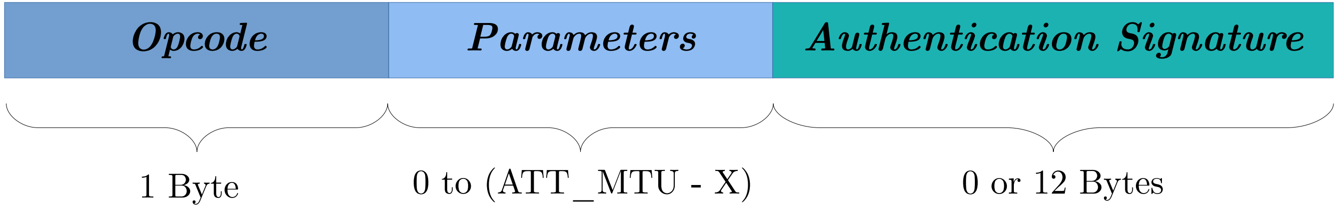


- Client-Server architecture
- Defines how data is represented and the methods by which that data can be read or written
- **Attribute = data structure**

Handle	***UUID***	***Value***	***Permissions***
2 Bytes	2 or 16 Bytes	Variable length	Implementation specific

ATT LAYER

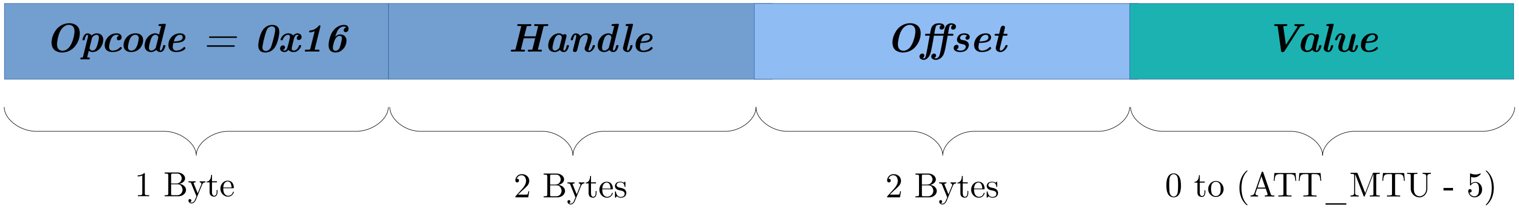
- 30 **ATT** Protocol Data Unit (**PDU**) defined to exchange data
- 6 Types: *Commands*, *Requests*, *Responses*, *Notifications*, *Indications*, *Confirmations*
- **ATT PDU** Format



Opcode	***Parameters***	***Authentication Signature***
1 Byte	0 to (ATT_MTU - X)	0 or 12 Bytes

ATT LAYER

- Long Attribute values i.e. size(ATT_Value) > (ATT_MTU - 1)
- To write entire value: **ATT_PREP_WRITE_REQ** & **ATT_EXECUTE_WRITE_REQ**
- **ATT_PREP_WRITE_REQ** Format



Opcode = 0x16	***Handle***	***Offset***	***Value***
1 Byte	2 Bytes	2 Bytes	0 to (ATT_MTU - 5)

ATT LAYER

- Concrete utilization of **Prepare Write** and **Execute Write Requests**

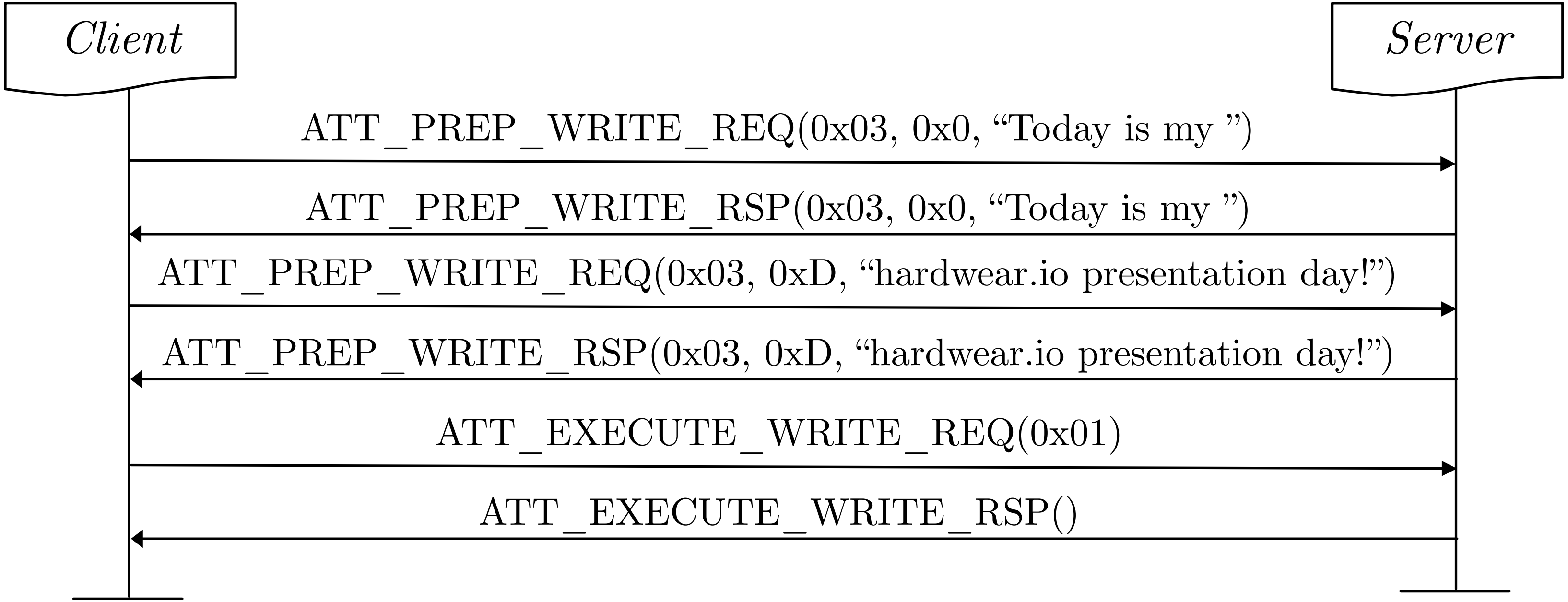

Client
Server
ATT_PREP_WRITE_REQ(0x03, 0x0, “Today is my ”)
ATT_PREP_WRITE_RSP(0x03, 0x0, “Today is my ”)
ATT_PREP_WRITE_REQ(0x03, 0xD, “hardwear.io presentation day!”)
ATT_PREP_WRITE_RSP(0x03, 0xD, “hardwear.io presentation day!”)
ATT_EXECUTE_WRITE_REQ(0x01)
ATT_EXECUTE_WRITE_RSP()

Fig: Write Long Attribute Values example

GATT LAYER

- Defines a framework built upon ATT layer of **procedures** and **formats**

Attributes

Service : collection of data and associated behaviors to accomplish a function
Characteristic: attribute used in a service along with properties and configuration information
Descriptor : contains related information about the Characteristic Value

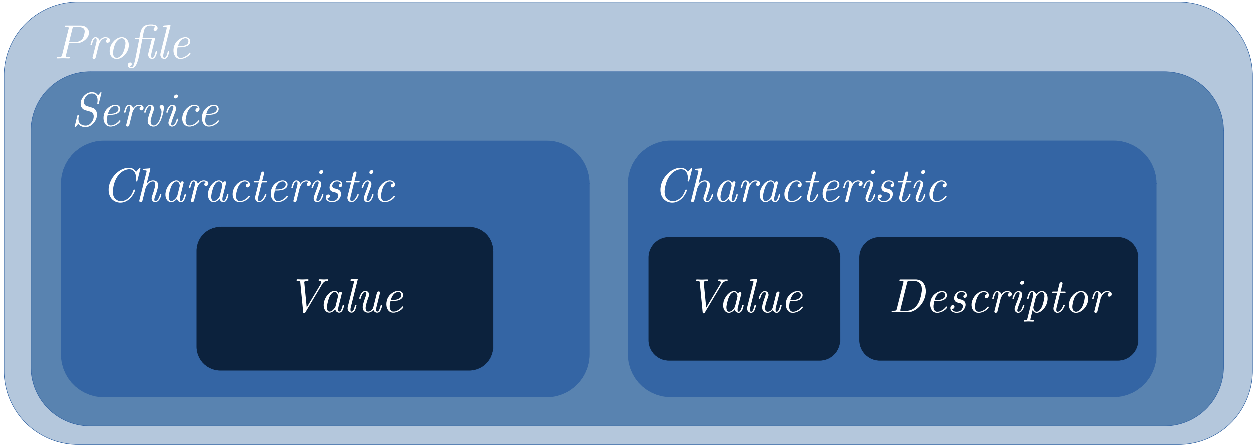
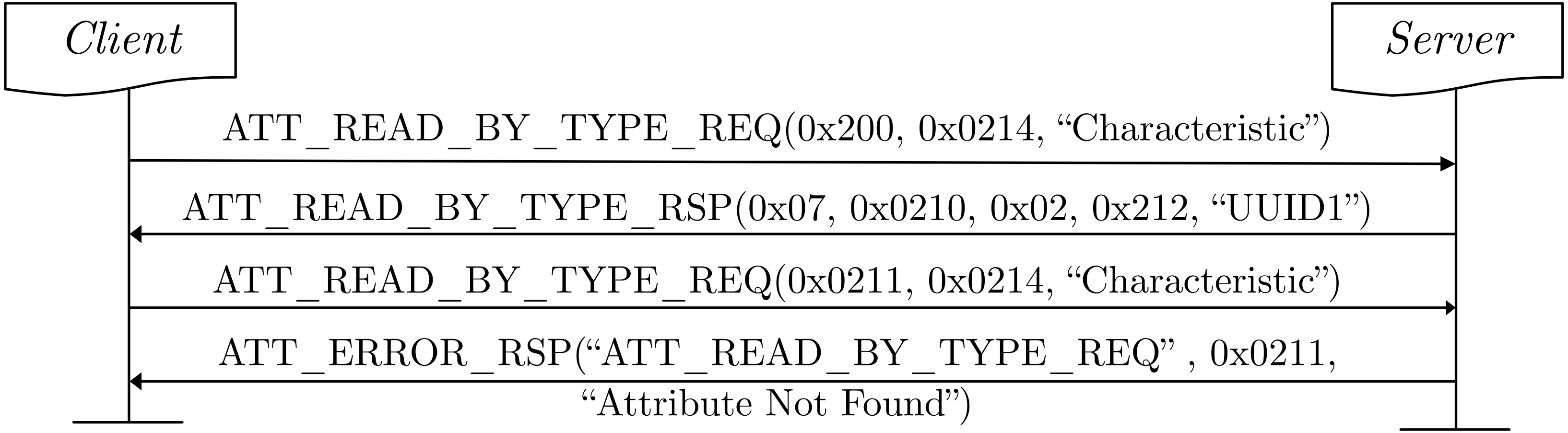


Fig: GATT Profile Hierarchy

GATT LAYER

- 11 **features** and **procedures**

  Server Configuration, Primary Service Discovery, Relation Discovery, Characteristic Discovery, Characteristic Descriptor Discovery, Reading/Writing a Characteristic Value, Reading/Writing a Characteristic Descriptor, Notification/Indication of a Characteristic Value



ATT_READ_BY_TYPE_REQ(0x200, 0x0214, “Characteristic”)

ATT_READ_BY_TYPE_RSP(0x07, 0x0210, 0x02, 0x212, “UUID1”)

ATT_READ_BY_TYPE_REQ(0x0211, 0x0214, “Characteristic”)

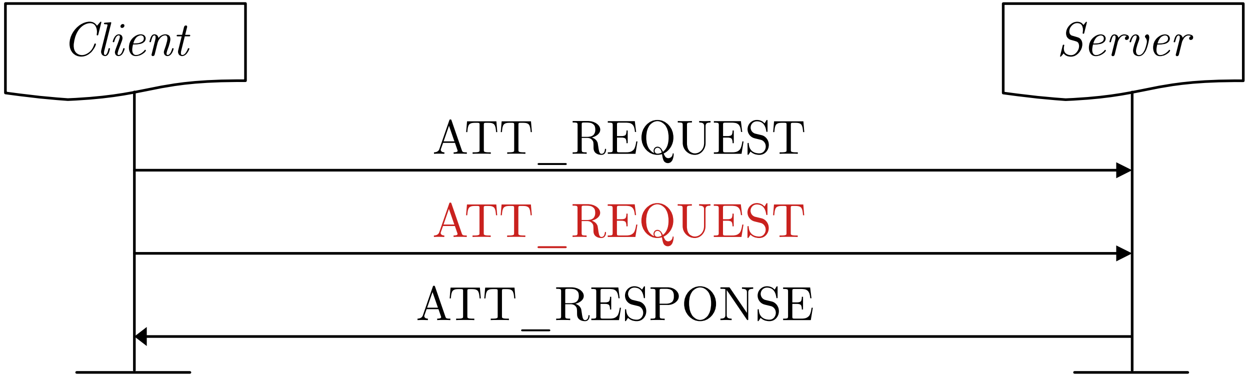
ATT_ERROR_RSP(“ATT_READ_BY_TYPE_REQ” , 0x0211, “Attribute Not Found”)

Fig: Discover All Characteristics of a Service

Attack Scenarios

SCENARIO #1

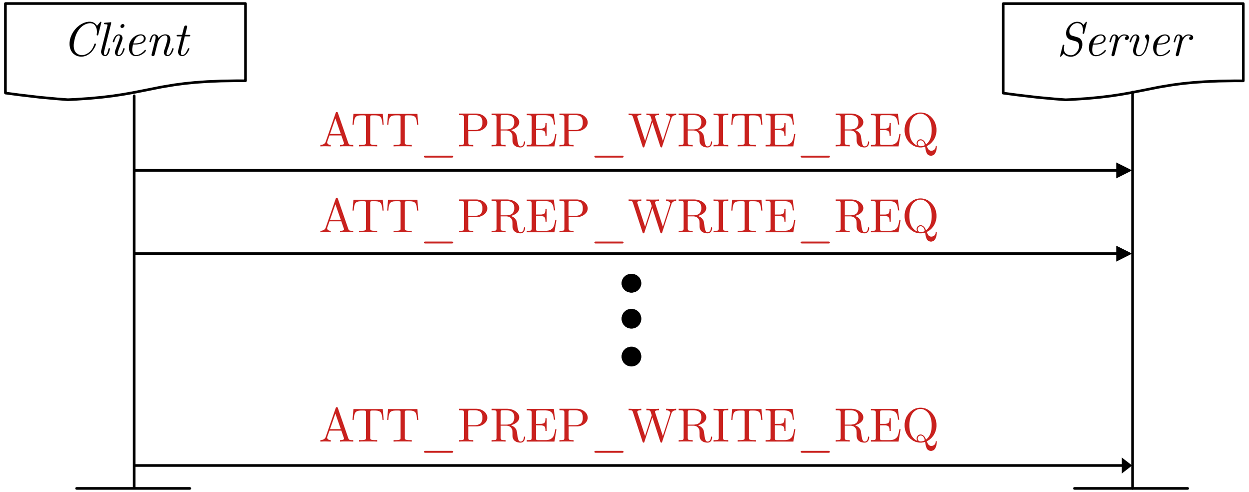
Observation	Scenario
"Once a client sends a request to a server, that client shall send no other request to the same server until a response PDU has been received." (BLE Spec: Vol 3. Part F. 3.3.2)	Send another request before a response PDU has been received



Ref: CVE-2019-19192 from SweynTooth

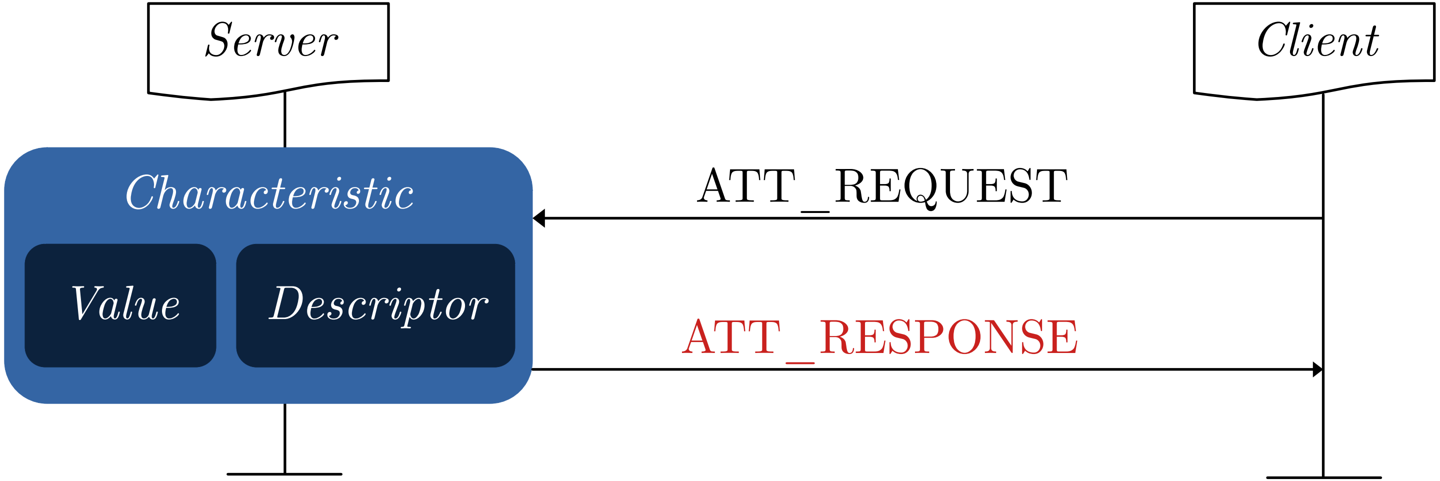
SCENARIO #2

Observation	Scenario
"A server may limit the number of prepared writes that it can queue. A higher layer specification should define this limit." (BLE Spec: Vol 3. Part F. 3.4.6.1)	Send many Prepare Write Request



SCENARIO #3

Observation	Scenario
Inconsistency of GATT server First action done by the client is to discover the Services, the Characteristics and the associated Descriptors	Trick the client by sending wrong responses



Setup & Implementations

Quarkslab

WHAD IS LOVE!

WHAD[1] is an open source framework for exploring, hacking and more generally playing with common wireless protocols.

- Supports wireless protocols such as BLE, Zigbee or LoRaWAN.
- Supports different hardware: HCI device, nRF52, etc.
- A lot of features: sniffing, replaying, hijacking, etc.

WHAD can be used to:

- Craft and send legitimate or custom PDUs.
- Handle received PDUs easily.
- Populate and spawn a GATT server.
- Log packet exchanges in PCAP files.

[1] https://whad.io/

FUZZING & MONITORING

Fuzzer illustration:

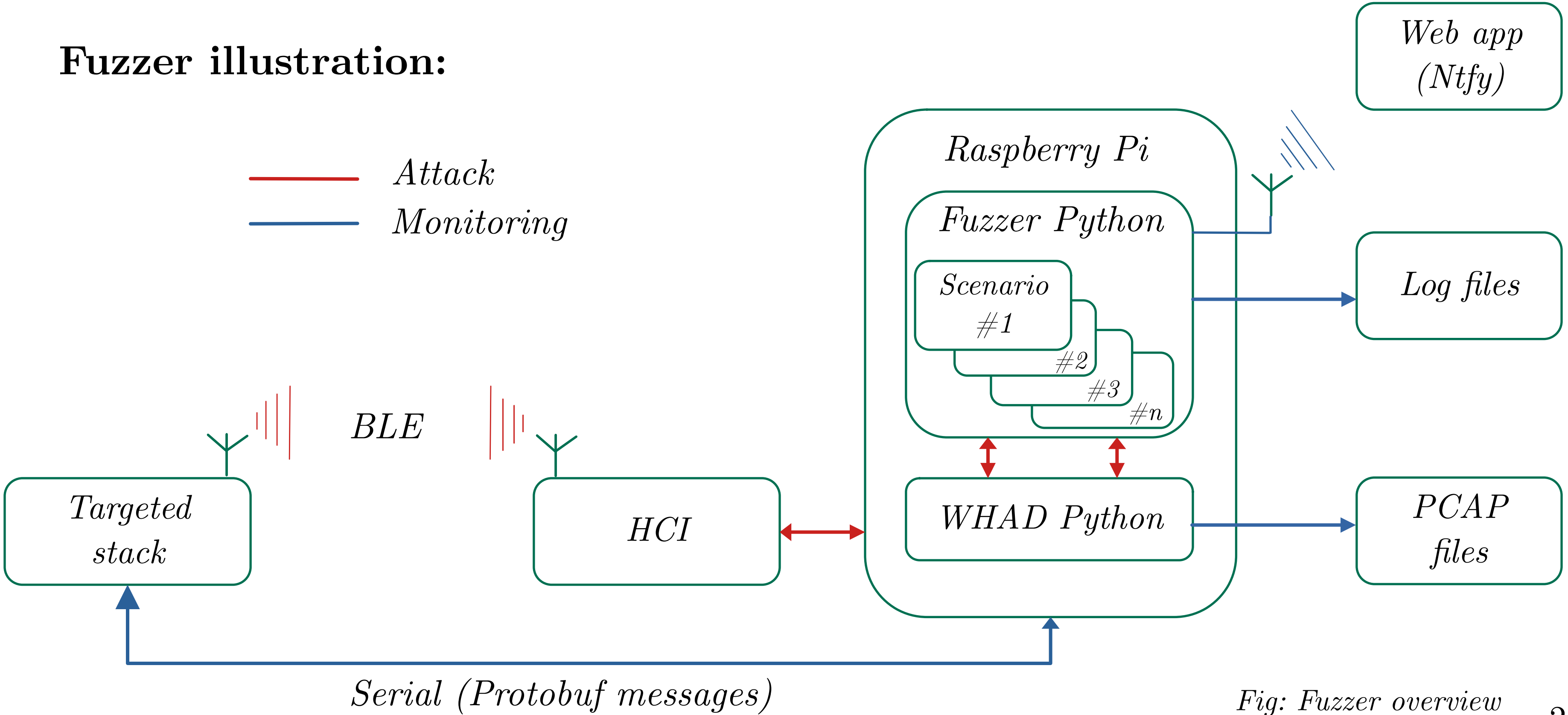


Fig: Fuzzer overview

FUZZING & MONITORING

Strategy:

- No mutation, random strategy
- 1 fuzzing session to test 1 scenario
- Keep the connection during the whole session
- Basic feedback: disconnections, unresponsive stack, crashs

FUZZING & MONITORING

Scenario #1:

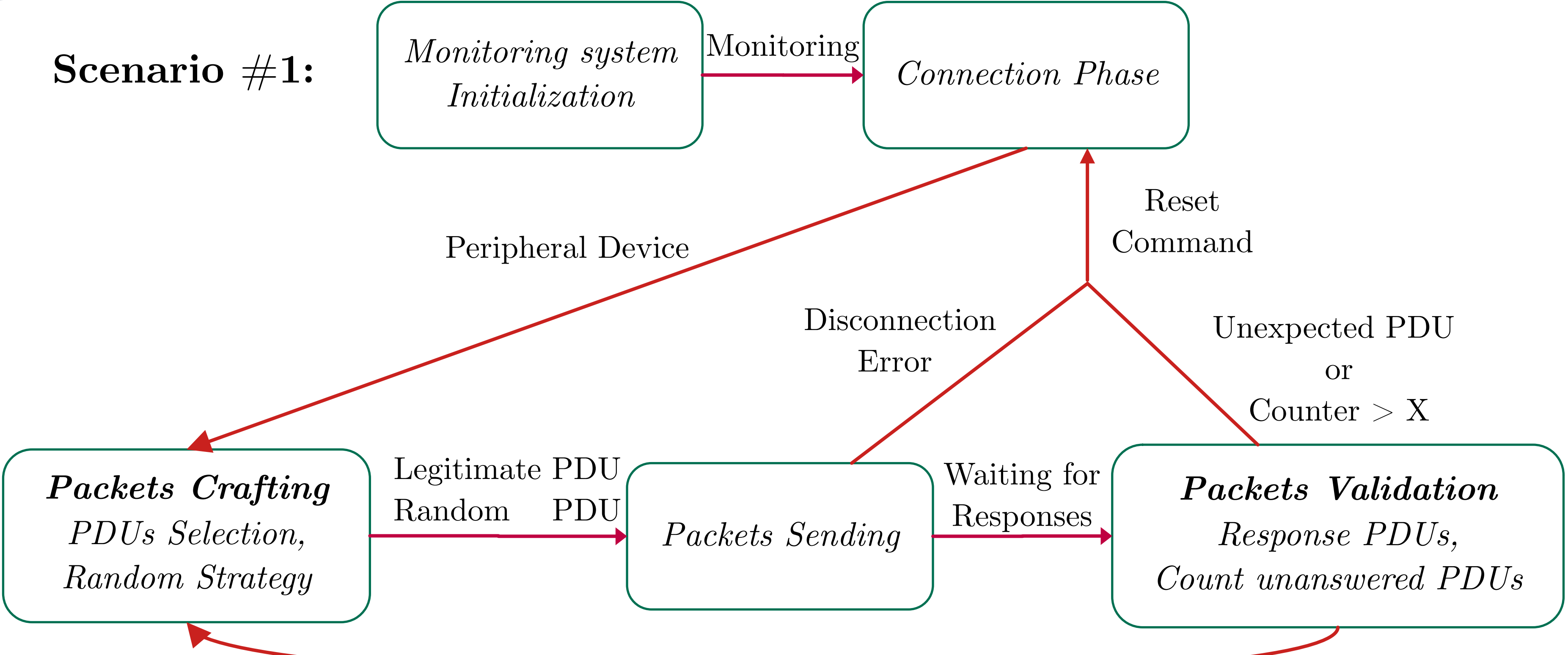
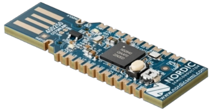
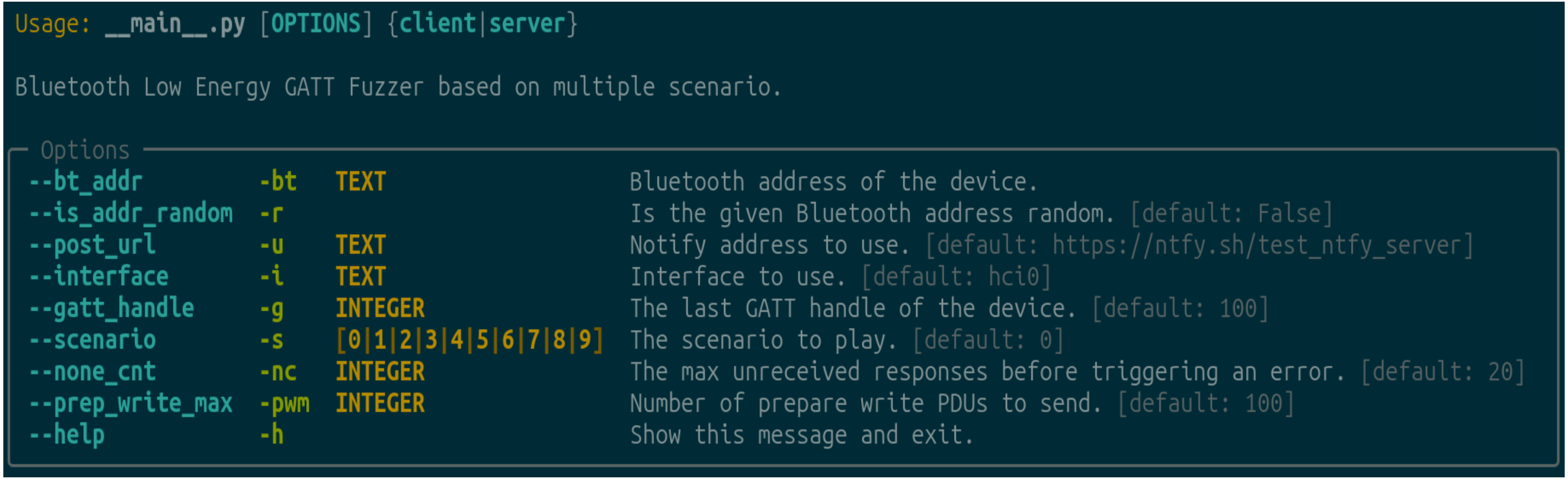


Fig: Scenario #1

TEST BENCH

Tool/Service	*Utilization*
 Raspberry Pi	Runs our fuzzing scenarios and WHAD Connected to targeted stack with serial port
protobuf Protocol Buffers Protocol Buffers	Standardized structures: Connection, Disconnection, Reset command, Crash logs
Ntfy	An effective monitoring system that delivers real-time notifications through a web app
 BLE Sniffer	Verify the packet exchanges for results confirmation

CLI

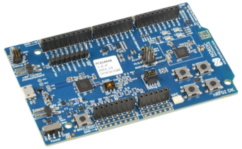
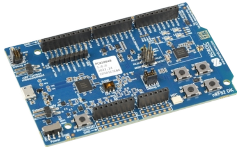


```
Usage: __main__.py [OPTIONS] {client|server}

Bluetooth Low Energy GATT Fuzzer based on multiple scenario.

╭─ Options ──────────────────────────────────────────────────────────────────────────────────────────────────────────╮
│ --bt_addr          -bt   TEXT                   Bluetooth address of the device.                                   │
│ --is_addr_random   -r                           Is the given Bluetooth address random. [default: False]            │
│ --post_url         -u    TEXT                   Notify address to use. [default: https://ntfy.sh/test_ntfy_server] │
│ --interface        -i    TEXT                   Interface to use. [default: hci0]                                  │
│ --gatt_handle      -g    INTEGER                The last GATT handle of the device. [default: 100]                 │
│ --scenario         -s    [0|1|2|3|4|5|6|7|8|9]  The scenario to play. [default: 0]                                 │
│ --none_cnt         -nc   INTEGER                The max unreceived responses before triggering an error. [default: 20] │
│ --prep_write_max   -pwm  INTEGER                Number of prepare write PDUs to send. [default: 100]               │
│ --help             -h                           Show this message and exit.                                        │
╰────────────────────────────────────────────────────────────────────────────────────────────────────────────────────╯
```

Fig: CLI help display

https://github.com/quarkslab/ble-gatt-fuzzing

BLE STACKS & HARDWARE

BLE Stack	Zephyr RTOS BLE Stack	MyNewt RTOS NimBLE Stack	Bluedroid/ Fluoride stack from Android
Compatible Hardware	 nRF52	 ESP32, nRF52	📱 Android, ESP32

⚠ **GATT servers based on provided examples by each SDK.**

TEST BENCH

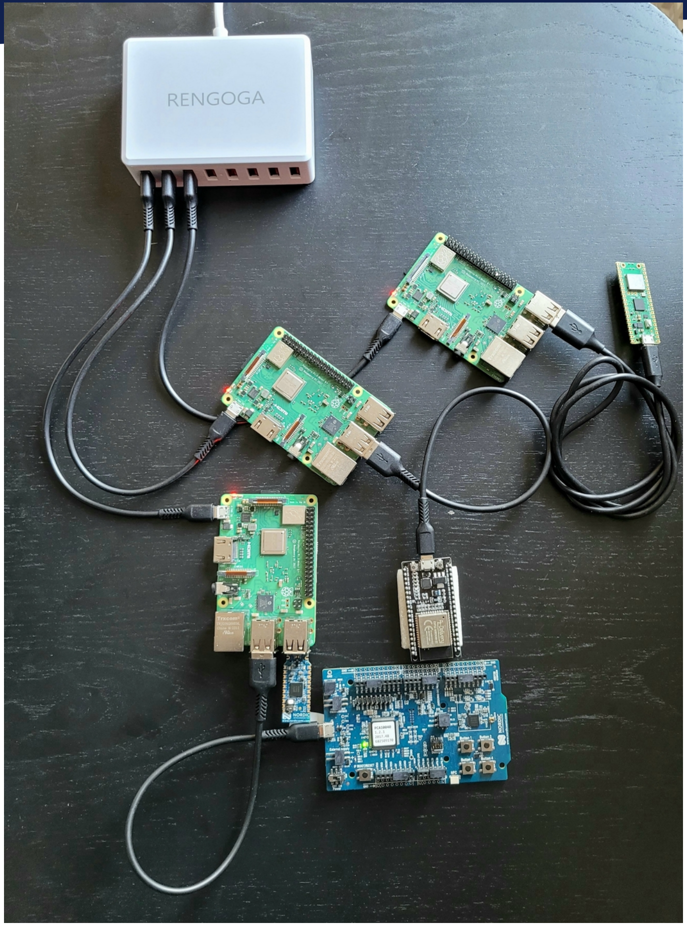
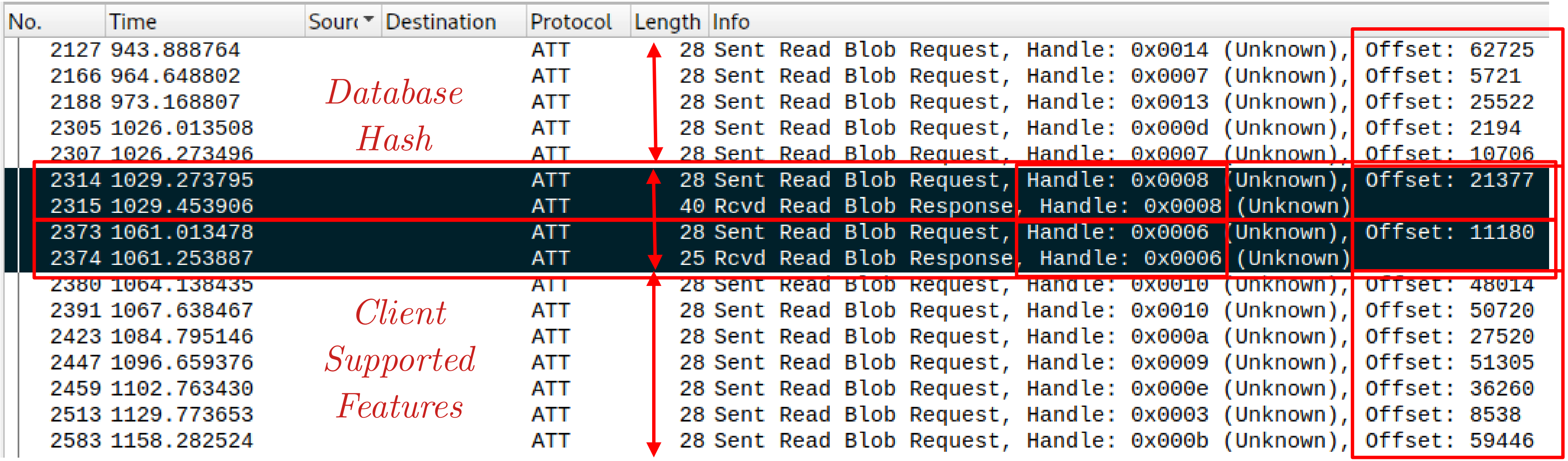


Fig: Test bench

Results

NC: UNEXPECTED READ BLOB RESPONSES

Bluedroid Read Blob Request process:



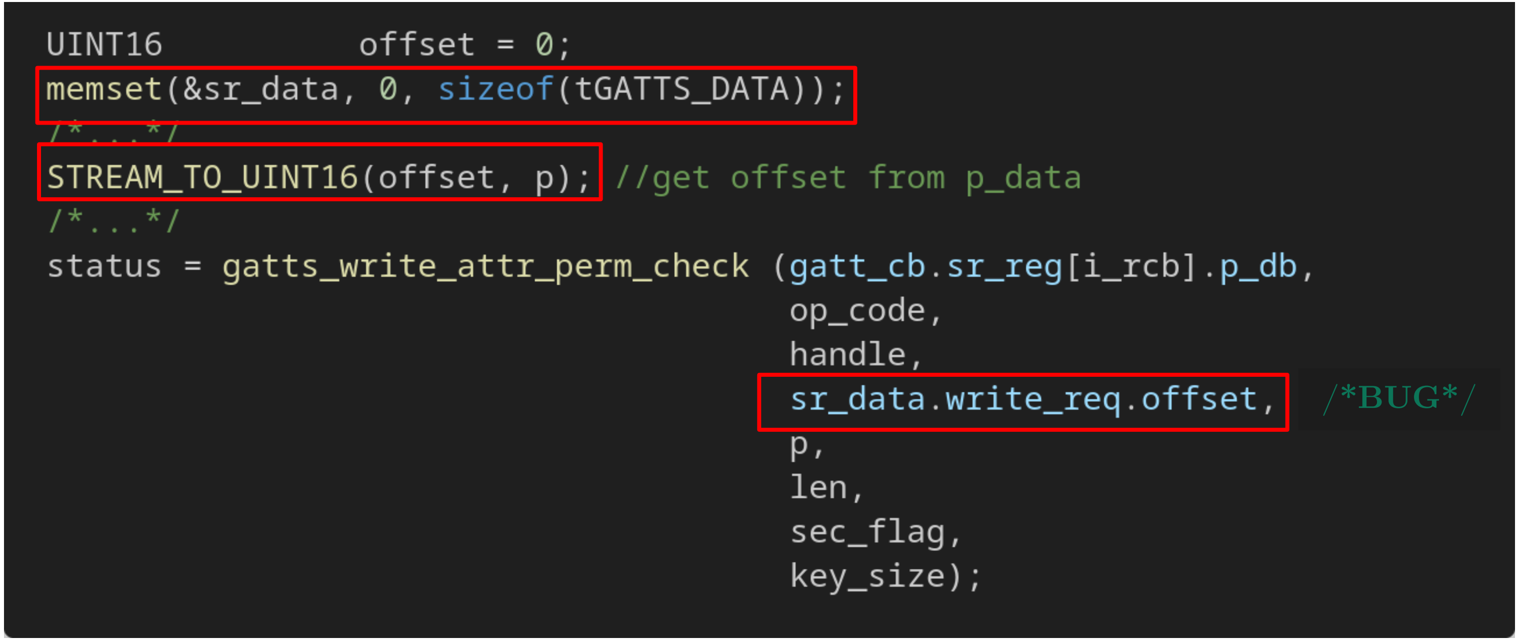
No.	Time	Sour	Destination	Protocol	Length	Info
2127	943.888764			ATT	28	Sent Read Blob Request, Handle: 0x0014 (Unknown), Offset: 62725
2166	964.648802			ATT	28	Sent Read Blob Request, Handle: 0x0007 (Unknown), Offset: 5721
2188	973.168807			ATT	28	Sent Read Blob Request, Handle: 0x0013 (Unknown), Offset: 25522
2305	1026.013508			ATT	28	Sent Read Blob Request, Handle: 0x000d (Unknown), Offset: 2194
2307	1026.273496			ATT	28	Sent Read Blob Request, Handle: 0x0007 (Unknown), Offset: 10706
2314	1029.273795			ATT	28	Sent Read Blob Request, Handle: 0x0008 (Unknown), Offset: 21377
2315	1029.453906			ATT	40	Rcvd Read Blob Response, Handle: 0x0008 (Unknown)
2373	1061.013478			ATT	28	Sent Read Blob Request, Handle: 0x0006 (Unknown), Offset: 11180
2374	1061.253887			ATT	25	Rcvd Read Blob Response, Handle: 0x0006 (Unknown)
2380	1064.138435			ATT	28	Sent Read Blob Request, Handle: 0x0010 (Unknown), Offset: 48014
2391	1067.638467			ATT	28	Sent Read Blob Request, Handle: 0x0010 (Unknown), Offset: 50720
2423	1084.795146			ATT	28	Sent Read Blob Request, Handle: 0x000a (Unknown), Offset: 27520
2447	1096.659376			ATT	28	Sent Read Blob Request, Handle: 0x0009 (Unknown), Offset: 51305
2459	1102.763430			ATT	28	Sent Read Blob Request, Handle: 0x000e (Unknown), Offset: 36260
2513	1129.773653			ATT	28	Sent Read Blob Request, Handle: 0x0003 (Unknown), Offset: 8538
2583	1158.282524			ATT	28	Sent Read Blob Request, Handle: 0x000b (Unknown), Offset: 59446

BLE

Bluedroid v4.0 vs v5.1 Standardized Attributes

BUG: BYPASS THE WRITE PERMISSION CHECK

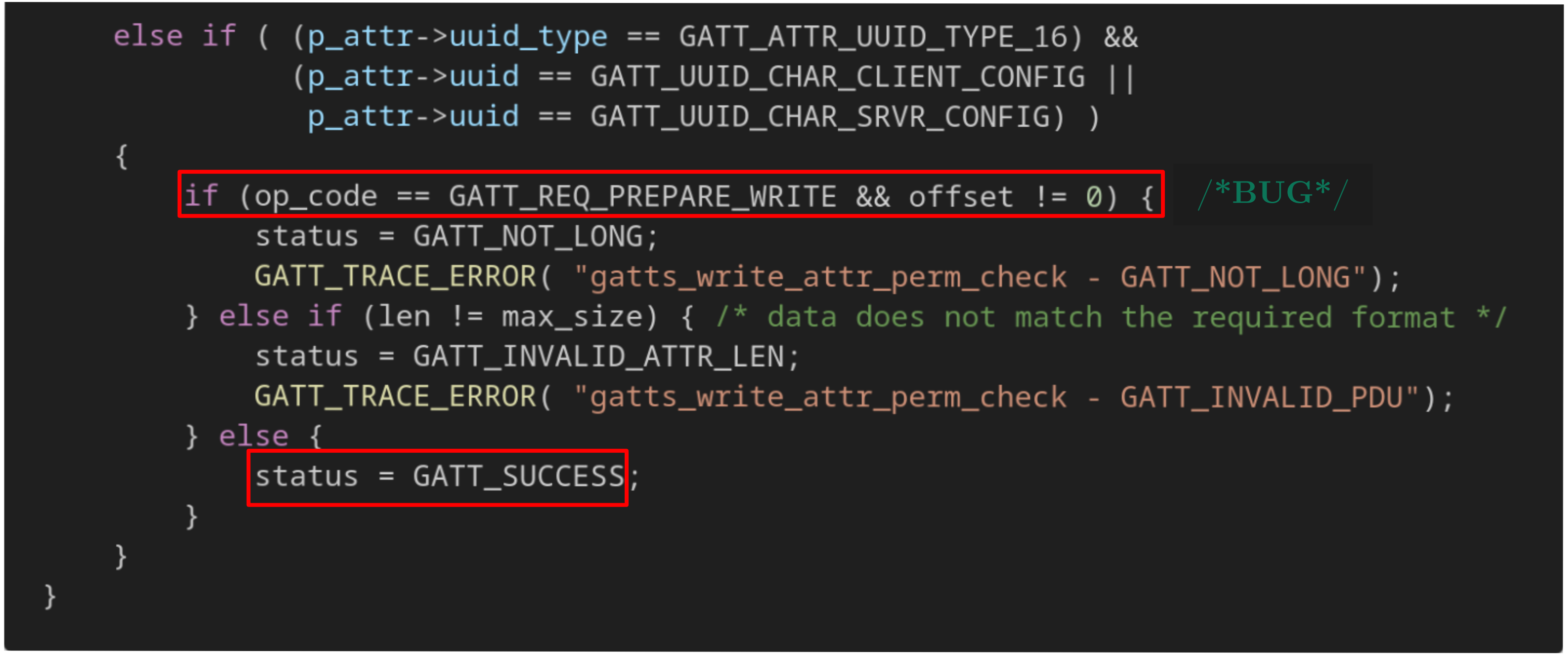
Bluedroid Prepare Write Request process:



```
UINT16          offset = 0;
memset(&sr_data, 0, sizeof(tGATTS_DATA));
/*...*/
STREAM_TO_UINT16(offset, p); //get offset from p_data
/*...*/
status = gatts_write_attr_perm_check (gatt_cb.sr_reg[i_rcb].p_db,
                                      op_code,
                                      handle,
                                      sr_data.write_req.offset,   /*BUG*/
                                      p,
                                      len,
                                      sec_flag,
                                      key_size);
```

BUG: BYPASS THE WRITE PERMISSION CHECK

Bluedroid Prepare Write Request process:

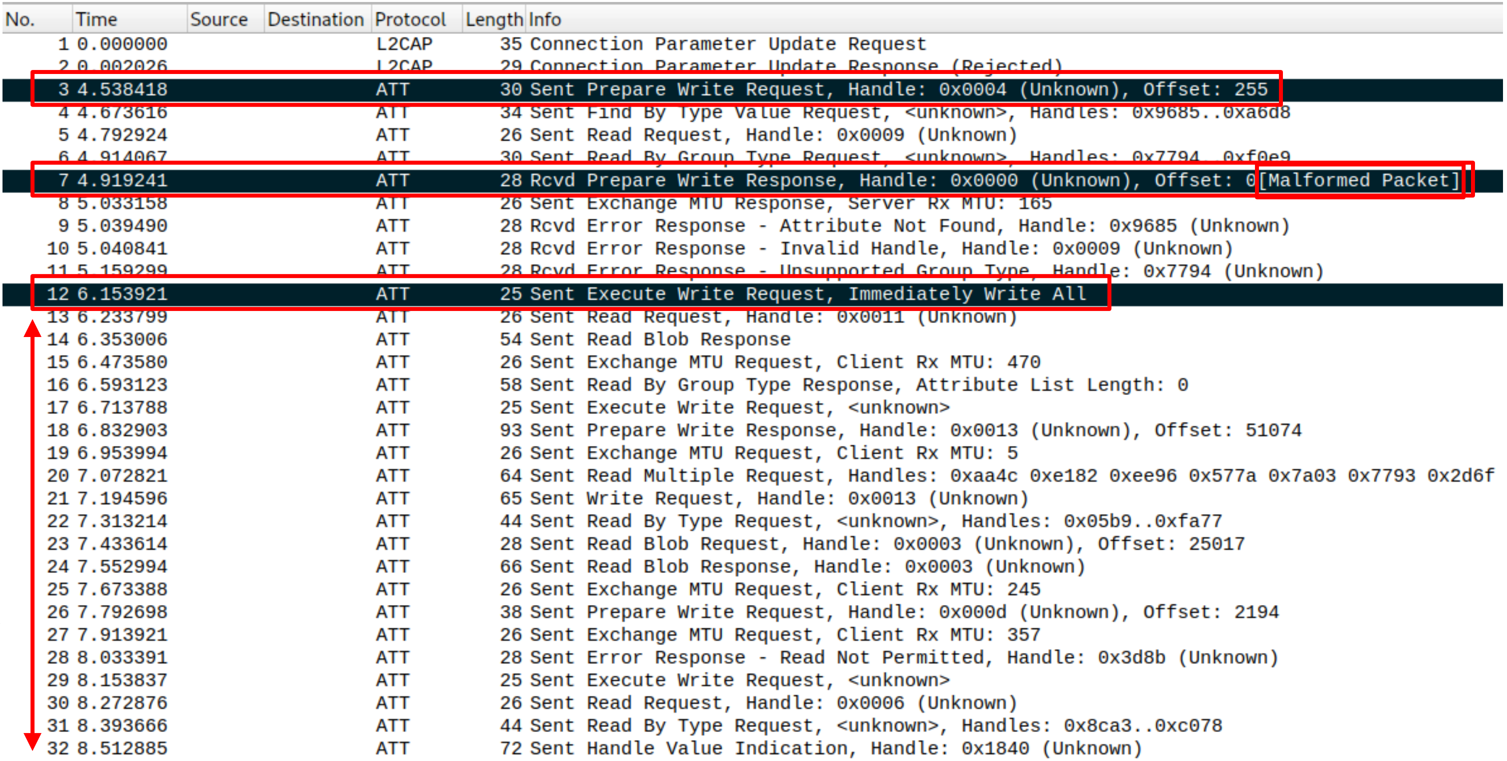


```
    else if ( (p_attr->uuid_type == GATT_ATTR_UUID_TYPE_16) &&
              (p_attr->uuid == GATT_UUID_CHAR_CLIENT_CONFIG ||
               p_attr->uuid == GATT_UUID_CHAR_SRVR_CONFIG) )
    {
        if (op_code == GATT_REQ_PREPARE_WRITE && offset != 0) {  /*BUG*/
            status = GATT_NOT_LONG;
            GATT_TRACE_ERROR( "gatts_write_attr_perm_check - GATT_NOT_LONG");
        } else if (len != max_size) { /* data does not match the required format */
            status = GATT_INVALID_ATTR_LEN;
            GATT_TRACE_ERROR( "gatts_write_attr_perm_check - GATT_INVALID_PDU");
        } else {
            status = GATT_SUCCESS;
        }
    }
}
```

BUG: UNRESPONSIVE SERVER

Bluedroid Prepare Write Request weird behavior

Client Characteristic Configuration descriptor or *Client Supported Features* characteristic



REQ_1

RSP_1

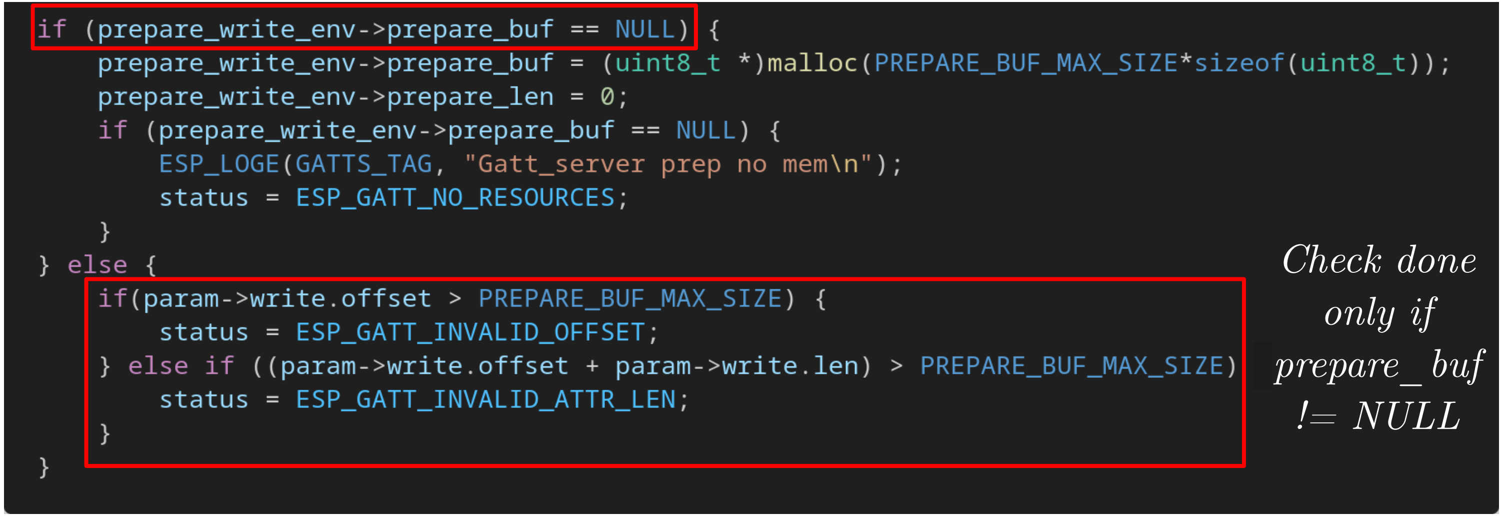
REQ_2

Server doesn't respond anymore

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000			L2CAP	35	Connection Parameter Update Request
2	0.002026			L2CAP	29	Connection Parameter Update Response (Rejected)
3	4.538418			ATT	30	Sent Prepare Write Request, Handle: 0x0004 (Unknown), Offset: 255
4	4.673616			ATT	34	Sent Find By Type Value Request, <unknown>, Handles: 0x9685..0xa6d8
5	4.792924			ATT	26	Sent Read Request, Handle: 0x0009 (Unknown)
6	4.914067			ATT	30	Sent Read By Group Type Request, <unknown>, Handles: 0x7794..0xf0e9
7	4.919241			ATT	28	Rcvd Prepare Write Response, Handle: 0x0000 (Unknown), Offset: 0[Malformed Packet]
8	5.033158			ATT	26	Sent Exchange MTU Response, Server Rx MTU: 165
9	5.039490			ATT	28	Rcvd Error Response - Attribute Not Found, Handle: 0x9685 (Unknown)
10	5.040841			ATT	28	Rcvd Error Response - Invalid Handle, Handle: 0x0009 (Unknown)
11	5.159299			ATT	28	Rcvd Error Response - Unsupported Group Type, Handle: 0x7794 (Unknown)
12	6.153921			ATT	25	Sent Execute Write Request, Immediately Write All
13	6.233799			ATT	26	Sent Read Request, Handle: 0x0011 (Unknown)
14	6.353006			ATT	54	Sent Read Blob Response
15	6.473580			ATT	26	Sent Exchange MTU Request, Client Rx MTU: 470
16	6.593123			ATT	58	Sent Read By Group Type Response, Attribute List Length: 0
17	6.713788			ATT	25	Sent Execute Write Request, <unknown>
18	6.832903			ATT	93	Sent Prepare Write Response, Handle: 0x0013 (Unknown), Offset: 51074
19	6.953994			ATT	26	Sent Exchange MTU Request, Client Rx MTU: 5
20	7.072821			ATT	64	Sent Read Multiple Request, Handles: 0xaa4c 0xe182 0xee96 0x577a 0x7a03 0x7793 0x2d6f
21	7.194596			ATT	65	Sent Write Request, Handle: 0x0013 (Unknown)
22	7.313214			ATT	44	Sent Read By Type Request, <unknown>, Handles: 0x05b9..0xfa77
23	7.433614			ATT	28	Sent Read Blob Request, Handle: 0x0003 (Unknown), Offset: 25017
24	7.552994			ATT	66	Sent Read Blob Response, Handle: 0x0003 (Unknown)
25	7.673388			ATT	26	Sent Exchange MTU Request, Client Rx MTU: 245
26	7.792698			ATT	38	Sent Prepare Write Request, Handle: 0x000d (Unknown), Offset: 2194
27	7.913921			ATT	26	Sent Exchange MTU Request, Client Rx MTU: 357
28	8.033391			ATT	28	Sent Error Response - Read Not Permitted, Handle: 0x3d8b (Unknown)
29	8.153837			ATT	25	Sent Execute Write Request, <unknown>
30	8.272876			ATT	26	Sent Read Request, Handle: 0x0006 (Unknown)
31	8.393666			ATT	44	Sent Read By Type Request, <unknown>, Handles: 0x8ca3..0xc078
32	8.512885			ATT	72	Sent Handle Value Indication, Handle: 0x1840 (Unknown)

VULN: OUT-OF-BOUNDS WRITE

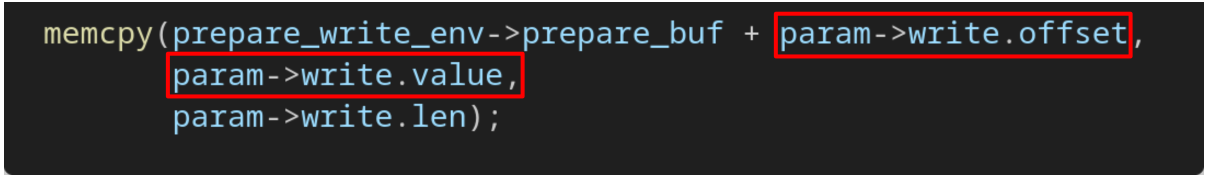
VULN: OUT-OF-BOUNDS WRITE



```c
if (prepare_write_env->prepare_buf == NULL) {
    prepare_write_env->prepare_buf = (uint8_t *)malloc(PREPARE_BUF_MAX_SIZE*sizeof(uint8_t));
    prepare_write_env->prepare_len = 0;
    if (prepare_write_env->prepare_buf == NULL) {
        ESP_LOGE(GATTS_TAG, "Gatt_server prep no mem\n");
        status = ESP_GATT_NO_RESOURCES;
    }
} else {
    if(param->write.offset > PREPARE_BUF_MAX_SIZE) {
        status = ESP_GATT_INVALID_OFFSET;
    } else if ((param->write.offset + param->write.len) > PREPARE_BUF_MAX_SIZE)
        status = ESP_GATT_INVALID_ATTR_LEN;
    }
}
```

Check done only if prepare_buf != NULL

Bluedroid Gatt server example from ESPRESSIF



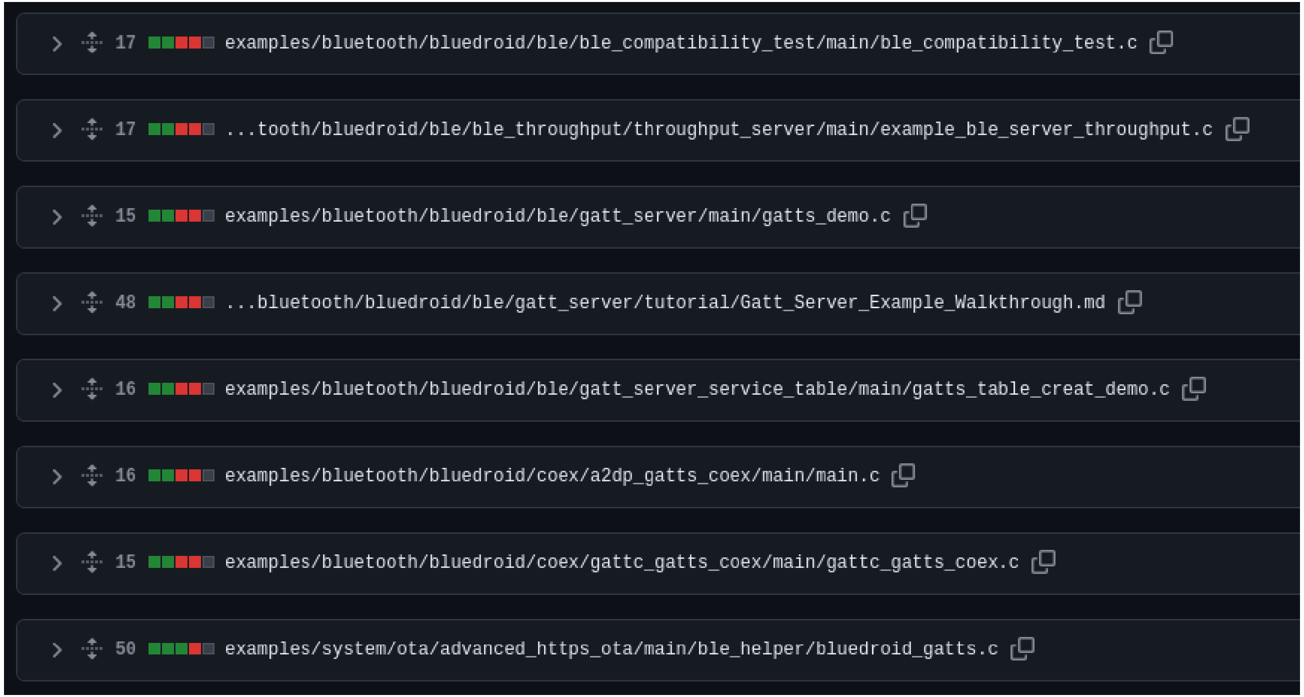
```c
memcpy(prepare_write_env->prepare_buf + param->write.offset,
       param->write.value,
       param->write.len);
```

← Out-of-Bounds Write

VULN: OUT-OF-BOUNDS WRITE

ESPRESSIF's response: " *...the impact of this issue on customers is minor, lacking any substantial consequences.*"

But...

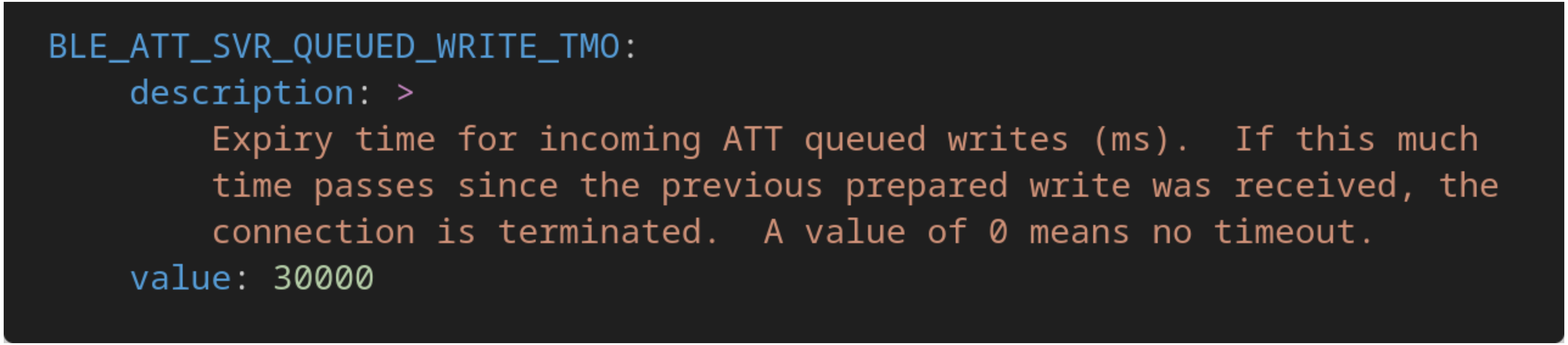


```
17 🟩🟩🟥🟥⬛ examples/bluetooth/bluedroid/ble/ble_compatibility_test/main/ble_compatibility_test.c
17 🟩🟩🟥🟥⬛ ...tooth/bluedroid/ble/ble_throughput/throughput_server/main/example_ble_server_throughput.c
15 🟩🟩🟥🟥⬛ examples/bluetooth/bluedroid/ble/gatt_server/main/gatts_demo.c
48 🟩🟩🟥🟥⬛ ...bluetooth/bluedroid/ble/gatt_server/tutorial/Gatt_Server_Example_Walkthrough.md
16 🟩🟩🟥🟥⬛ examples/bluetooth/bluedroid/ble/gatt_server_service_table/main/gatts_table_creat_demo.c
16 🟩🟩🟥🟥⬛ examples/bluetooth/bluedroid/coex/a2dp_gatts_coex/main/main.c
15 🟩🟩🟥🟥⬛ examples/bluetooth/bluedroid/coex/gattc_gatts_coex/main/gattc_gatts_coex.c
50 🟩🟩🟩🟥⬛ examples/system/ota/advanced_https_ota/main/ble_helper/bluedroid_gatts.c
```

7 code examples were impacted!

VULN: DENIAL OF SERVICE

NimBLE timeout feature:



```
BLE_ATT_SVR_QUEUED_WRITE_TMO:
    description: >
        Expiry time for incoming ATT queued writes (ms).  If this much
        time passes since the previous prepared write was received, the
        connection is terminated.  A value of 0 means no timeout.
    value: 30000
```

"***A transaction not completed within 30 seconds shall time out.*** *Such a transaction shall be considered to have failed, and the local higher layers shall be informed of this failure.*" [Spec Vol.3 Part.F 3.3.3]

VULN: DENIAL OF SERVICE

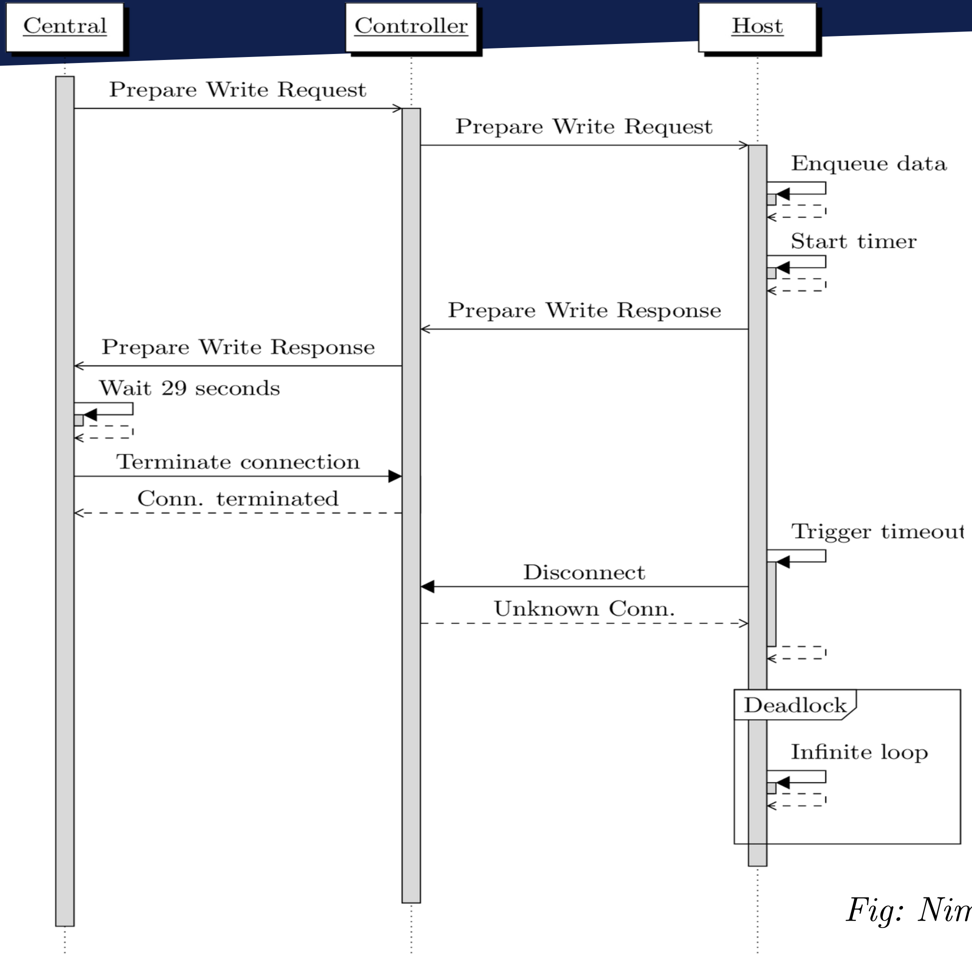
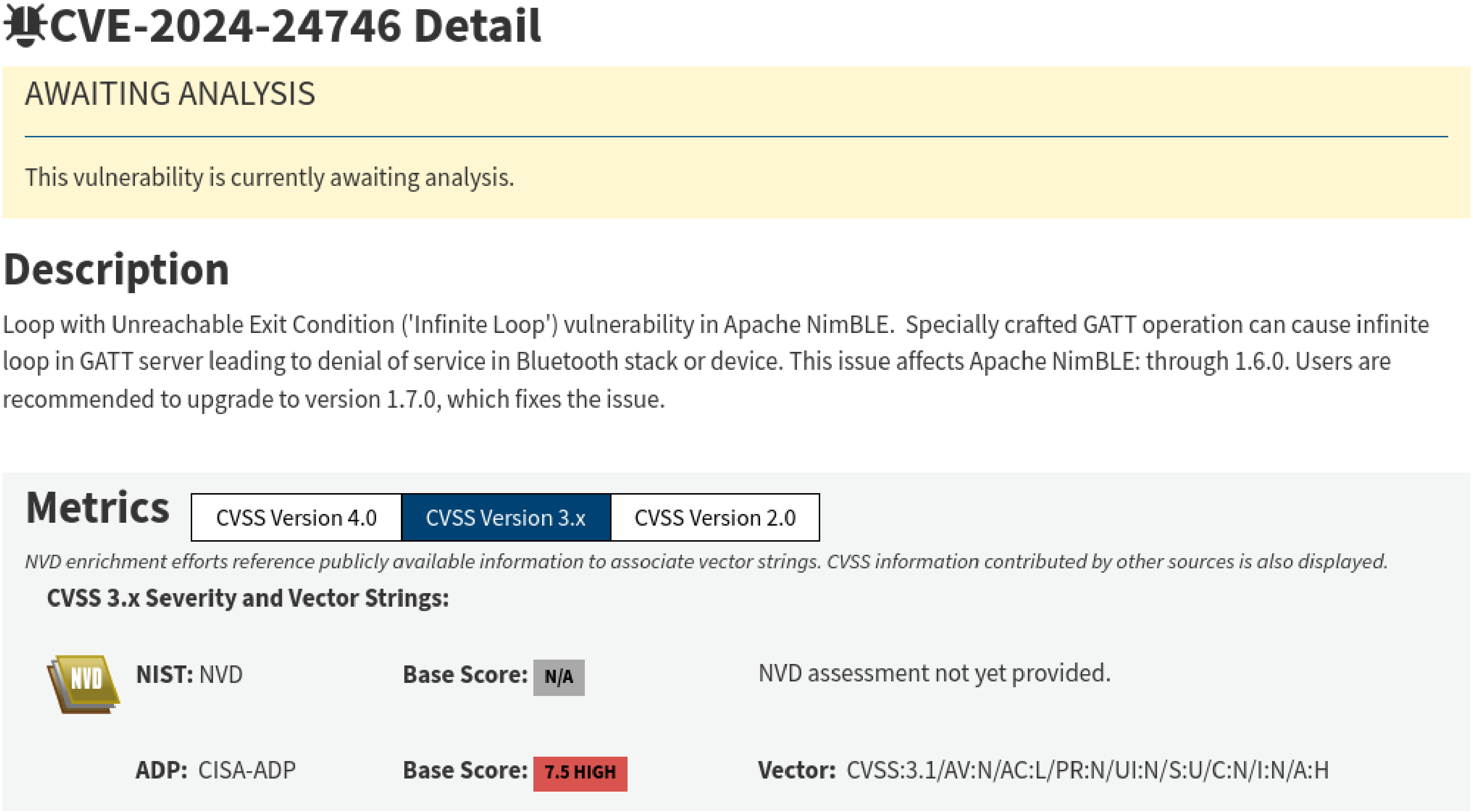


Fig: NimBLE Denial of Service

VULN: DENIAL OF SERVICE



CVE-2024-24746 Detail

AWAITING ANALYSIS

This vulnerability is currently awaiting analysis.

Description

Loop with Unreachable Exit Condition ('Infinite Loop') vulnerability in Apache NimBLE. Specially crafted GATT operation can cause infinite loop in GATT server leading to denial of service in Bluetooth stack or device. This issue affects Apache NimBLE: through 1.6.0. Users are recommended to upgrade to version 1.7.0, which fixes the issue.

Metrics

NVD enrichment efforts reference publicly available information to associate vector strings. CVSS information contributed by other sources is also displayed.

CVSS 3.x Severity and Vector Strings:

NIST: NVD	**Base Score:** N/A	NVD assessment not yet provided.
ADP: CISA-ADP	**Base Score:** 7.5 HIGH	**Vector:** CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

~~Fuzzing real-world devices~~

How to annoy your colleagues!

HOW TO ANNOY YOUR COLLEAGUES!

HOW TO ANNOY YOUR COLLEAGUES!

Which devices to target?

SONY
WF-1000XM4

SONY
WH-1000XM4

SONY
WH-1000XM5

HOW TO ANNOY YOUR COLLEAGUES!

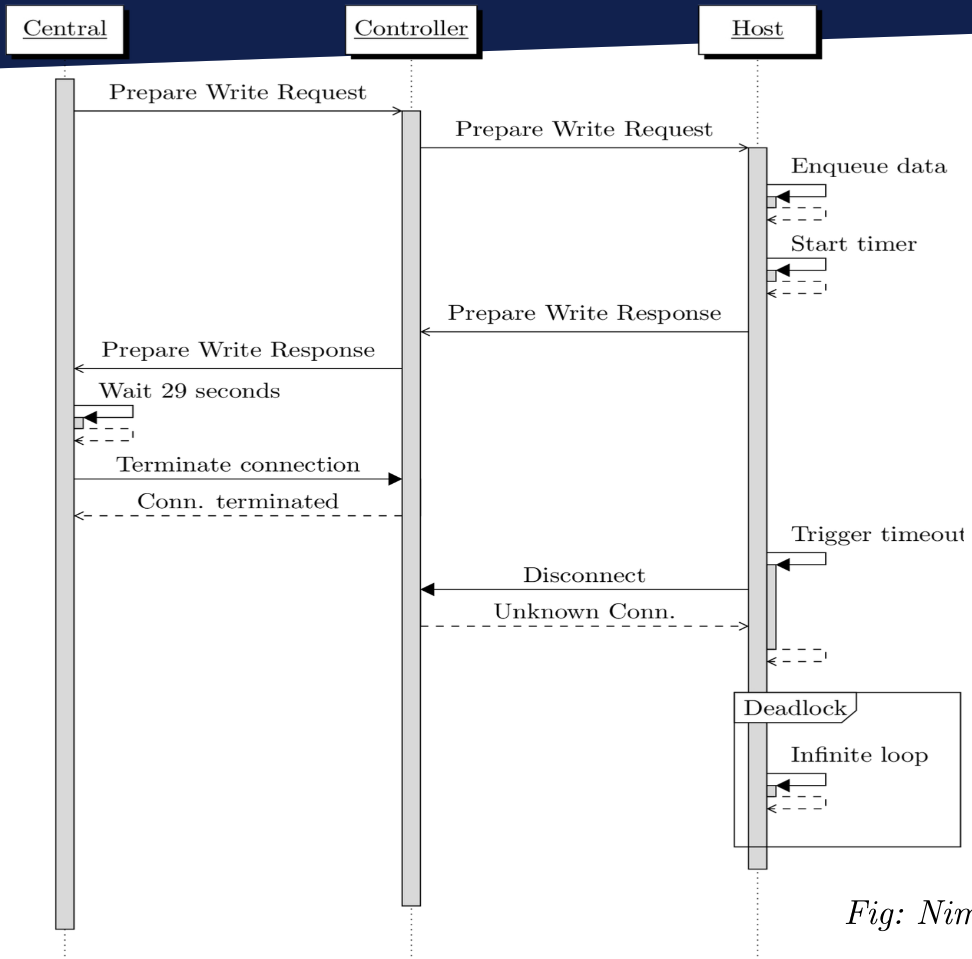
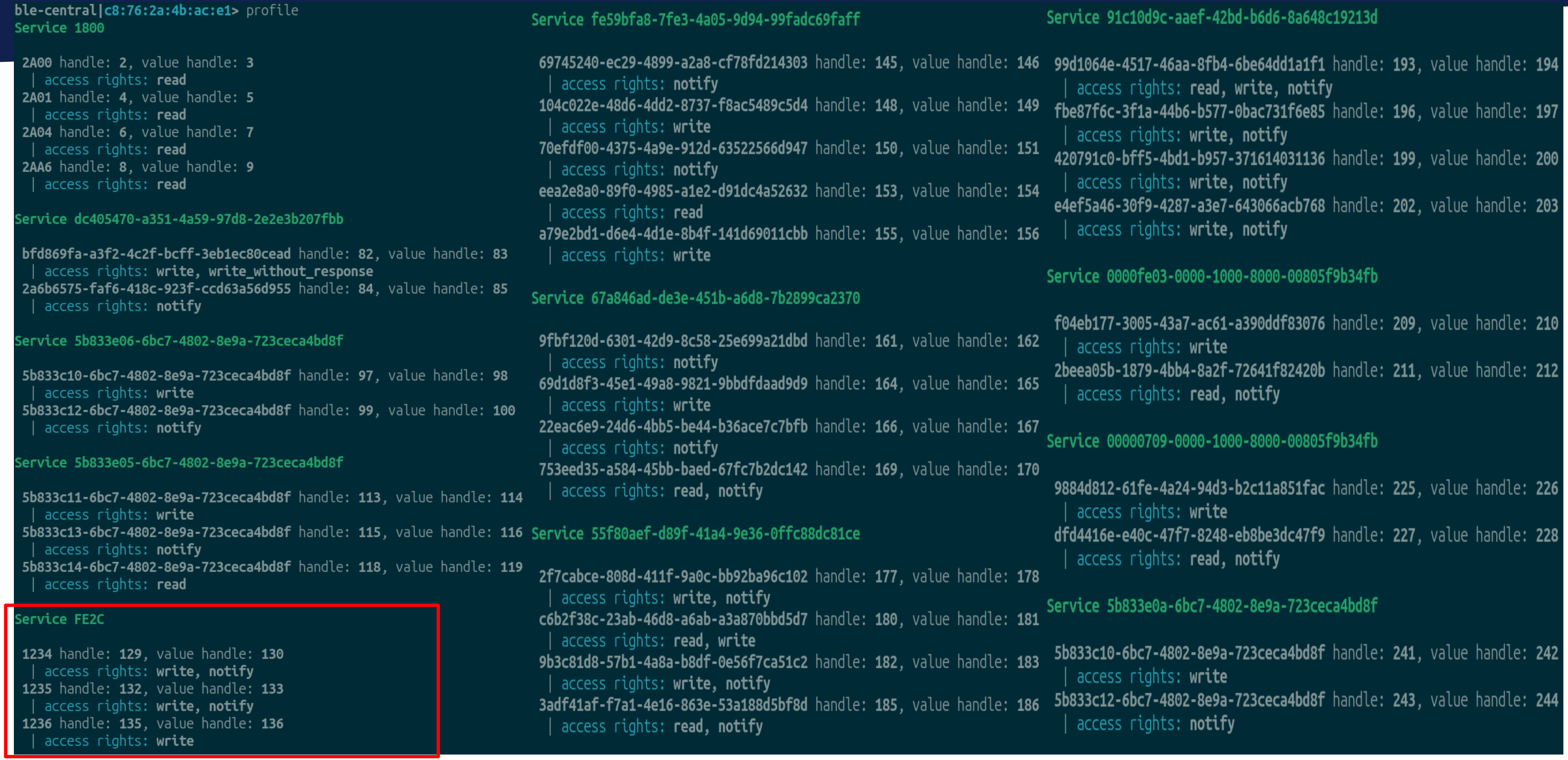


Fig: NimBLE Denial of Service

HOW TO ANNOY YOUR COLLEAGUES!



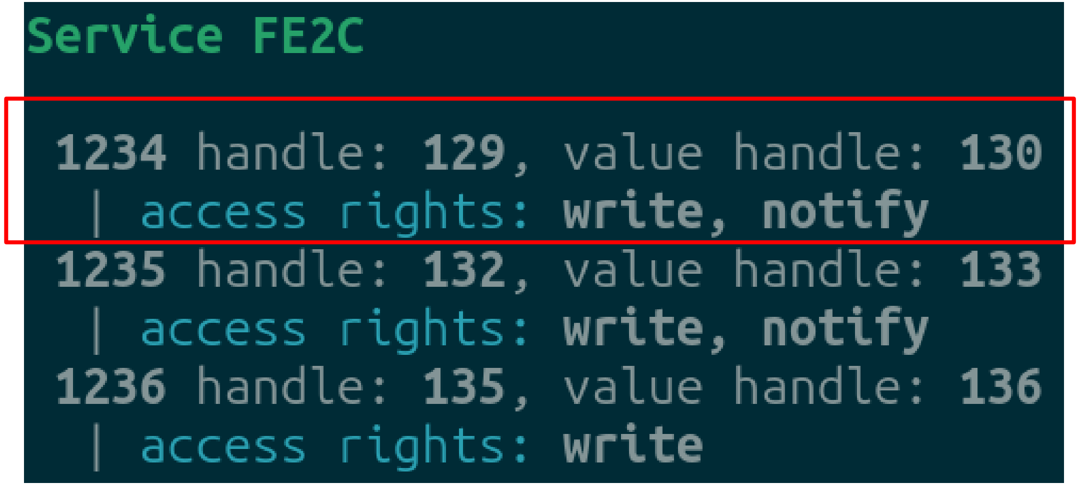
```
ble-central|c8:76:2a:4b:ac:e1> profile
Service 1800

 2A00 handle: 2, value handle: 3
  | access rights: read
 2A01 handle: 4, value handle: 5
  | access rights: read
 2A04 handle: 6, value handle: 7
  | access rights: read
 2AA6 handle: 8, value handle: 9
  | access rights: read

Service dc405470-a351-4a59-97d8-2e2e3b207fbb

 bfd869fa-a3f2-4c2f-bcff-3eb1ec80cead handle: 82, value handle: 83
  | access rights: write, write_without_response
 2a6b6575-faf6-418c-923f-ccd63a56d955 handle: 84, value handle: 85
  | access rights: notify

Service 5b833e06-6bc7-4802-8e9a-723ceca4bd8f

 5b833c10-6bc7-4802-8e9a-723ceca4bd8f handle: 97, value handle: 98
  | access rights: write
 5b833c12-6bc7-4802-8e9a-723ceca4bd8f handle: 99, value handle: 100
  | access rights: notify

Service 5b833e05-6bc7-4802-8e9a-723ceca4bd8f

 5b833c11-6bc7-4802-8e9a-723ceca4bd8f handle: 113, value handle: 114
  | access rights: write
 5b833c13-6bc7-4802-8e9a-723ceca4bd8f handle: 115, value handle: 116
  | access rights: notify
 5b833c14-6bc7-4802-8e9a-723ceca4bd8f handle: 118, value handle: 119
  | access rights: read

Service FE2C

 1234 handle: 129, value handle: 130
  | access rights: write, notify
 1235 handle: 132, value handle: 133
  | access rights: write, notify
 1236 handle: 135, value handle: 136
  | access rights: write
```

```
Service fe59bfa8-7fe3-4a05-9d94-99fadc69faff

 69745240-ec29-4899-a2a8-cf78fd214303 handle: 145, value handle: 146
  | access rights: notify
 104c022e-48d6-4dd2-8737-f8ac5489c5d4 handle: 148, value handle: 149
  | access rights: write
 70efdf00-4375-4a9e-912d-63522566d947 handle: 150, value handle: 151
  | access rights: notify
 eea2e8a0-89f0-4985-a1e2-d91dc4a52632 handle: 153, value handle: 154
  | access rights: read
 a79e2bd1-d6e4-4d1e-8b4f-141d69011cbb handle: 155, value handle: 156
  | access rights: write

Service 67a846ad-de3e-451b-a6d8-7b2899ca2370

 9fbf120d-6301-42d9-8c58-25e699a21dbd handle: 161, value handle: 162
  | access rights: notify
 69d1d8f3-45e1-49a8-9821-9bbdfdaad9d9 handle: 164, value handle: 165
  | access rights: write
 22eac6e9-24d6-4bb5-be44-b36ace7c7bfb handle: 166, value handle: 167
  | access rights: notify
 753eed35-a584-45bb-baed-67fc7b2dc142 handle: 169, value handle: 170
  | access rights: read, notify

Service 55f80aef-d89f-41a4-9e36-0ffc88dc81ce

 2f7cabce-808d-411f-9a0c-bb92ba96c102 handle: 177, value handle: 178
  | access rights: write, notify
 c6b2f38c-23ab-46d8-a6ab-a3a870bbd5d7 handle: 180, value handle: 181
  | access rights: read, write
 9b3c81d8-57b1-4a8a-b8df-0e56f7ca51c2 handle: 182, value handle: 183
  | access rights: write, notify
 3adf41af-f7a1-4e16-863e-53a188d5bf8d handle: 185, value handle: 186
  | access rights: read, notify
```

```
Service 91c10d9c-aaef-42bd-b6d6-8a648c19213d

 99d1064e-4517-46aa-8fb4-6be64dd1a1f1 handle: 193, value handle: 194
  | access rights: read, write, notify
 fbe87f6c-3f1a-44b6-b577-0bac731f6e85 handle: 196, value handle: 197
  | access rights: write, notify
 420791c0-bff5-4bd1-b957-371614031136 handle: 199, value handle: 200
  | access rights: write, notify
 e4ef5a46-30f9-4287-a3e7-643066acb768 handle: 202, value handle: 203
  | access rights: write, notify

Service 0000fe03-0000-1000-8000-00805f9b34fb

 f04eb177-3005-43a7-ac61-a390ddf83076 handle: 209, value handle: 210
  | access rights: write
 2beea05b-1879-4bb4-8a2f-72641f82420b handle: 211, value handle: 212
  | access rights: read, notify

Service 00000709-0000-1000-8000-00805f9b34fb

 9884d812-61fe-4a24-94d3-b2c11a851fac handle: 225, value handle: 226
  | access rights: write
 dfd4416e-e40c-47f7-8248-eb8be3dc47f9 handle: 227, value handle: 228
  | access rights: read, notify

Service 5b833e0a-6bc7-4802-8e9a-723ceca4bd8f

 5b833c10-6bc7-4802-8e9a-723ceca4bd8f handle: 241, value handle: 242
  | access rights: write
 5b833c12-6bc7-4802-8e9a-723ceca4bd8f handle: 243, value handle: 244
  | access rights: notify
```

Fig: Sony WH-1000XM4 GATT Server

HOW TO ANNOY YOUR COLLEAGUES!



```
Service FE2C

 1234 handle: 129, value handle: 130
   | access rights: write, notify
 1235 handle: 132, value handle: 133
   | access rights: write, notify
 1236 handle: 135, value handle: 136
   | access rights: write
```

Fig: Fast Pair Service

HOW TO ANNOY YOUR COLLEAGUES!

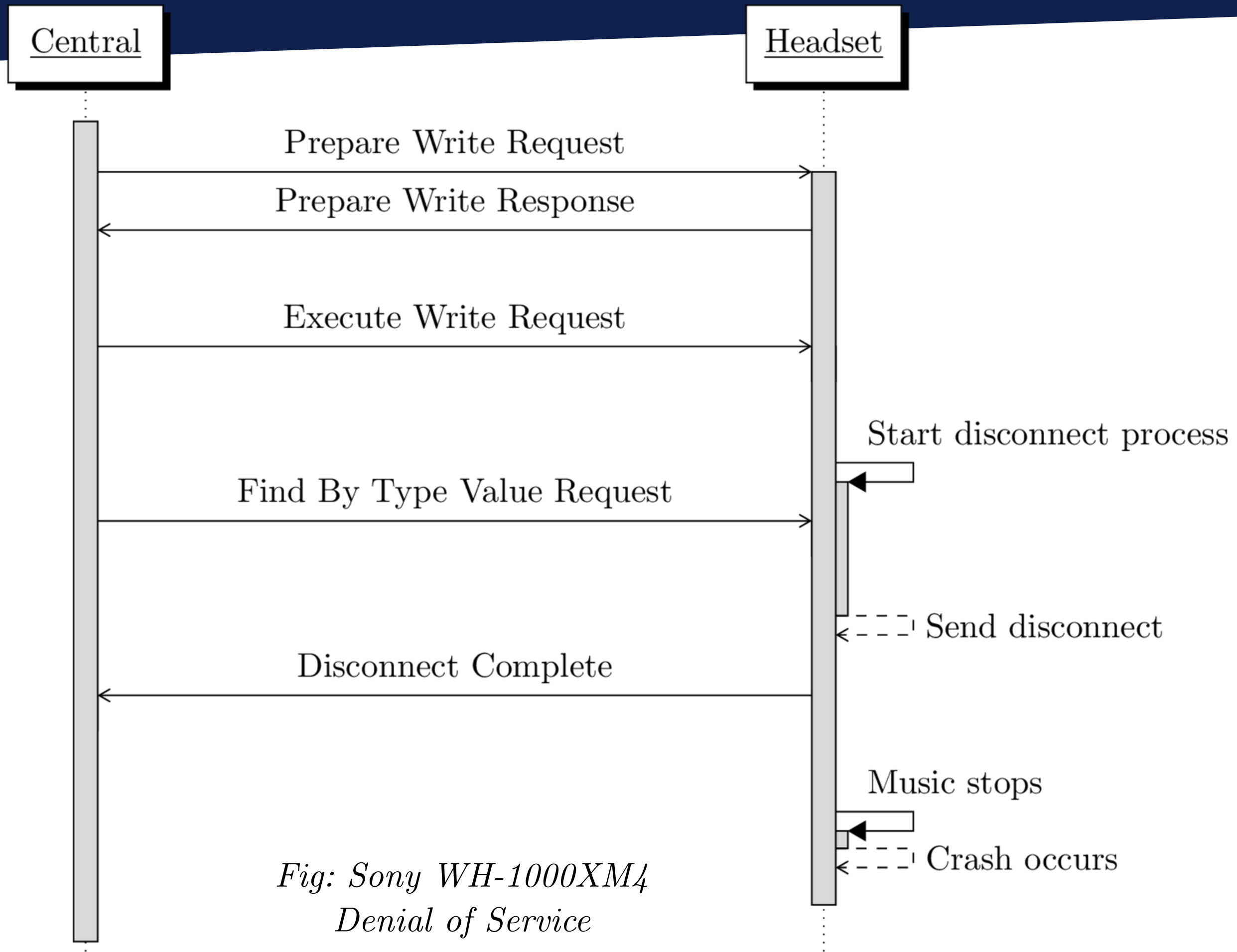


Fig: Sony WH-1000XM4
Denial of Service

HOW TO ANNOY YOUR COLLEAGUES!

Affected devices:

SONY
WF-1000XM4

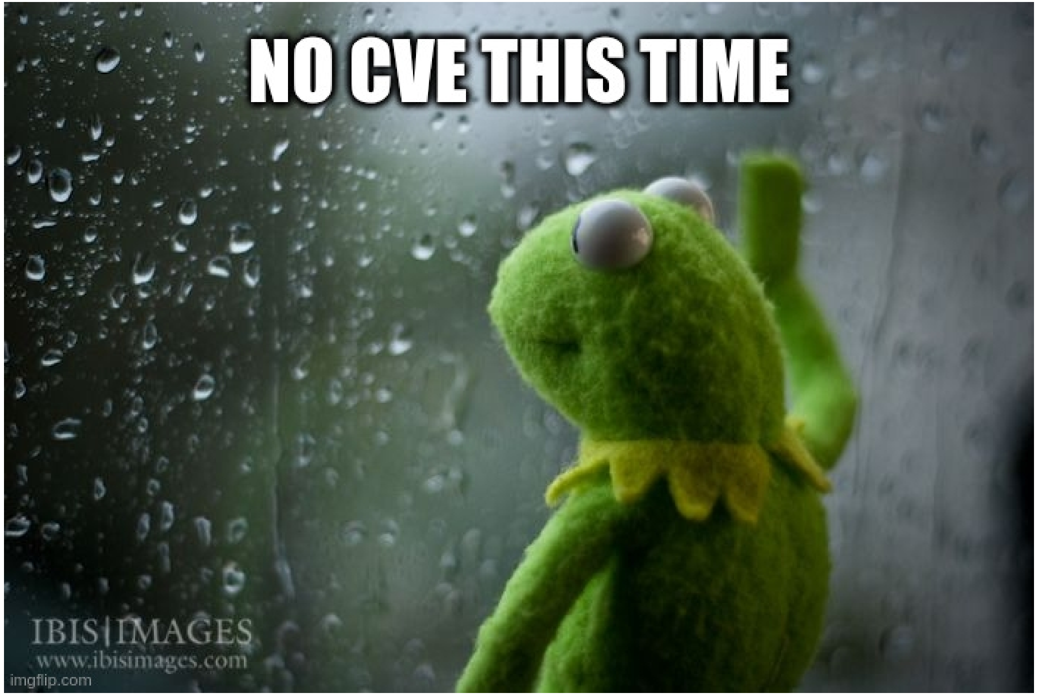
SONY
WH-1000XM4

SONY
WH-1000XM5

https://github.com/quarkslab/ble-gatt-fuzzing/poc

HOW TO ANNOY YOUR COLLEAGUES!

Unfortunately...



HOW TO ANNOY YOUR COLLEAGUES!

Fixes:

- *WH-1000XM4* version **2.6.0** (Released on October 17th)
- *WF-1000XM4* version **2.1.0** (Released on October 17th)
- *WH-1000XM5* version **2.3.1** (Released on October 2nd)

Limitations

Quarkslab

LIMITATIONS

BLE version

- Based our attack scenarios on BLE version 4.2 and not on last one which is 6.0

GATT Servers

- Since a GATT server is populated by the stack and by the application, a poorly implemented GATT server is less likely to trigger bugs

Results Analysis

- Didn’t have enough time to incorporate an automated analysis method

Conclusion

CONCLUSION

Observations

- Lack of standardization of BLE stack implementation leads to developer errors.
- Proximity between GATT and application layers may lead to more vulnerabilities.
- Over-the-air fuzzing is relevant even if not fast.

For more details

- Check out the blogpost and paper.
- *https://blog.quarkslab.com/bluetooth-low-energy-gatt-fuzzing.html*

Thank you!

Quarkslab

Questions?

bboyer@quarkslab.com