JULY 9–13, 2023

MOSCONE WEST CENTER

SAN FRANCISCO, CA, USA

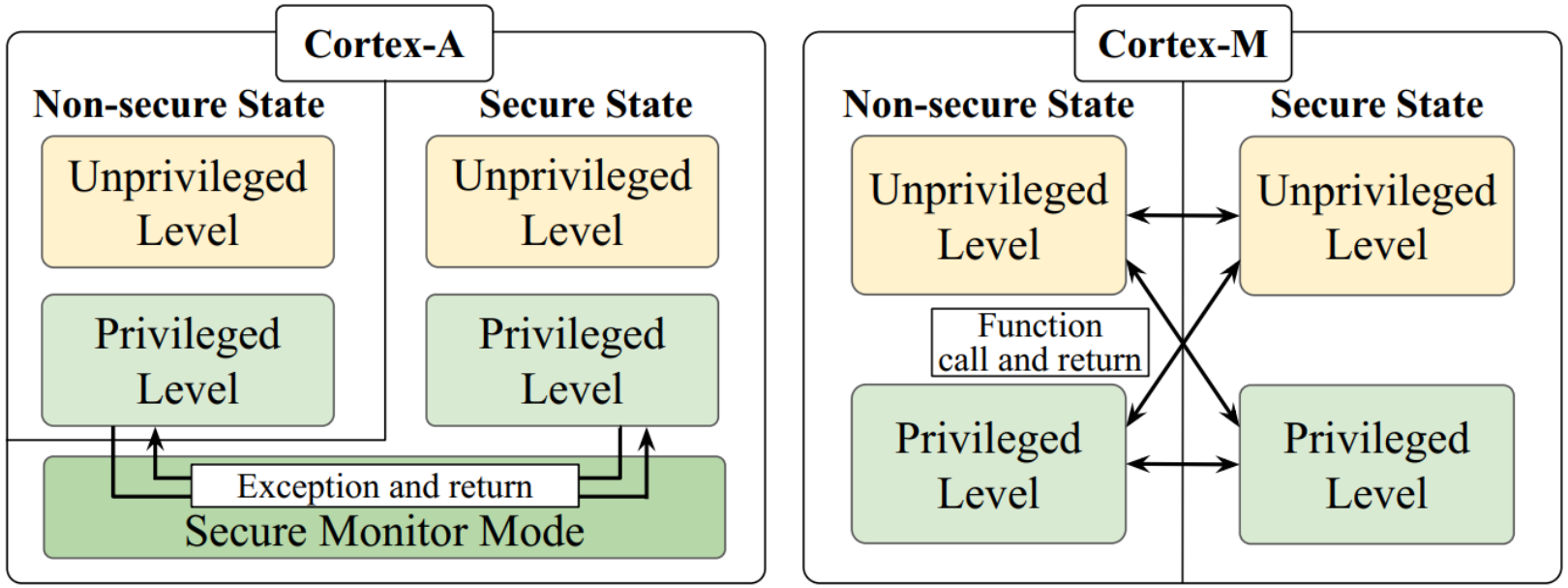
University at Buffalo
The State University of New York

Return-to-Non-Secure Vulnerabilities on ARM Cortex-M TrustZone: Attack and Defense

Zheyuan Ma†‡, Xi Tan†‡, Lukasz Ziarek†, Ning Zhang*, Hongxin Hu†, Ziming Zhao†‡
†University at Buffalo ‡CactiLab *Washington University in St. Louis
{zheyuanm, xitan, lziarek, hongxinh, zimingzh}@buffalo.edu, zhang.ning@wustl.edu

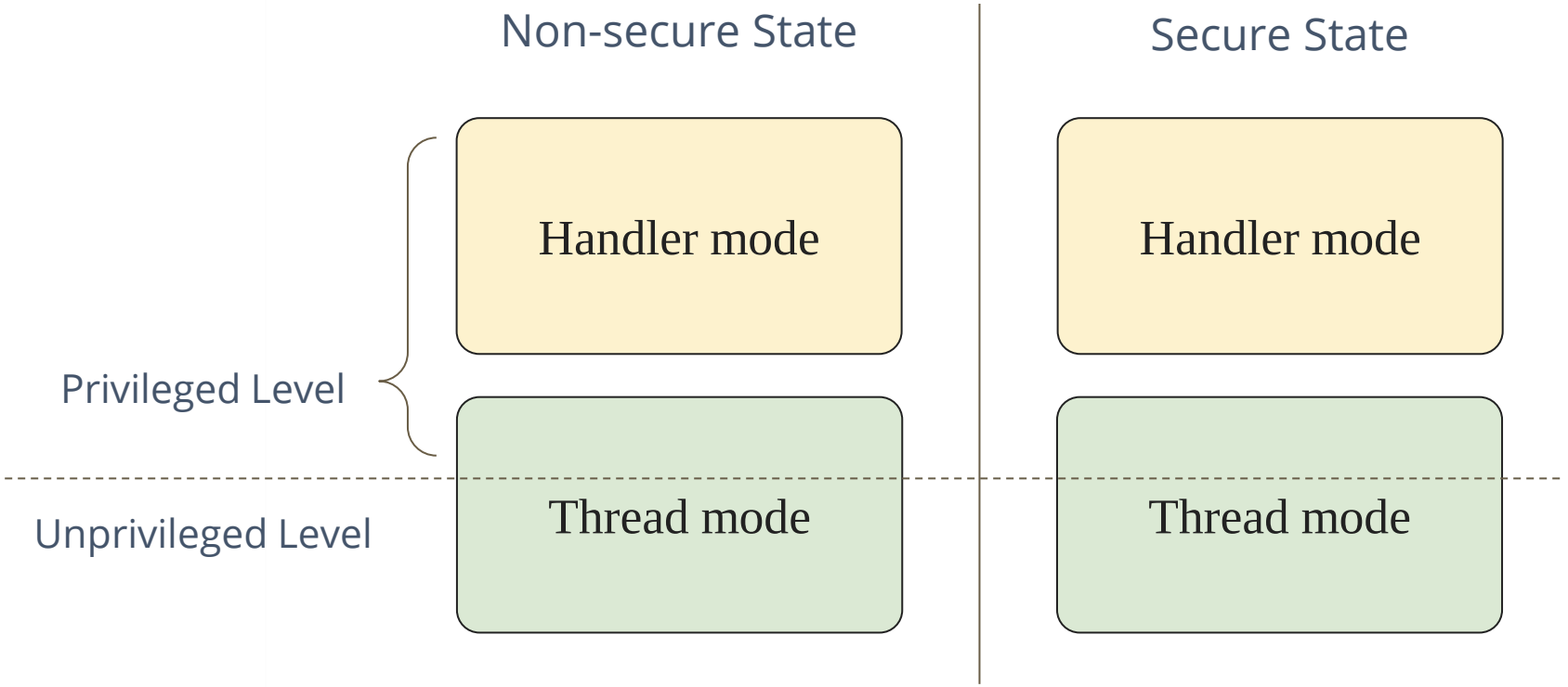
Background

- TrustZone on Cortex-A vs Cortex-M
  - Cortex-M's rapid state switch has security implications
  - The semantic gap resulting in potential confused-deputy attacks



S: Secure State
NS: Non-secure State

Background



`IPSR != 0`
Handler mode

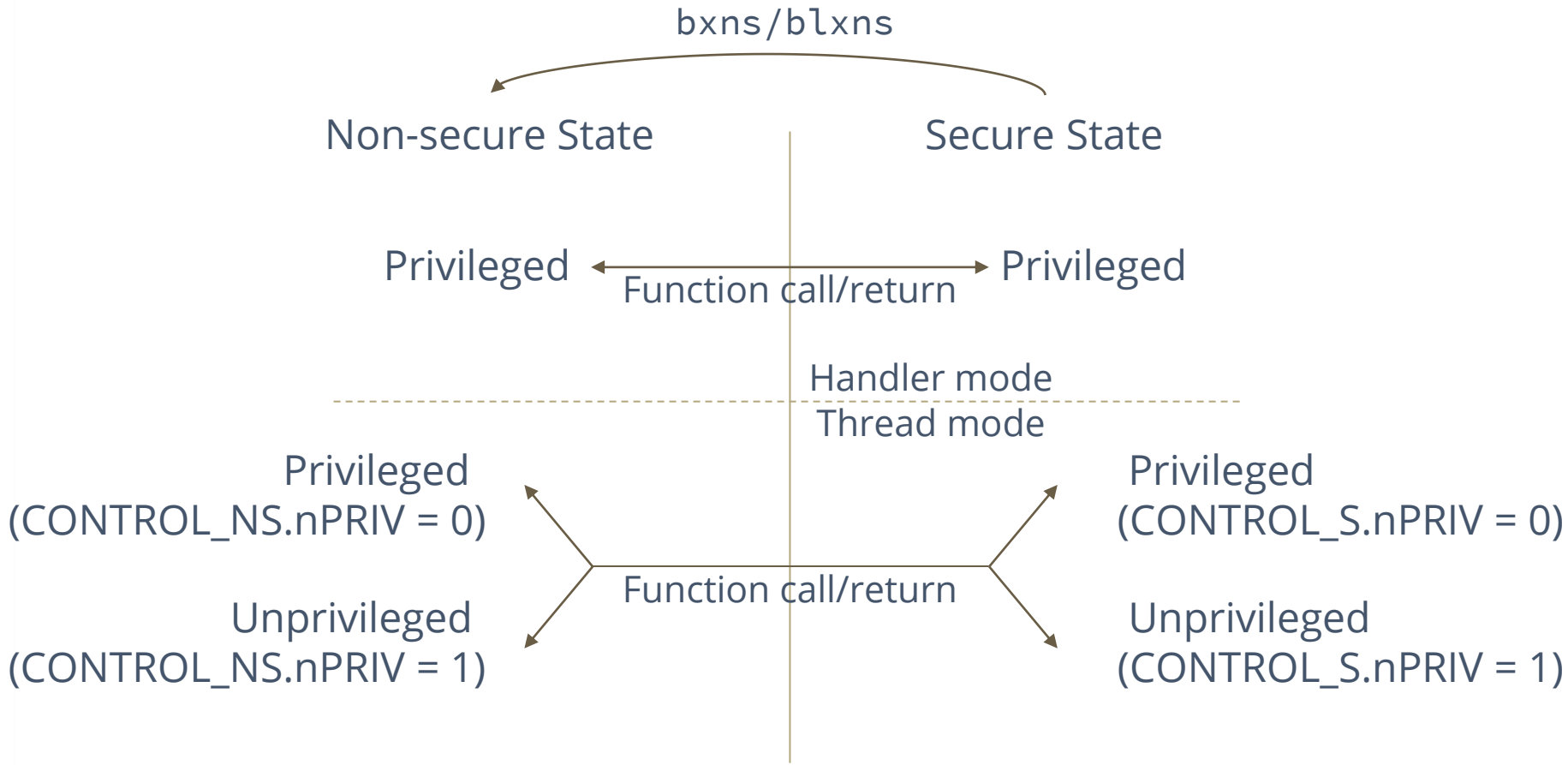
`IPSR = 0`
Thread mode

`CONTROL.nPRIV = 0`
Privileged Level

`CONTROL.nPRIV = 1`
Unprivileged Level

`IPSR` register is shared between states; `CONTROL` register is banked for each state.

Background



`IPSR` register is shared between states; `CONTROL` register is banked for each state.

Related Work

- Ret2user, ret2dir, and boomerang attacks
  - Targeting microprocessors with MMUs and modern OSes like Linux
- Secure communication mechanisms on Cortex-M TrustZone
  - Our attack does not rely on tempering cross-state messages

*Kemerlis, Vasileios P., Georgios Portokalidis, and Angelos D. Keromytis. "kGuard: Lightweight Kernel Protection against **Return-to-User** Attacks." USENIX Security, 2012.*
*Kemerlis, Vasileios P., Michalis Polychronakis, and Angelos D. Keromytis. "**ret2dir**: Rethinking kernel isolation." USENIX Security, 2014.*
*Machiry, Aravind, et al. "**BOOMERANG**: Exploiting the Semantic Gap in Trusted Execution Environments." NDSS, 2017.*
Iannillo, Antonio Ken, et al. "An REE-independent Approach to Identify Callers of TEEs in TrustZone-enabled Cortex-M Devices." CPSS@AsiaCCS, 2022.
Khurshid, Anum, et al. "ShieLD: Shielding Cross-Zone Communication Within Limited-Resourced IoT Devices Running Vulnerable Software Stack." IEEE TDSC, 2022.

The ret2ns Attacks – Threat Model

Goal: user-space attacker in NS conducts privilege escalation

Assumptions:

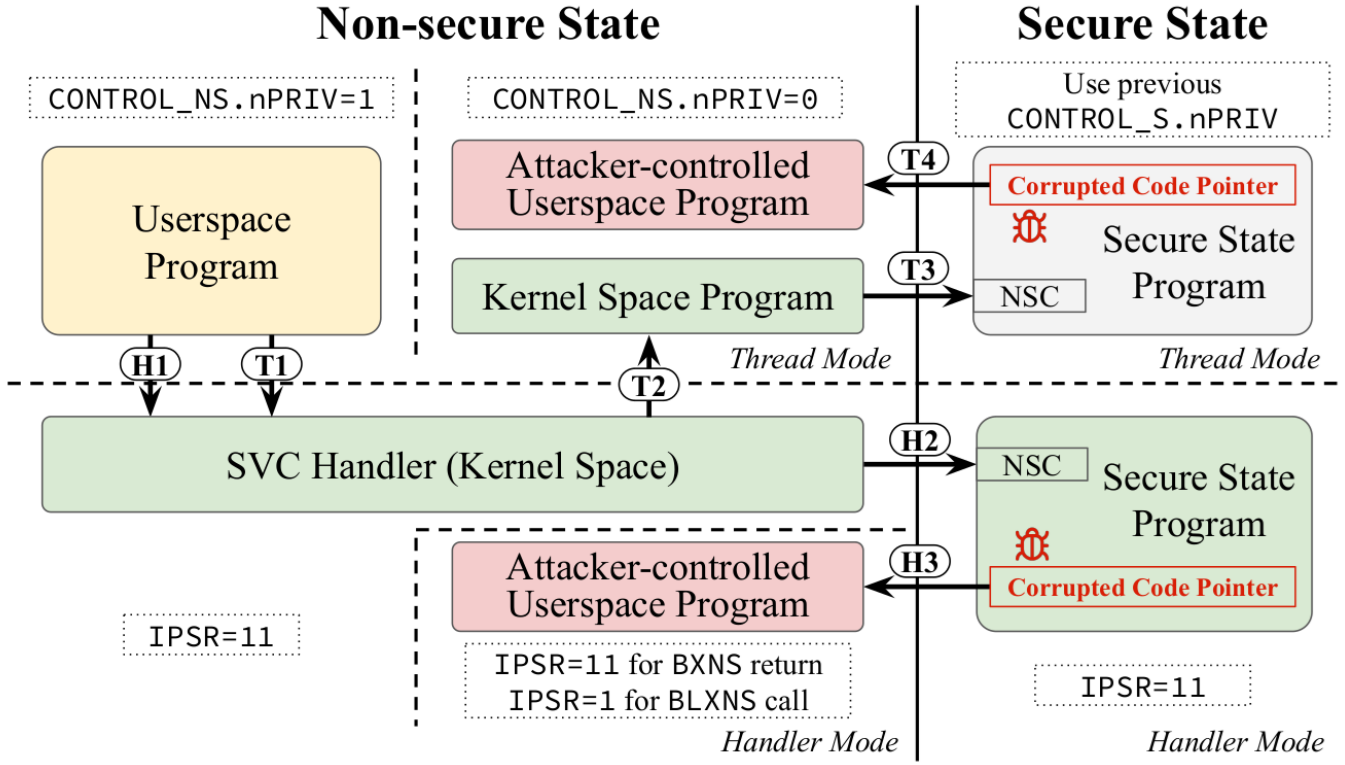
- memory corruption vulnerability in S
- attacker utilizes NS system calls (SVC) for S interaction
- arbitrary code execution in S not possible for attacker

Target: to corrupt code pointer used by bxns/blxns in S

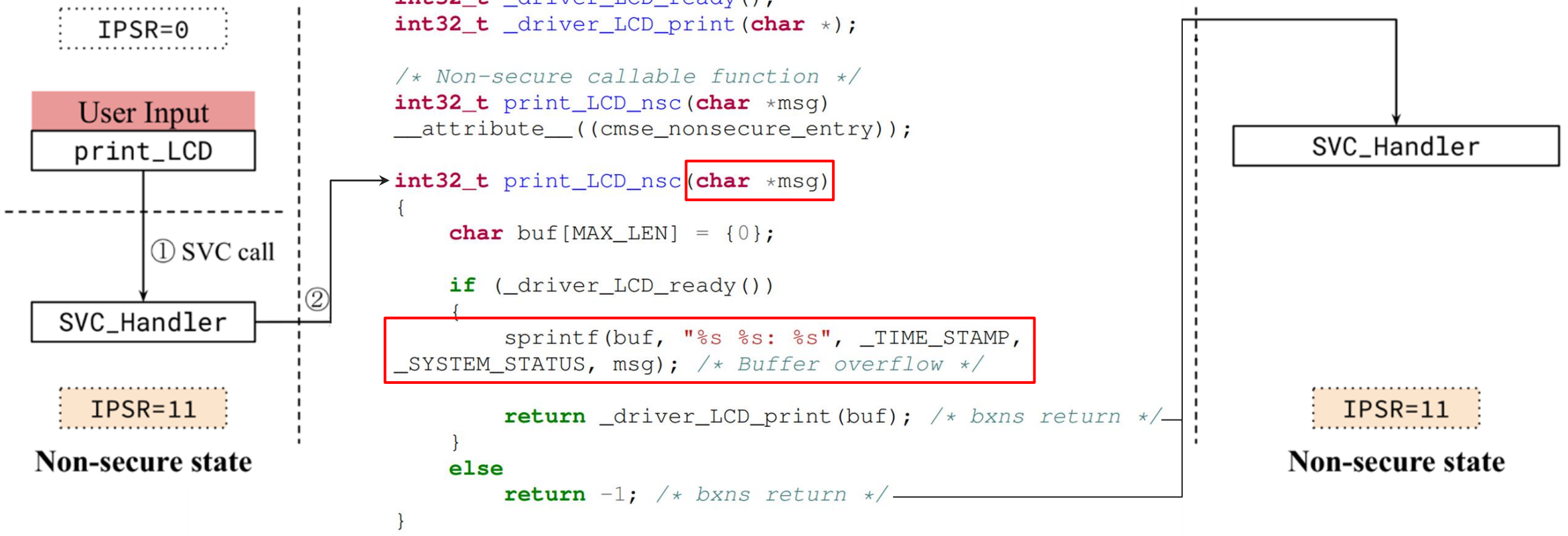
The ret2ns Attacks – Overview

Handler-mode-originated attacks
- IPSR is shared between states

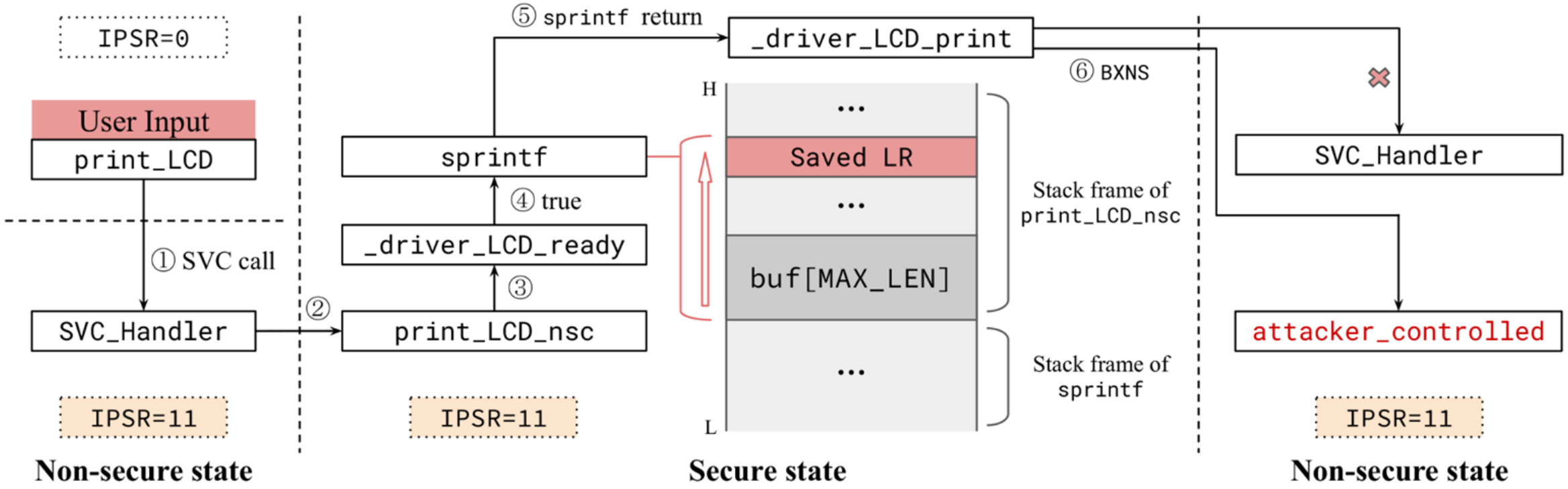
Thread-mode-originated attacks
- CONTROL.nPRIV is banked between states



The ret2ns Attacks – A Walking Example



The ret2ns Attacks – A Walking Example



Defense 1

MPU-assisted Address Sanitizer

- Validate memory access permissions for NS target
- Verify NS destination address against NS MPU configuration before bxns/blxns branches

```
1   ldr   R0, [SP, #12]   ; only need for bxns
2   mrs   R3, IPSR        ; read IPSR value
3   cbnz  R3, #6          ; check if IPSR is zero
4   mrs   R3, CONTROL_NS  ; read CONTROL_NS
5   lsls  R3, R3, #31     ; get CONTROL_NS.nPRIV bit
6   bne   #28             ; check if nPRIV bit zero
7   tta   R0, R0          ; test target on target addr
8   lsls  R3, R0, #15     ; get MRVALID bit
9   bpl   #20             ; check if region is valid
10  uxtb  R0, R0          ; get MPU region number
11  movw  R3, #0xED98     ; load MPU_NS->RNR addr
12  movt  R3, #0xE002     ; 0xE002ED98
13  str   R0, [R3, #0]    ; set MPU_NS->RNR to region
14  ldr   R0, [R3, #4]    ; get MPU_NS->RBAR
15  lsls  R0, R0, #30     ; get UP read permission bit
16  beq   #2              ; check UP read permission
17  cpsid i               ; disable IRQ
18  b     .               ; error handling
```

Defense 2

Address Masking

- Assume user/kernel space programs in distinct, known memory regions
- Apply bit-wise masking to NS target address

```
1   ldr     R1, [SP, #4]    ; get return addr (not for blxns)
2   mrs     R2, IPSR        ; read IPSR
3   cbnz    R2, #6          ; check if IPSR is zero
4   mrs     R2, control_ns  ; read CONTROL_NS
5   lsls    R2, R2, #31     ; get CONTROL_NS.nPRIV bit
6   bne     #8              ; check nPRIV bit
7   cmn     R1, #0x100      ; Is EXC_RETURN? (not for blxns)
8   it      cc              ; not EXC_RETURN (not for blxns)
9   movtcc  R1, #0x21       ; address masking
10  strcc   R1, [SP, #4]    ; store result (not for blxns)
```

Defense Evaluation

A modified Blinky application with S and NS states

- Higher T: less frequent routine communications between states
- Higher N: less frequent service requests from NS to S

T, N	Blinky	MPU-assisted Addr Sanitizer	Address Masking
10^7, 10^7	1,200,503,441	1,200,508,359 (0.0004%)	1,200,506,190 (0.0002%)
10^5, 10^5	12,503,869	12,508,385 (0.0361%)	12,506,793 (0.0234%)
10^7, 10^5	12,473,897	12,474,465 (0.0046%)	12,474,243 (0.0028%)
10^5, 10^7	1,203,433,289	1,203,892,073 (0.0381%)	1,203,674,733 (0.0201%)

The Blinky application is a cross-world project with both non-secure and secure state programs, and it works on a system with 3 LEDs and a UART peripheral. T controls the triggering frequency of the SysTick timer, and the N defines the number of nop instructions before each NSC function call.

Q&A

Open-sourced at: https://github.com/CactiLab/ret2ns-Cortex-M-TrustZone

Zheyuan Ma • CactiLab

zheyuanm@buffalo.edu