ERNW
providing security.

SECURE MOBILE NETWORKING

FRANKENSTEIN

git://github.com/seemoo-lab/frankenstein

Advanced Wireless Fuzzing to Exploit New Bluetooth Escalation Targets

Jan Ruge, Jiska Classen, Francesco Gringoli, Matthias Hollick

ERNW
providing security.

SECURE MOBILE NETWORKING

Motivation

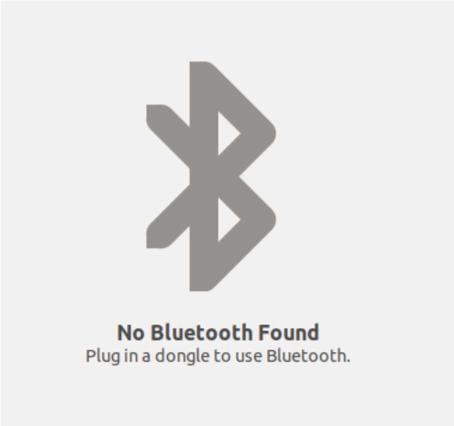
- Radio protocols have huge attack surface
- We focus on Bluetooth firmware
- Large unauthenticated attack surface
  - Devices are connectable by default
  - Lower protocols terminated in firmware

```
hammel@ernw ~ > sudo l2ping 20:73:5B:17:69:31
Ping: 20:73:5B:17:69:31 from 4C:1D:96:B7:12:03 (data size 44) ...
44 bytes from 20:73:5B:17:69:31 id 0 time 16.69ms
44 bytes from 20:73:5B:17:69:31 id 1 time 23.70ms
44 bytes from 20:73:5B:17:69:31 id 2 time 23.68ms
44 bytes from 20:73:5B:17:69:31 id 3 time 24.93ms
44 bytes from 20:73:5B:17:69:31 id 4 time 23.69ms
44 bytes from 20:73:5B:17:69:31 id 5 time 23.71ms
44 bytes from 20:73:5B:17:69:31 id 6 time 23.70ms
44 bytes from 20:73:5B:17:69:31 id 7 time 23.61ms
44 bytes from 20:73:5B:17:69:31 id 8 time 23.75ms
44 bytes from 20:73:5B:17:69:31 id 9 time 22.41ms
44 bytes from 20:73:5B:17:69:31 id 10 time 24.93ms
44 bytes from 20:73:5B:17:69:31 id 11 time 23.66ms
44 bytes from 20:73:5B:17:69:31 id 12 time 25.06ms
44 bytes from 20:73:5B:17:69:31 id 13 time 23.58ms
44 bytes from 20:73:5B:17:69:31 id 14 time 22.35ms
44 bytes from 20:73:5B:17:69:31 id 15 time 25.01ms
44 bytes from 20:73:5B:17:69:31 id 16 time 39.93ms
```

Why Emulation ?

- Reverse engineering is hard
- Embedded device debugging is hard
- Static analysis does not give context
  - ~300 functions with *RX*
  - How are those invoked?

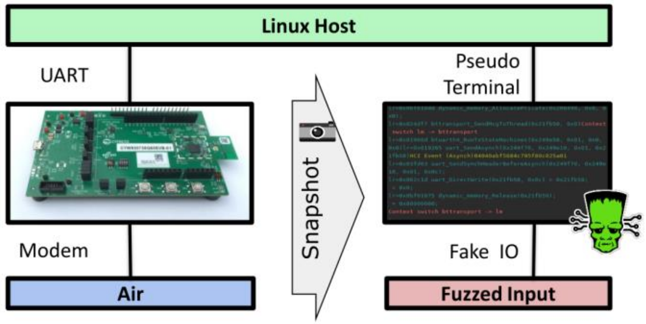
- Narrow down relevant code



Firmware Emulation - Modifications

- Added debug output
- Supports
  - Threading
  - HCI injection/extraction
  - Raw wireless frames injection/extraction

- Attach emulator as Bluetooth device
- Read wireless frames from stdin



Firmware Emulation - Concept

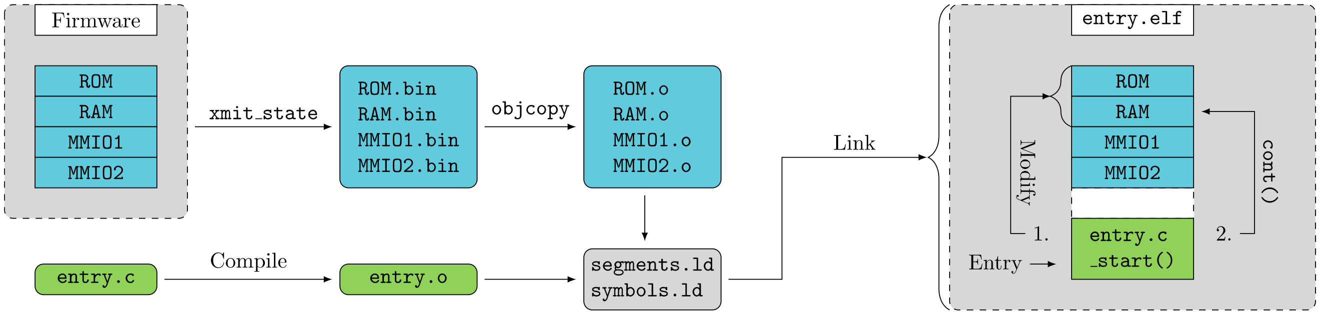
- Hooking by rewriting function prologue
- Extract clean firmware state
  - Memory
  - Memory Mapped I/O
  - Registers

- Restore memory layout
- Restore registers
- Go?

```
xmit_state:
push {r0-r12,lr}
ldr r0, =saved_regs
str sp, [r0]

bl xmit_memory

cont:
ldr r0, =saved_regs
ldr sp, [r0]
pop {r0-r12,lr}

bx lr
```

Firmware Emulation - Concept

- Link C code against firmware
- Assemble to ELF file
- Execute using `qemu-arm`

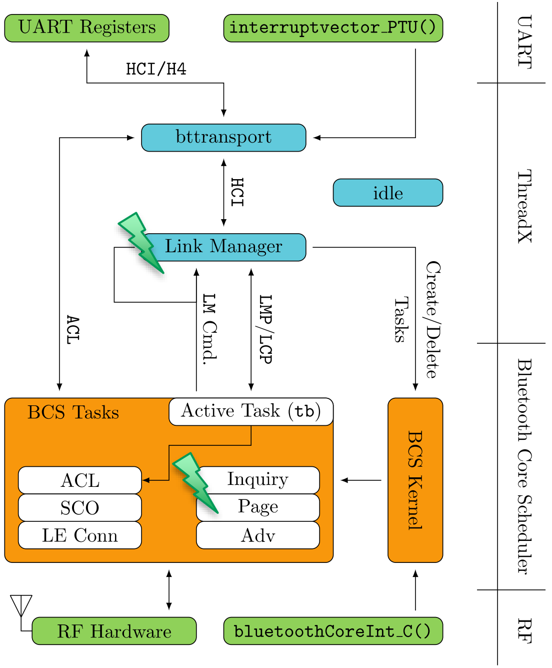


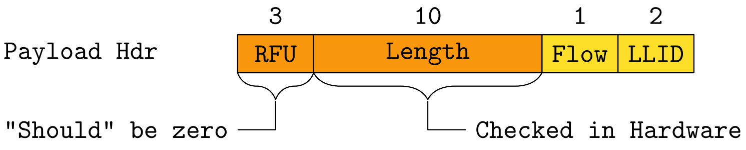
Oops, that’s not my bug…

```
Slot01
lr=0x098aab bcs_kernelBtProgIntEnable(0x02, 0x02, 0x02, 0x50);
lr=0x040519 bcs_kernelSlotCbFunctions()lr=0x0bf01469 bcs_SlotCbFunctions()lr=0x02124d lm_sendInqFHS(0x282d94)lr=0x02cb61 dynamic_me
Context switch idle -> lm
lr=0x02d12f lm_handleInqFHS(0x40)lr=0x02cc53 lc_handleInqResult(0x21fb50)lr=0x041d91 inqfilter_isBdAddrRegistered(0x21fb58, 0x01);
lr=0x041dc3 inqfilter_registerBdAddr(0x21fb58, 0x01);
lr=0x041dfb bthci_event_SendInquiryResultEvent(0x21fb50)lr=0x024ea5 dynamic_memory_AllocateOrDie(0x0109) = 0x2209d4;
lr=0x02503f eir_getReceivedEIR(0x21fb58)lr=0x04db5f __rt_memcpy(0x2209ed, 0x221484, 0x0645)Heap Corruption Detected
pool = 0x2004e0
pool->size = 0x010c
free_chunk = 0x220bf0
8788898a | 8b8c8d8e8f909192939495969798999a9b9c9d9e9fa0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbebfc0c1c2c3c4c5c6
dedfe0e1e2e3e4e5e6e7e8e9eaebecedeeef10037e0ac56f450460467a0f0ca2bc9a950a24f562bd95da324c294eb51c1323862e67dfd8480502ccc884946674786
e16d4fb39a72d501f4176a084d444dc4958be24472ec4dc77f42f718a804d519b94ffc72f6e72d5fa1efd21131a0ba501721c1c7cb6d5b62254c39036fa54c3a249
4242424242424242
qemu: uncaught target signal 11 (Segmentation fault) - core dumped
Segmentation fault (core dumped)
```

CVE-2019-11516 (Broadcom)

- Heap corruption during device inquiry
  - Affects devices ~2010 - 2018
  - Exploited for chip RCE
  - Used by: Samsung, Apple, …
- Bug located in Link Manager and BCS





Outcome & Conclusion

- Large scale firmware emulation possible
- Advantages
  - Same layer of abstraction – More readable code
  - No API calls – More performance
  - Same code base for patching device*
- Disadvantages
  - Process remains (less) tedious

- git://github.com/seemoo-lab/frankenstein

* and find Android 0-Click RCE by accident

Type	Severity	Updated AOSP versions
DoS	Moderate	10
RCE	Critical	8.0, 8.1, 9