Grey Box Fuzzing the IoT

Program Monitoring for Memory and Power Constraint Embedded Devices

Kevin Läufer <laeufer@berkeley.edu>

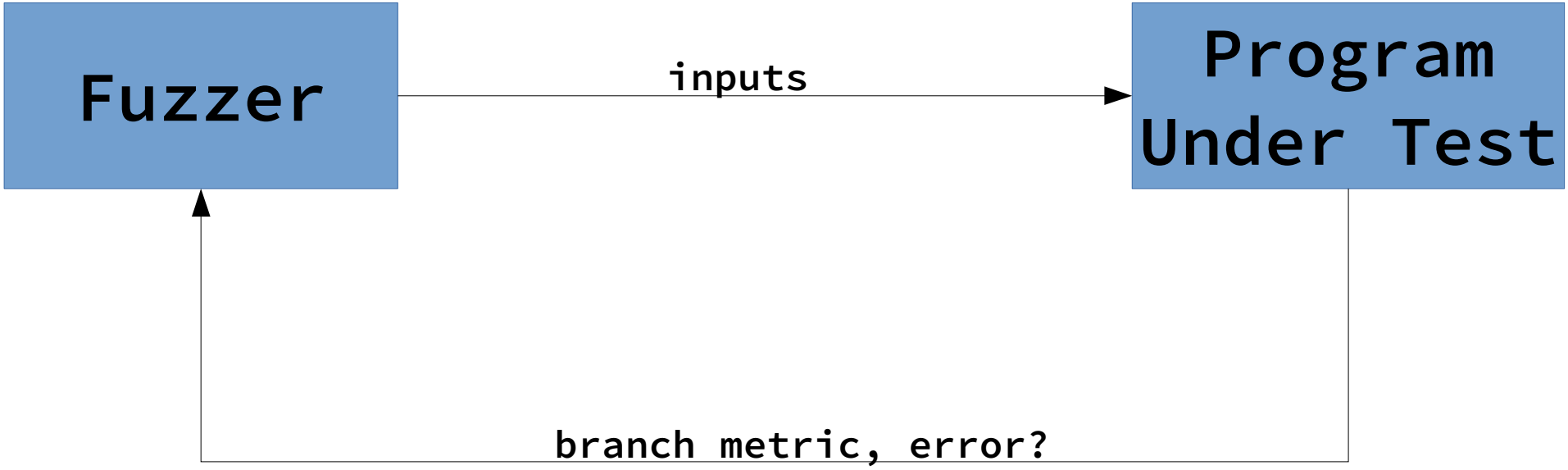
Problems in Embedded Programming and Testing

- No memory protection:
  - all processes can access all memory
  - hard to detect and contain failures
- Still predominantly programmed in C
- No logging, no automatic stack traces
- Lacks tools for automated testing

Goal: Enable Fuzzing on Low Power Embedded Devices

How?

Fuzzing



Fuzzing Embedded Devices

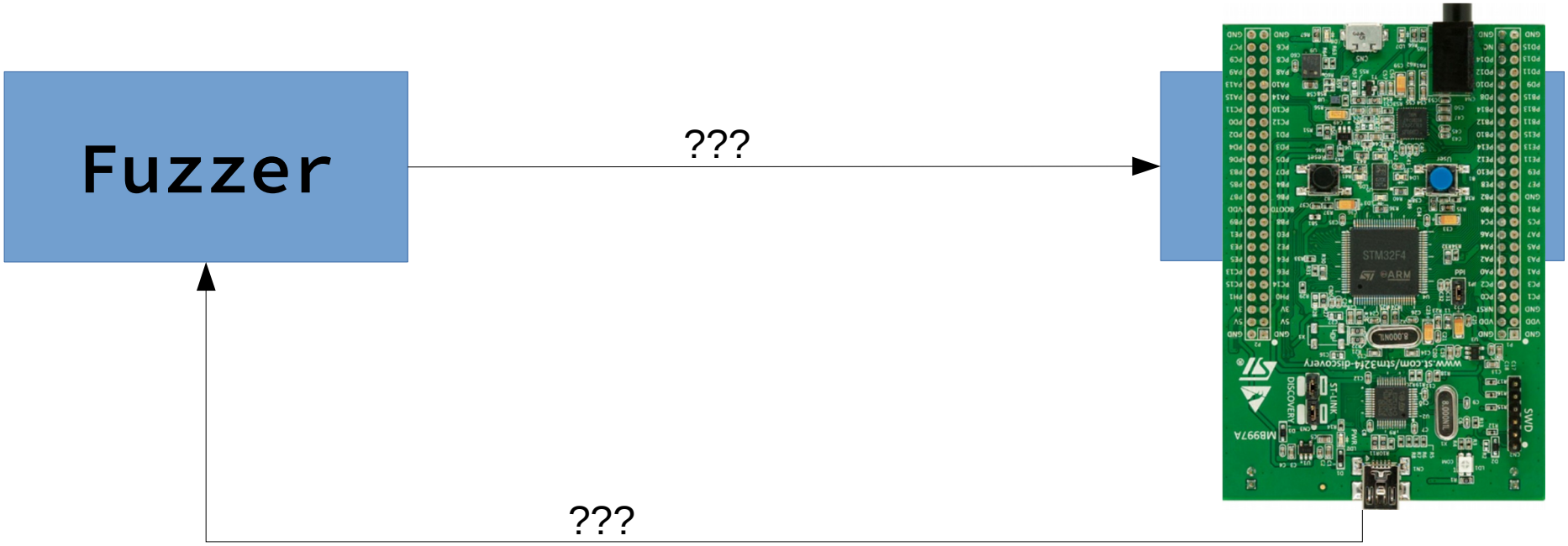


Image Source: STMicroelectronics

Fuzzing Embedded Devices

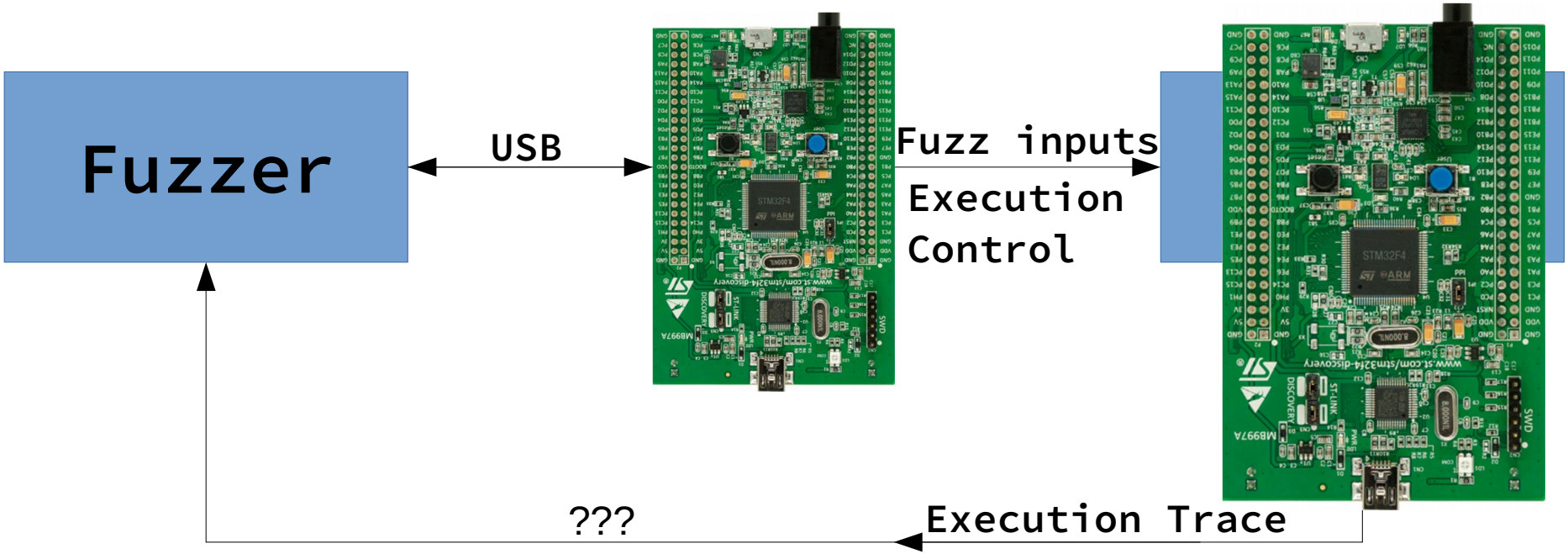


Image Source: STMicroelectronics

Fuzzing Embedded Devices

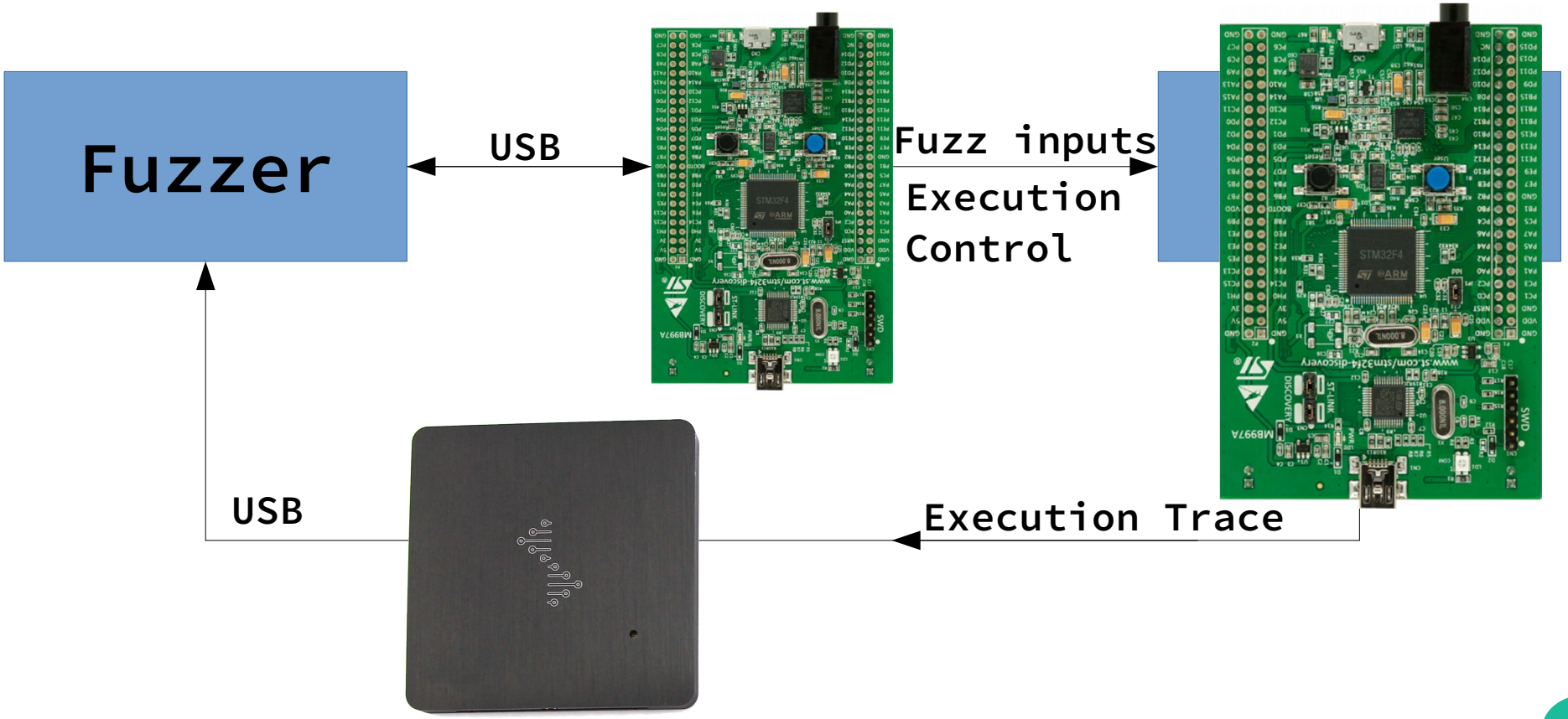
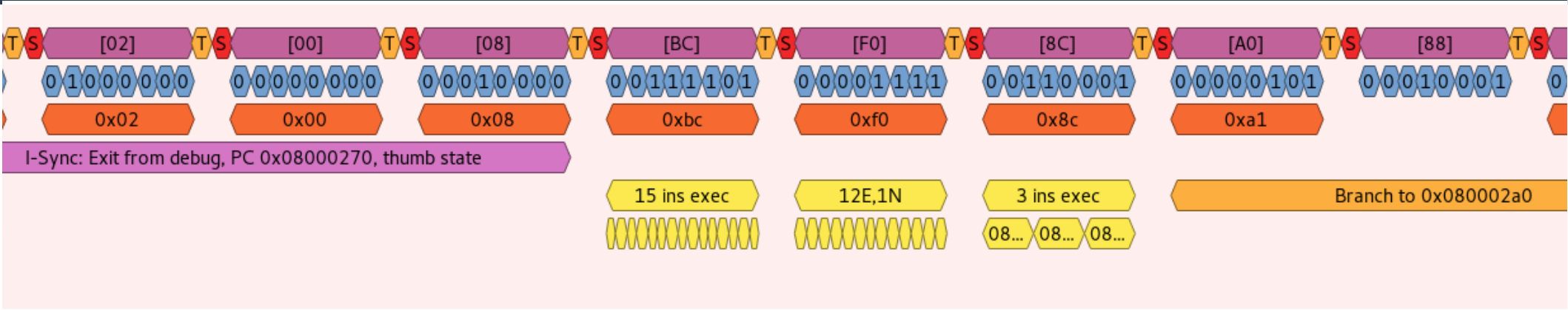


Image Source: STMicroelectronics

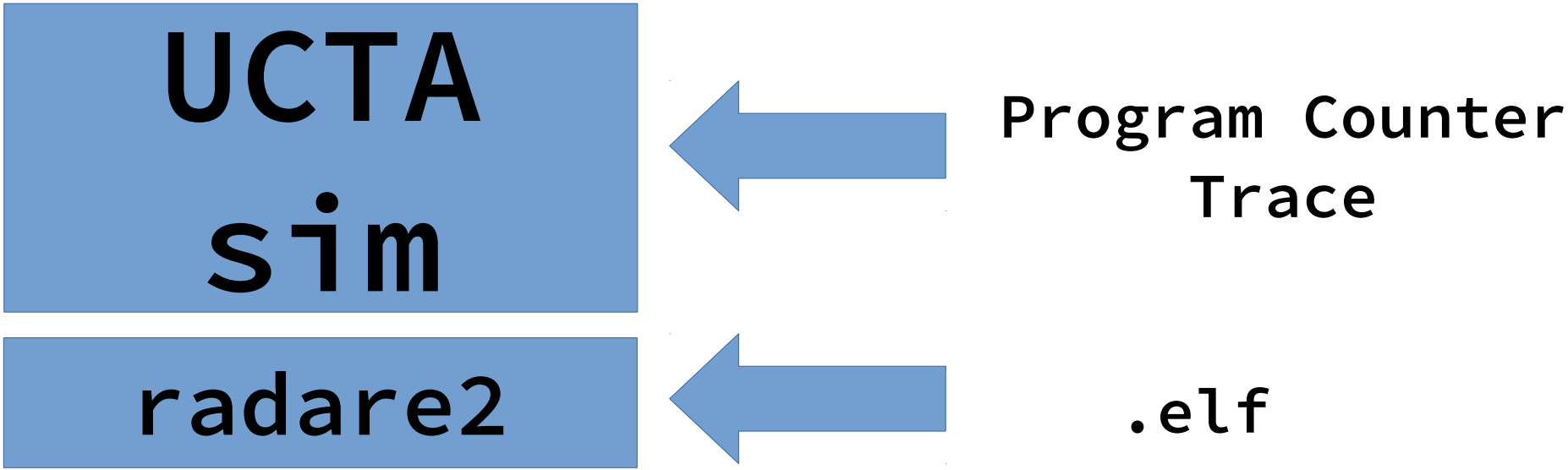
Image Source: seeedstudio.com

Fuzzing Embedded Devices



Instruction trace provides us with all program counter values, but no information about the memory state

Micro Controller Trace Analysis (UCTA)



Example: Stack Overflow

```
void buggy_function(const uint8_t* packet) {
   char buffer[8];
   const uint8_t length = packet[0];
   std::memcpy(buffer, packet + 1, length);
}
```

Example: Stack Overflow

```
void buggy_function(const uint8_t* packet) {
   char buffer[8];
   const uint8_t length = packet[0];
   std::memcpy(buffer, packet + 1, length);
}
```

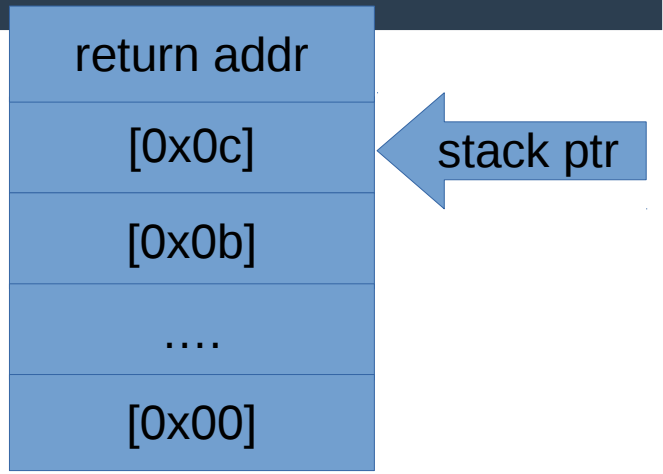
Example: Stack Overflow

```
;-- buggy_function:
0x080001a0      00b5        push {lr}
0x080001a2      83b0        sub sp, 0xc
0x080001a4      0278        ldrb r2, [r0]
0x080001a6      411c        adds r1, r0, 1
0x080001a8      6846        mov r0, sp
0x080001aa      064b        ldr r3, [0x080001c6]
0x080001ac      9847        blx r3
0x080001c0      03b0        add sp, 0xc
0x080001c2      00bd        pop {pc}
```

```
void buggy_function(const uint8_t* packet) {
   char buffer[8];
   const uint8_t length = packet[0];
   std::memcpy(buffer, packet + 1, length);
}
```

Example: Stack Overflow

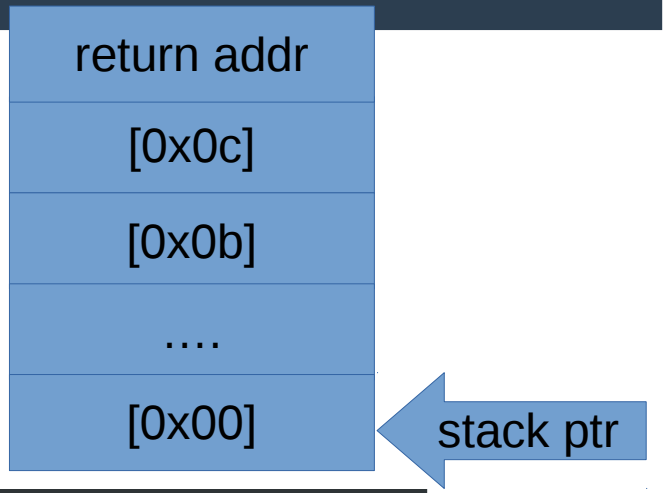
```
;-- buggy_function:
0x080001a0      00b5      push {lr}
0x080001a2      83b0      sub sp, 0xc
0x080001a4      0278      ldrb r2, [r0]
0x080001a6      411c      adds r1, r0, 1
0x080001a8      6846      mov r0, sp
0x080001aa      064b      ldr r3, [0x080001c6]
0x080001ac      9847      blx r3
0x080001c0      03b0      add sp, 0xc
0x080001c2      00bd      pop {pc}
```




```
void buggy_function(const uint8_t* packet) {
   char buffer[8];
   const uint8_t length = packet[0];
   std::memcpy(buffer, packet + 1, length);
}
```

Example: Stack Overflow

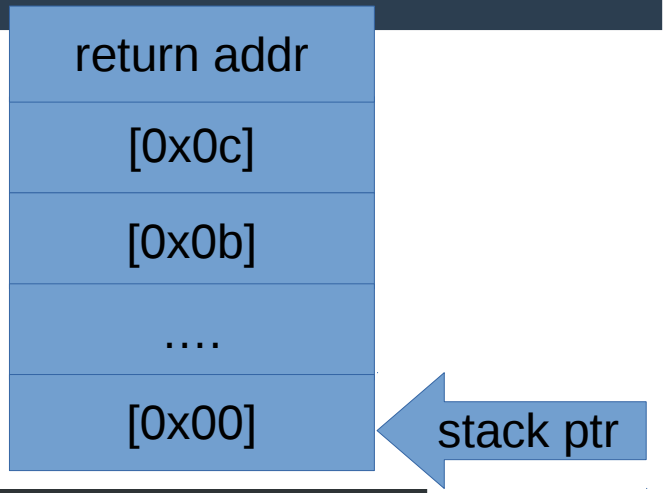
```
;-- buggy_function:
0x080001a0      00b5      push {lr}
0x080001a2      83b0      sub sp, 0xc
0x080001a4      0278      ldrb r2, [r0]
0x080001a6      411c      adds r1, r0, 1
0x080001a8      6846      mov r0, sp
0x080001aa      064b      ldr r3, [0x080001c6]
0x080001ac      9847      blx r3
0x080001c0      03b0      add sp, 0xc
0x080001c2      00bd      pop {pc}
```




```
void buggy_function(const uint8_t* packet) {
   char buffer[8];
   const uint8_t length = packet[0];
   std::memcpy(buffer, packet + 1, length);
}
```

Example: Stack Overflow

```
;-- buggy_function:
0x080001a0      00b5      push {lr}
0x080001a2      83b0      sub sp, 0xc
0x080001a4      0278      ldrb r2, [r0]
0x080001a6      411c      adds r1, r0, 1
0x080001a8      6846      mov r0, sp
0x080001aa      064b      ldr r3, [0x080001c6]
0x080001ac      9847      blx r3
0x080001c0      03b0      add sp, 0xc
0x080001c2      00bd      pop {pc}
```




```
void buggy_function(const uint8_t* packet) {
   char buffer[8];
   const uint8_t length = packet[0];
   std::memcpy(buffer, packet + 1, length);
}
```

Example: Stack Overflow

```
;-- buggy_function:
0x080001a0      00b5      push {lr}
0x080001a2      83b0      sub sp, 0xc
0x080001a4      0278      ldrb r2, [r0]
0x080001a6      411c      adds r1, r0, 1
0x080001a8      6846      mov r0, sp
0x080001aa      064b      ldr r[illegible] [0x080001c6]
0x080001ac      9847      blx r[illegible]
0x080001c0      03b0      add s[illegible], 0xc
0x080001c2      00bd      pop {p[illegible]c}
```

return addr
[0x0c]
[0x0b]
....
[0x00]

stack ptr

```
void buggy_function(const uint8_t* packet) {
   char buffer[8];
   const uint8_t length = packet[0];
   std::memcpy(buffer, packet + 1, length);
}
```

Example: Stack Overflow

```
;-- buggy_function:
0x080001a0      00b5      push {lr}
0x080001a2      83b0      sub sp, 0xc
0x080001a4      0278      ldrb r2, [r0]
0x080001a6      411c      adds r1, r0, 1
0x080001a8      6846      mov r0, sp
0x080001aa      064b      ldr r3, [0x080001c6]
0x080001ac      9847      blx r3
0x080001c0      03b0      add sp, 0xc
0x080001c2      00bd      pop {pc}
```

return addr
[0x0c]
[0x0b]
....
[0x00]

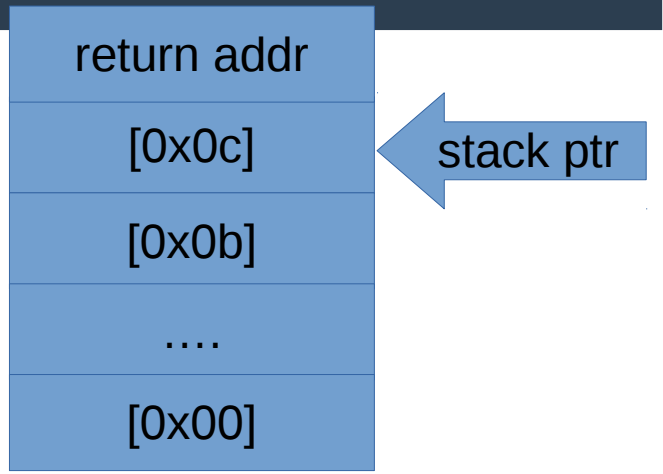
stack ptr

```
void buggy_function(const uint8_t* packet) {
    char buffer[8];
    const uint8_t length = packet[0];
    std::memcpy(buffer, packet + 1, length);
}
```

std::memcpy

Example: Stack Overflow

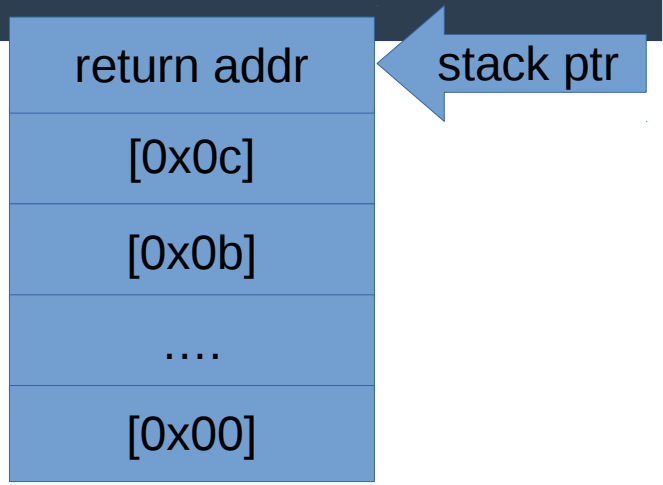
```
;-- buggy_function:
0x080001a0      00b5      push {lr}
0x080001a2      83b0      sub sp, 0xc
0x080001a4      0278      ldrb r2, [r0]
0x080001a6      411c      adds r1, r0, 1
0x080001a8      6846      mov r0, sp
0x080001aa      064b      ldr r3, [0x080001c6]
0x080001ac      9847      blx r3
0x080001c0      03b0      add sp, 0xc
0x080001c2      00bd      pop {pc}
```




```
void buggy_function(const uint8_t* packet) {
  char buffer[8];
  const uint8_t length = packet[0];
  std::memcpy(buffer, packet + 1, length);
}
```

Example: Stack Overflow

```
;-- buggy_function:
0x080001a0      00b5      push {lr}
0x080001a2      83b0      sub sp, 0xc
0x080001a4      0278      ldrb r2, [r0]
0x080001a6      411c      adds r1, r0, 1
0x080001a8      6846      mov r0, sp
0x080001aa      064b      ldr r3, [0x080001c6]
0x080001ac      9847      blx r3
0x080001c0      03b0      add sp, 0xc
0x080001c2      00bd      pop {pc}
```



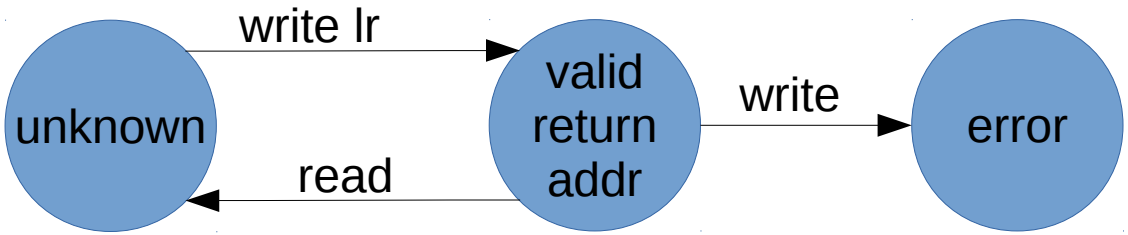

```
void buggy_function(const uint8_t* packet) {
   char buffer[8];
   const uint8_t length = packet[0];
   std::memcpy(buffer, packet + 1, length);
}
```

Example: Stack Overflow

```
return_addr_locs = []

def on_store(addr, value, src_reg, pc, instr_count):
    if addr in return_addr_locs:
        raise Exception("Return address overwriten with 0x{:08x} @ pc=0x{:08x}".format(value, pc))
    elif src_reg == 14:
        return_addr_locs.append(addr)

def on_load(addr, value, dst_reg, pc, instr_count):
    if addr in return_addr_locs:
        return_addr_locs.remove(addr)
```



Demo Time:

```
$ wc -l pc
$ ./sim.sh
```

Outlook

- Scale up: add support for more instructions, interrupts and fix bugs
- Automate trace analysis: this is required for automated fuzzing
- Collect branch metrics
- Implement more error detectors: stack overflow detector, detect out of bounds accesses, etc.