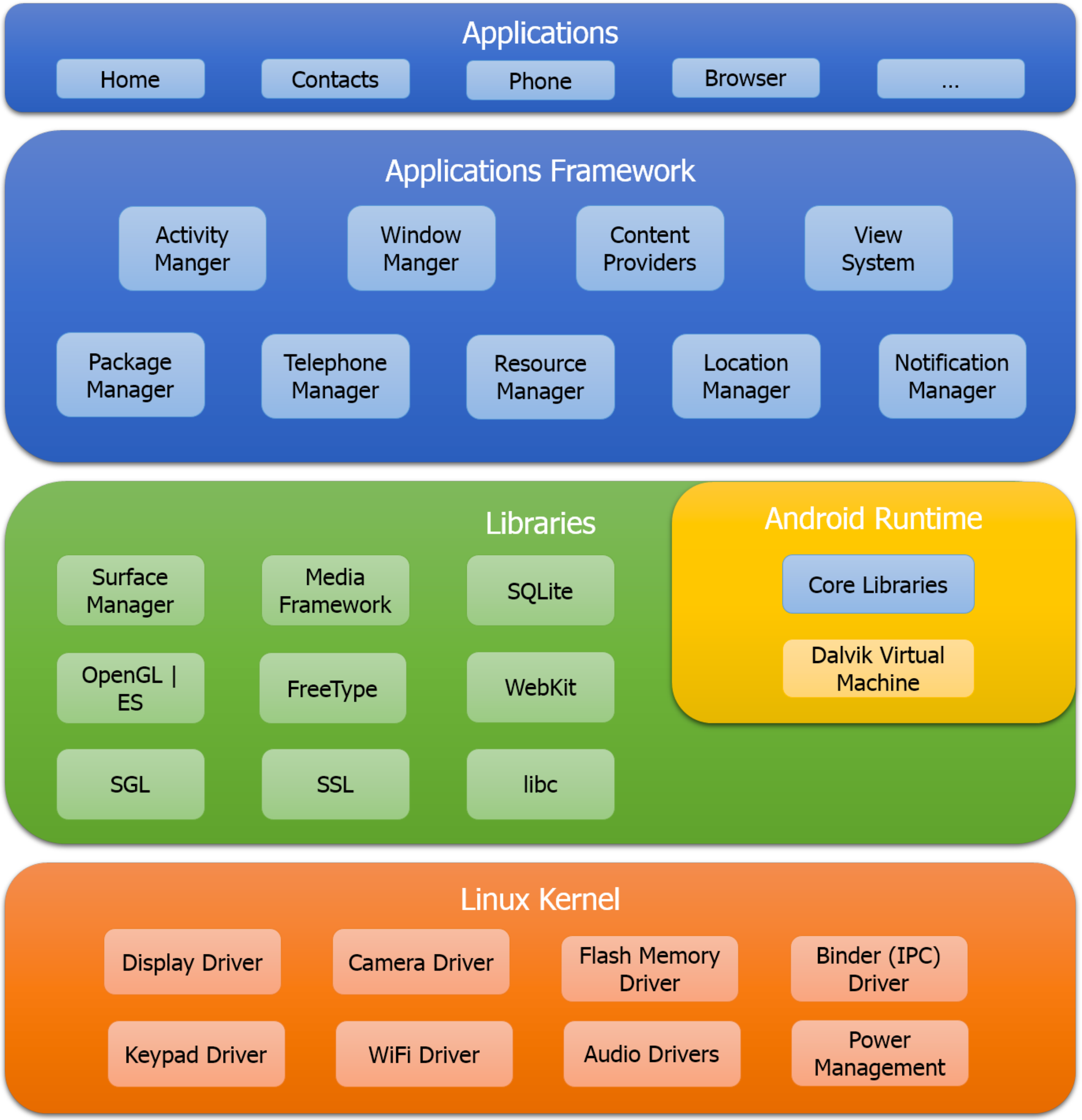
SPROBE: Enforcing Kernel Code Integrity on the TrustZone Architecture

Xinyang Ge, Hayawardh Vijayakumar, and Trent Jaeger

May 17th, 2014

Problem


Applications
Home
Contacts
Phone
Browser
...
Applications Framework
Activity Manger
Window Manger
Content Providers
View System
Package Manager
Telephone Manager
Resource Manager
Location Manager
Notification Manager
Libraries
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
libc
Android Runtime
Core Libraries
Dalvik Virtual Machine
Linux Kernel
Display Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

Problem

PENNSTATE

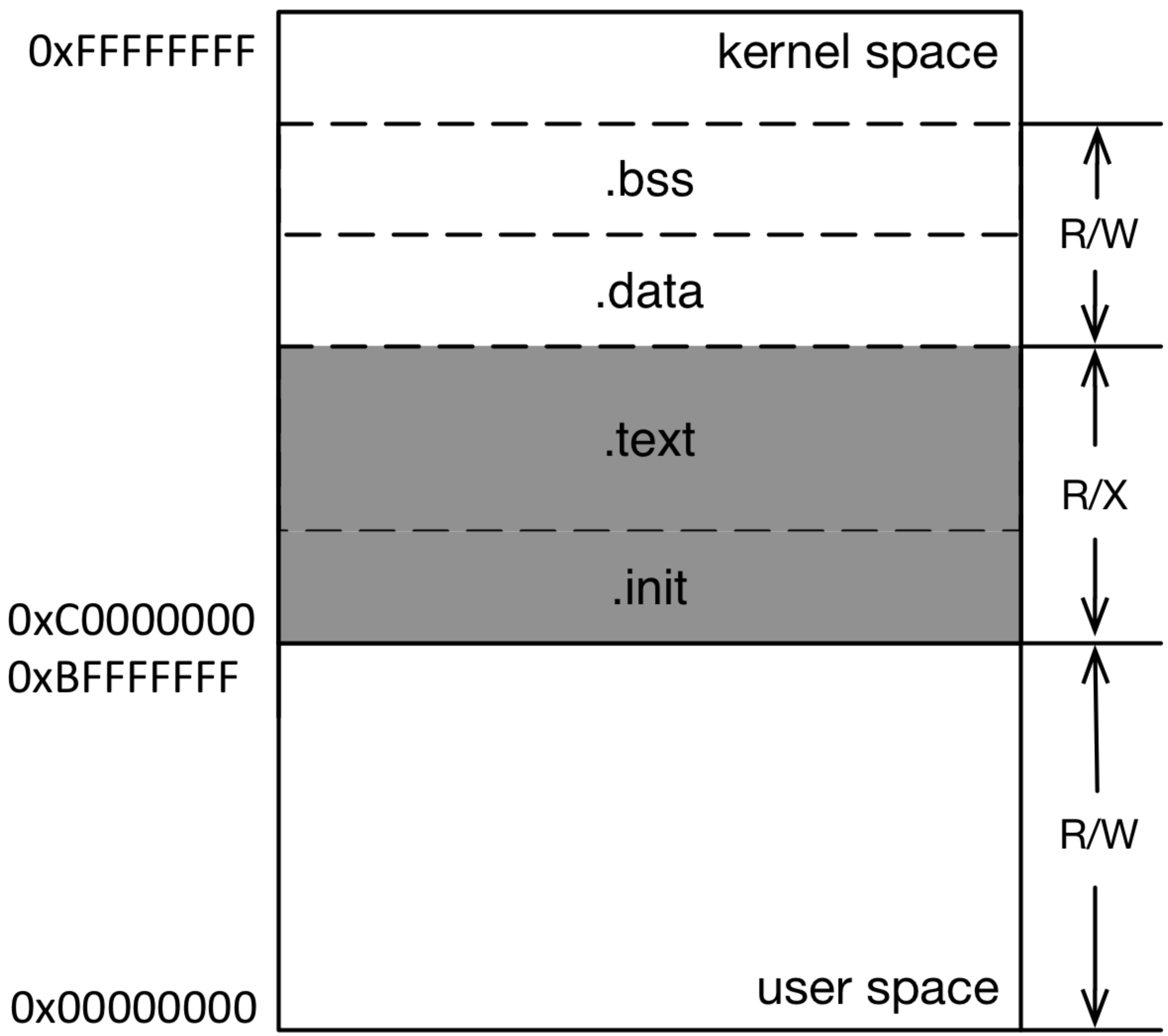
1855

- Kernel rootkits are now becoming a serious threat to smartphone operating systems (e.g., Android)
  - CVE-2011-1823: an integer overflow bug in a daemon process on Android 3.0 enables an adversary to gain root privilege and install a kernel rootkit
  - Motivation: Protect the kernel code integrity despite of the presence of kernel rootkits

Preliminary Defenses

- W⊕X Protection
  - Background: eXecute-Never (XN) bit
  - A virtual memory page cannot be set as writable and executable at the same time
- Privileged eXecute-Never
  - Prohibit user code from executing in the kernel

Preliminary Defenses



Possible Attacks

PENNSTATE 1855

- Code-Reuse Attacks
  - ret2libc attack
  - return-oriented programming
- Attack Vectors
  1. Attacks that modify the initial VM layout
  2. Attacks that remain the initial VM layout

Type #1 Attacks

- Goal: Enable write over code or execute over data

Type #1 Attacks

- Goal: Enable write over code or execute over data
  - Disable the W$\oplus$X protection

Type #1 Attacks

- Goal: Enable write over code or execute over data
  - Disable the W⊕X protection
  - Change to a different set of page tables that are under attacker's control

Type #1 Attacks

- Goal: Enable write over code or execute over data
  - Disable the W⊕X protection
  - Change to a different set of page tables that are under attacker's control
  - Modify page table entries in place

Type #1 Attacks

- Goal: Enable write over code or execute over data
  - Disable the W⊕X protection
  - Change to a different set of page tables that are under attacker's control
  - Modify page table entries in place
  - Enable execution over code pages in the user space

Type #1 Attacks

- Goal: Enable write over code or execute over data
  - Disable the W⊕X protection
  - Change to a different set of page tables that are under attacker's control
  - Modify page table entries in place
  - Enable execution over code pages in the user space
  - Disable the MMU

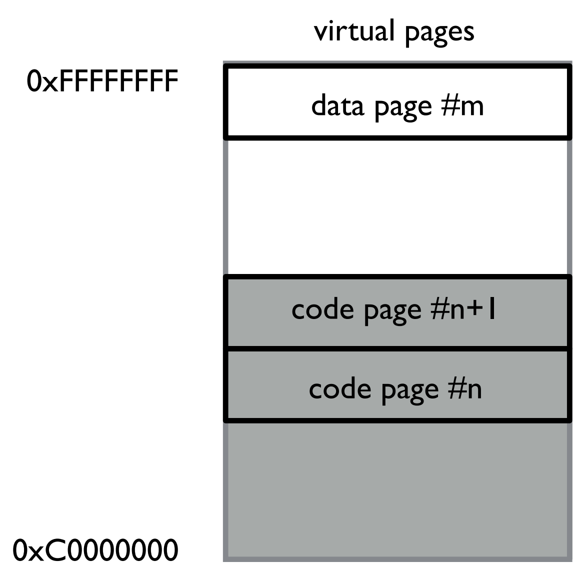
Type #2 Attacks

PENNSTATE 1855

- Goal: (1) Remap a code page to a physical frame containing data or (2) map a data page to a physical frame containing code

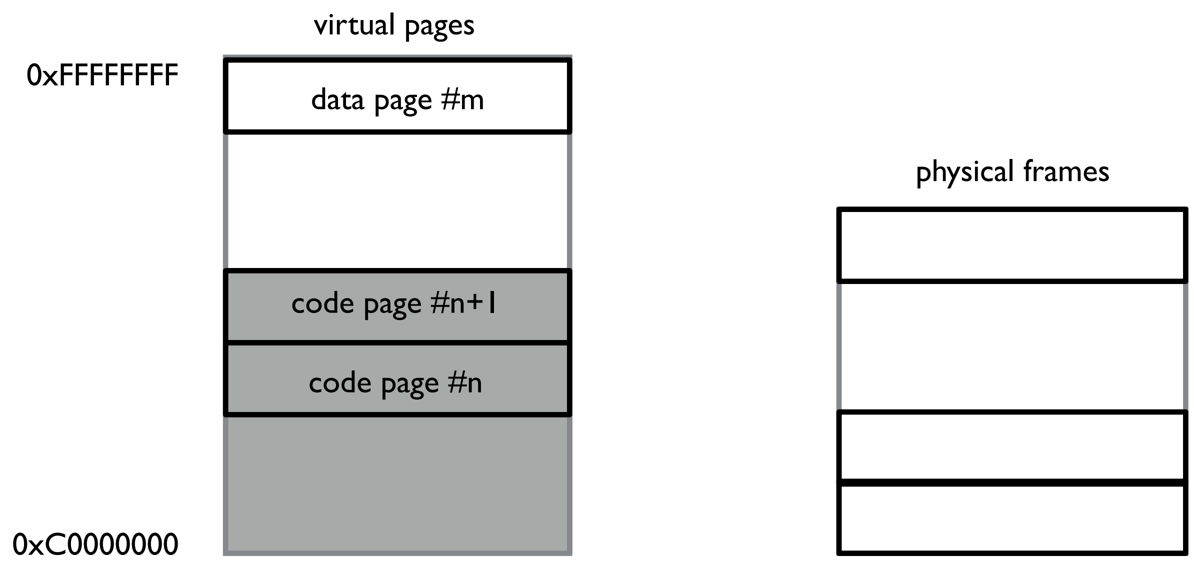
Type #2 Attacks

- Goal: (1) Remap a code page to a physical frame containing data or (2) map a data page to a physical frame containing code



Type #2 Attacks

- Goal: (1) Remap a code page to a physical frame containing data or (2) map a data page to a physical frame containing code

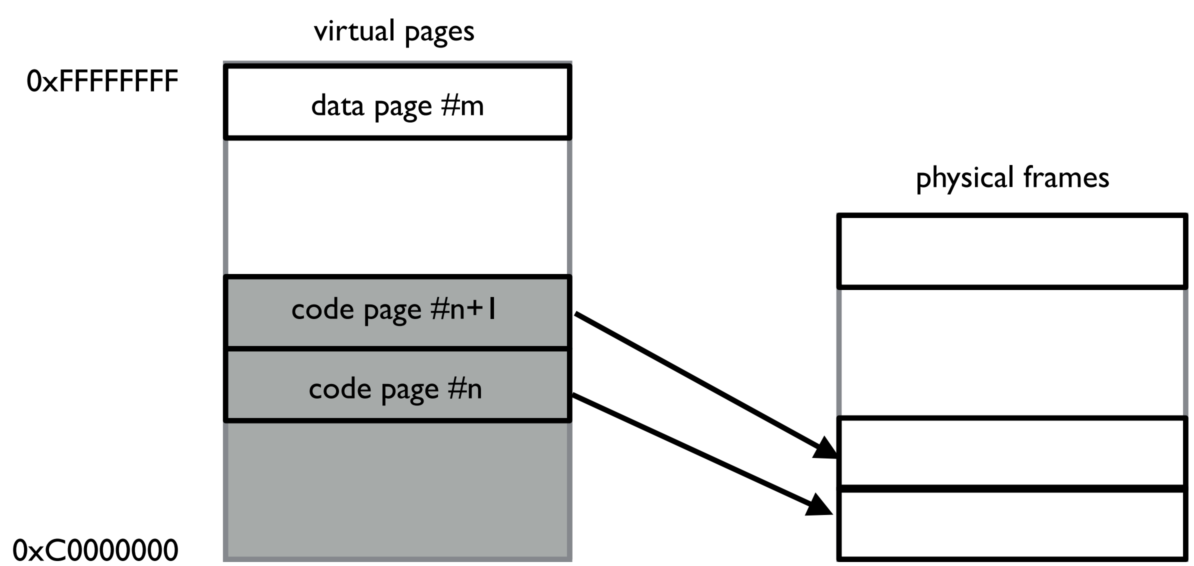


Type #2 Attacks

PENNSTATE

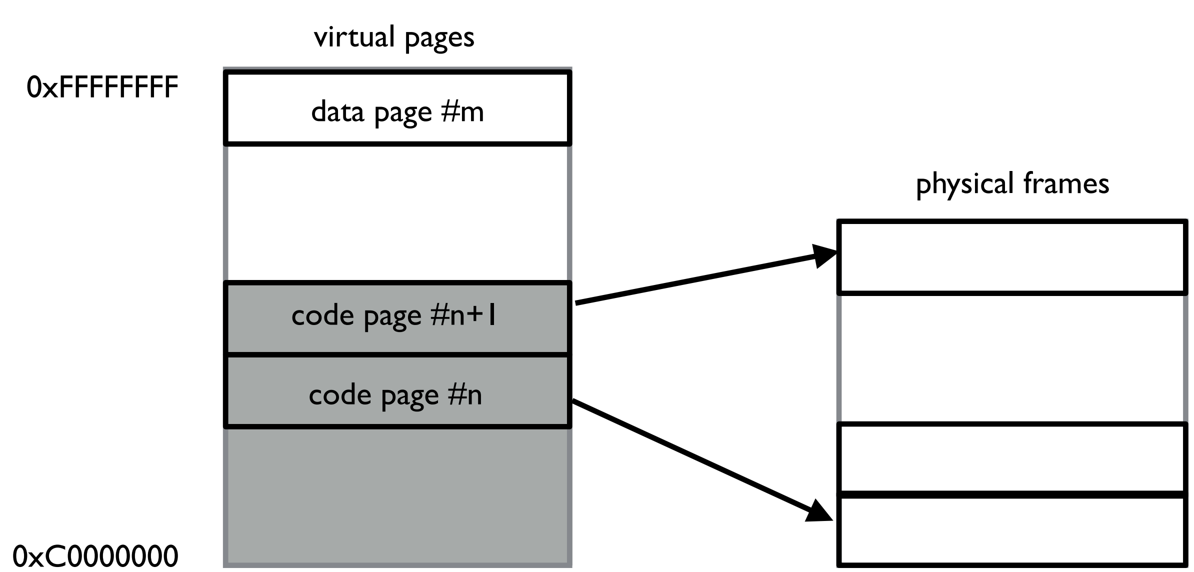
1855

- Goal: (1) Remap a code page to a physical frame containing data or (2) map a data page to a physical frame containing code



Type #2 Attacks

- Goal: (1) Remap a code page to a physical frame containing data or (2) map a data page to a physical frame containing code



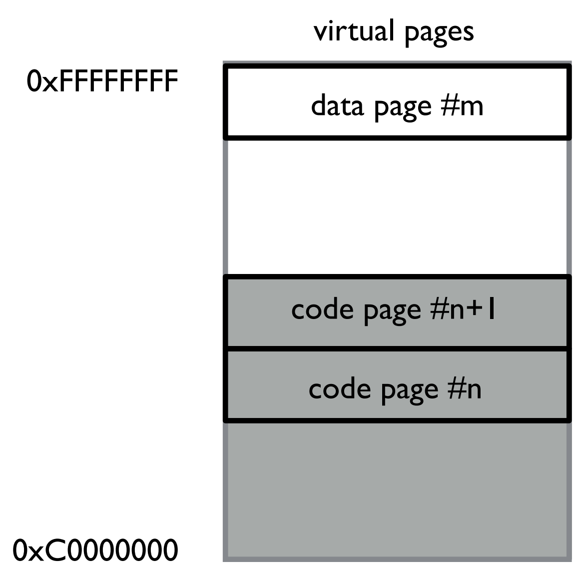
Type #2 Attacks

PENNSTATE 1855

- Goal: (1) Remap a code page to a physical frame containing data or (2) map a data page to a physical frame containing code

Type #2 Attacks

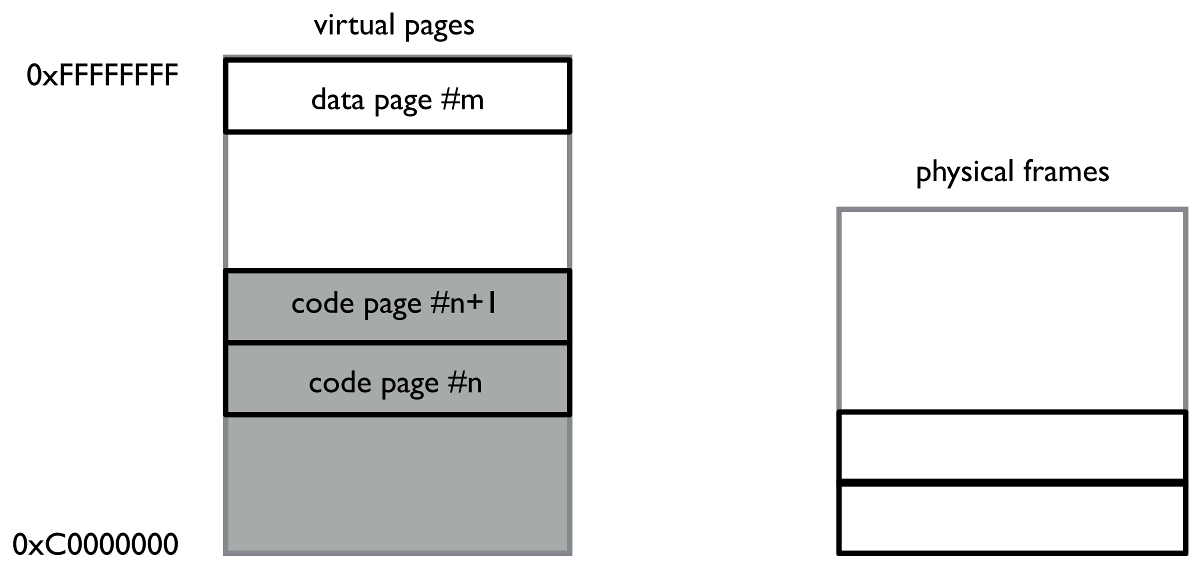
- Goal: (1) Remap a code page to a physical frame containing data or (2) map a data page to a physical frame containing code



Type #2 Attacks

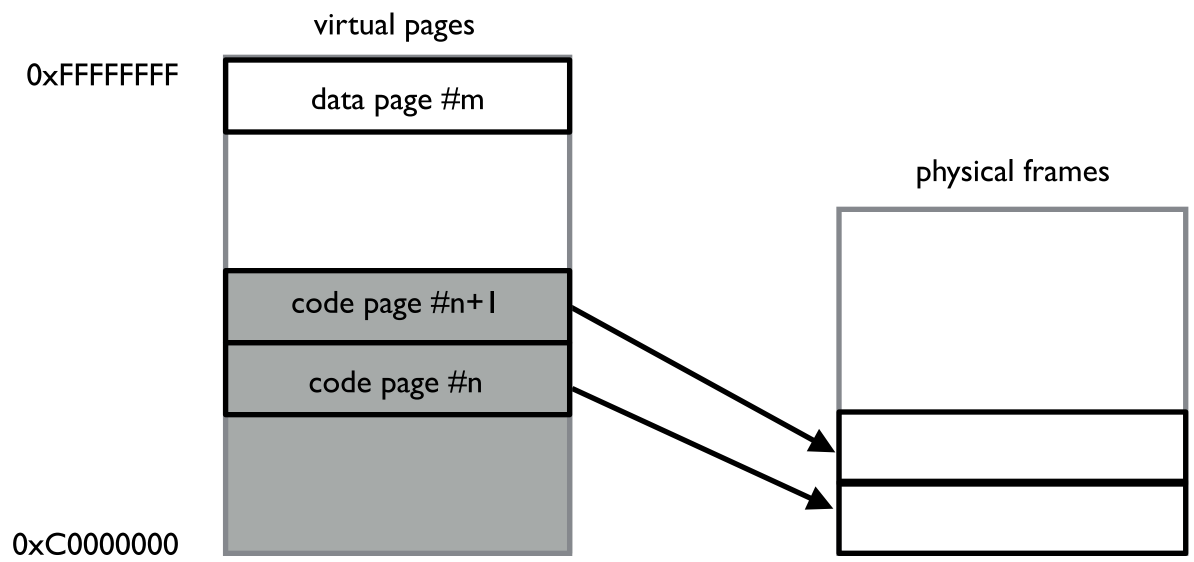
PENNSTATE 1855

- Goal: (1) Remap a code page to a physical frame containing data or (2) map a data page to a physical frame containing code



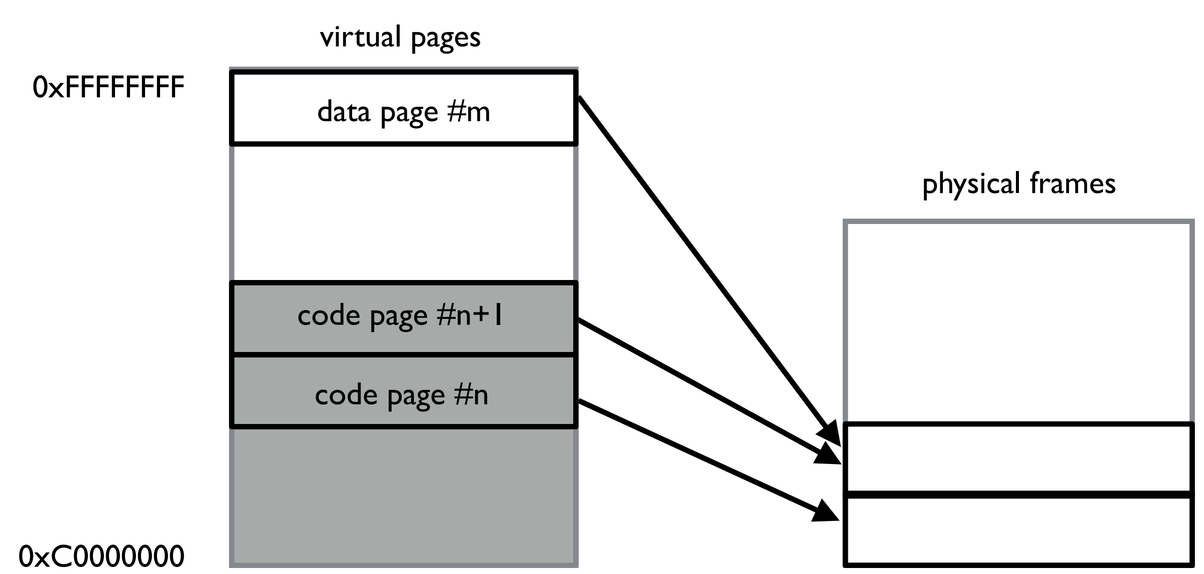
Type #2 Attacks

- Goal: (1) Remap a code page to a physical frame containing data or (2) map a data page to a physical frame containing code



Type #2 Attacks

- Goal: (1) Remap a code page to a physical frame containing data or (2) map a data page to a physical frame containing code



Type #2 Attacks

- Goal: (1) Remap a code page to a physical frame containing data or (2) map a data page to a physical frame containing code

Type #2 Attacks

- Goal: (1) Remap a code page to a physical frame containing data or (2) map a data page to a physical frame containing code
  - Change to a different set of page tables that are under attacker's control

Type #2 Attacks

- Goal: (1) Remap a code page to a physical frame containing data or (2) map a data page to a physical frame containing code
  - Change to a different set of page tables that are under attacker's control
  - Modify page table entries in place

Type #2 Attacks

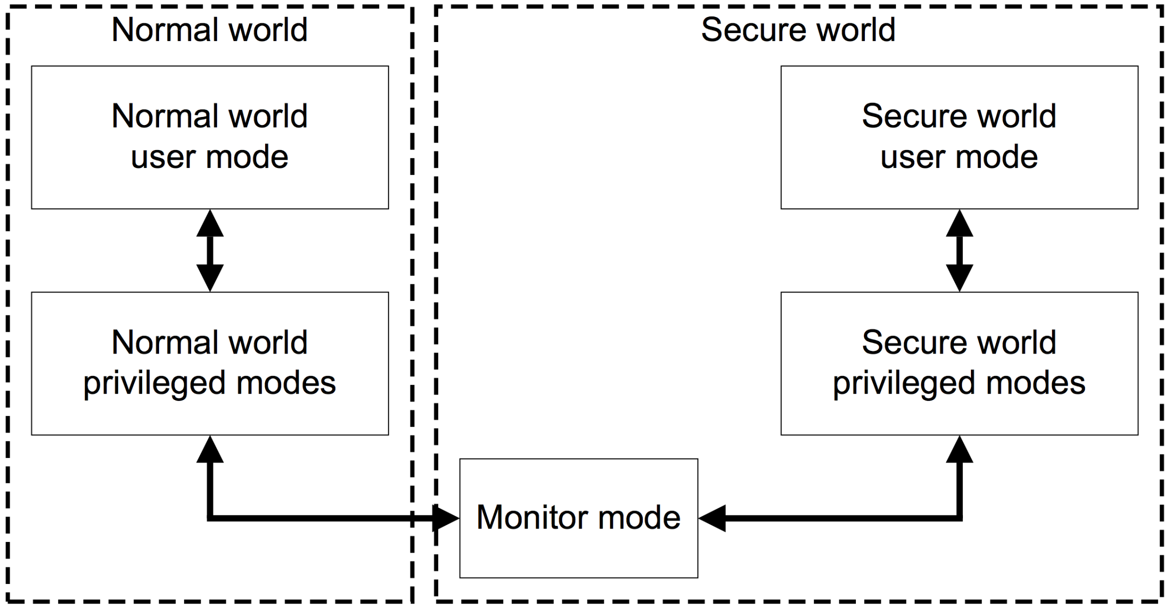
- Goal: (1) Remap a code page to a physical frame containing data or (2) map a data page to a physical frame containing code
  - Change to a different set of page tables that are under attacker's control
  - Modify page table entries in place

Goal

Prevent both types of attacks and limit the adversary to approved kernel code on the TrustZone

Background: TrustZone

- Resources are partitioned into two distinct worlds
  - physical memory, interrupts, peripherals, etc.
- Each world has its own autonomy over its own resources
- Secure world can access normal world resources, but *not* vice versa
- Run in time-sliced fashion



SPROBE Mechanism

- We need an instrument mechanism that enables the secure world to be notified upon events of its choice in the normal world

SPROBE Mechanism

- We need an instrument mechanism that enables the secure world to be notified upon events of its choice in the normal world

Normal World

```
push    {r1-r3}
stmia   sp!, r10
…
mov     pc, lr
```

SPROBE Mechanism

- We need an instrument mechanism that enables the secure world to be notified upon events of its choice in the normal world

Normal World

```
push    {r1-r3}
stmia   sp!, r10
…
smc     #0
```

SPROBE Mechanism

- We need an instrument mechanism that enables the secure world to be notified upon events of its choice in the normal world

Normal World

```
push    {r1-r3}
stmia   sp!, r10
…
smc     #0
```

SPROBE Mechanism

- We need an instrument mechanism that enables the secure world to be notified upon events of its choice in the normal world

Normal World

```
push    {r1-r3}
stmia   sp!, r10
...
smc     #0
```

SPROBE Mechanism

- We need an instrument mechanism that enables the secure world to be notified upon events of its choice in the normal world

Normal World

```
push     {r1-r3}
stmia    sp!, r10
...
smc      #0
```

SPROBE Mechanism

- We need an instrument mechanism that enables the secure world to be notified upon events of its choice in the normal world

Normal World

```
push     {r1-r3}
stmia    sp!, r10
…
smc      #0
```

Secure World

```
sprobe_handler()
{
  check_kernel();
  restore_instn();
  return_to_ns();
}
```

SPROBE Mechanism

- We need an instrument mechanism that enables the secure world to be notified upon events of its choice in the normal world

Normal World

```
push    {r1-r3}
stmia   sp!, r10
…
smc     #0
```

Secure World

```
sprobe_handler()
{
  check_kernel();
  restore_instn();
  return_to_ns();
}
```

SPROBE Mechanism

PENNSTATE

- We need an instrument mechanism that enables the secure world to be notified upon events of its choice in the normal world

Normal World

```
push     {r1-r3}
stmia    sp!, r10
…
smc      #0
```

Secure World

```
sprobe_handler()
{
  check_kernel();
  restore_instn();
  return_to_ns();
}
```

SPROBE Mechanism

- We need an instrument mechanism that enables the secure world to be notified upon events of its choice in the normal world

Normal World

```
push    {r1-r3}
stmia   sp!, r10
…
mov     pc, lr
```

Secure World

```
sprobe_handler()
{
  check_kernel();
  restore_instn();
  return_to_ns();
}
```

SPROBE Mechanism

- We need an instrument mechanism that enables the secure world to be notified upon events of its choice in the normal world

Normal World

```
push     {r1-r3}
stmia    sp!, r10
…
mov      pc, lr
```

Secure World

```
sprobe_handler()
{
  check_kernel();
  restore_instn();
  return_to_ns();
}
```

SPROBE Mechanism

- We need an instrument mechanism that enables the secure world to be notified upon events of its choice in the normal world

Normal World

```
push     {r1-r3}
stmia    sp!, r10
…
mov      pc, lr
```

Secure World

```
sprobe_handler()
{
  check_kernel();
  restore_instn();
  return_to_ns();
}
```

SPROBE Placement

- Recall the specific attacks

SPROBE Placement

- Recall the specific attacks
  - Change to a different set of page tables that are under attacker's control

SPROBE Placement

- Recall the specific attacks
  - Change to a different set of page tables that are under attacker's control
    - instrument all instructions that can be potentially used to switch the page table root

SPROBE Placement

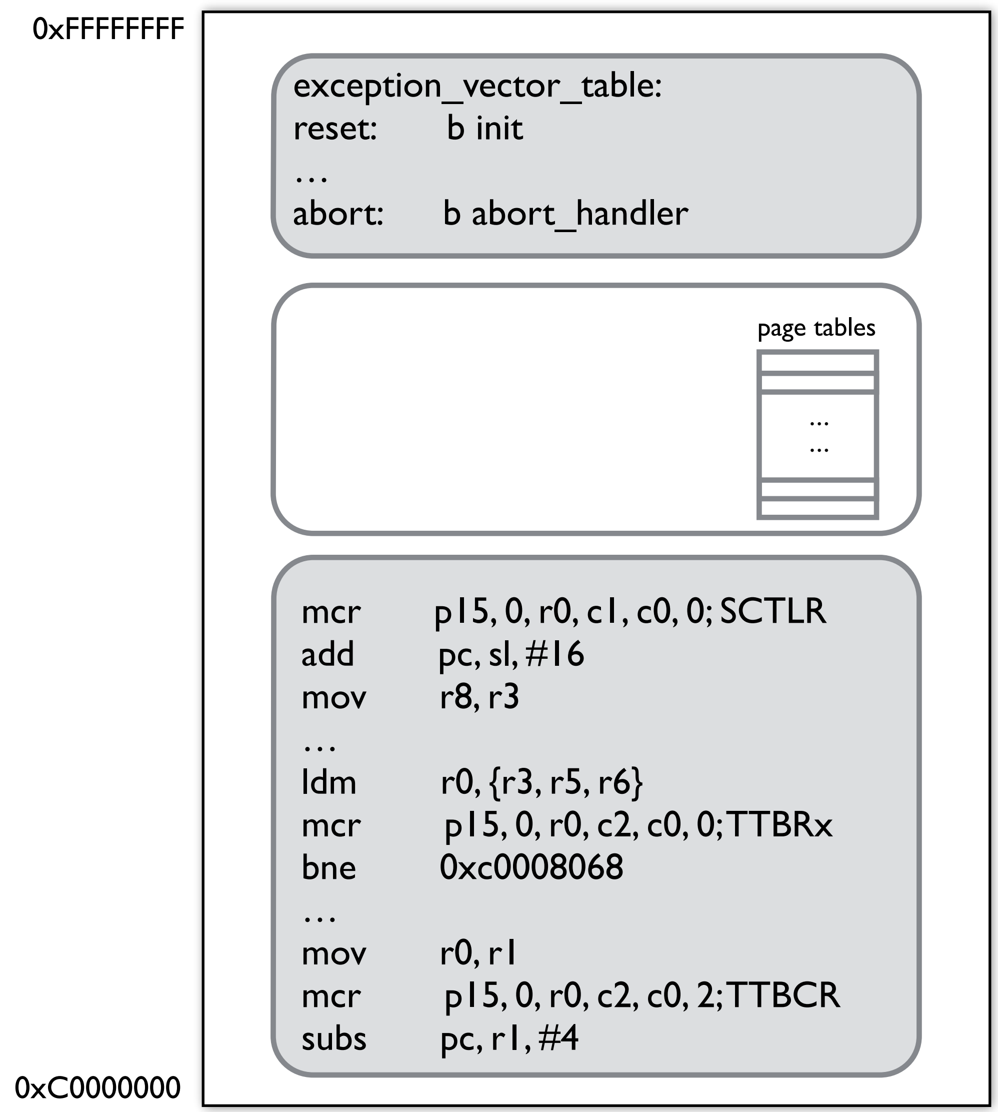
- Recall the specific attacks
  - Change to a different set of page tables that are under attacker's control
    - instrument all instructions that can be potentially used to switch the page table root
  - Modify page table entries in place

SPROBE Placement

- Recall the specific attacks
  - Change to a different set of page tables that are under attacker's control
    - instrument all instructions that can be potentially used to switch the page table root
  - Modify page table entries in place
    - write-protect the whole page tables and instrument the first instruction in page fault handler

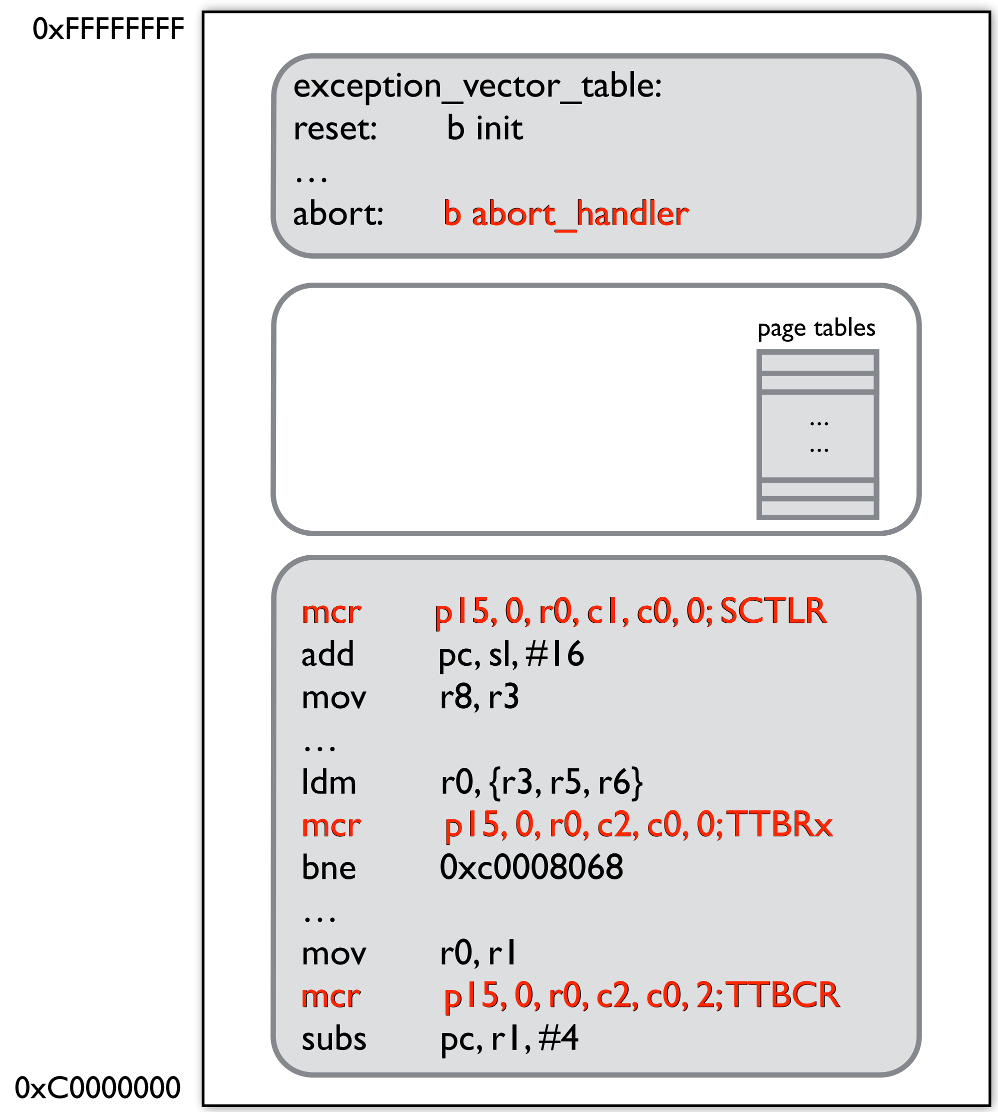
SPROBE Placement

Normal World Kernel Space



SPROBE Placement

Normal World Kernel Space



Evaluation

- Environment setup
  - Linux 2.6.38 in the normal world
  - Fast Models 8.1 for emulation
- Types of SPROBES
  - Type #1: 6 SPROBES for enforcing W⊕X protection
  - Type #2: 4 SPROBES for monitoring page table root
  - Type #3: 1 SPROBE for monitoring page table configuration
  - Type #4: 1 SPROBE for monitoring page table entries

Evaluation

- Def. Hit Frequency: the average number of instructions elapsed between two contiguous SPROBE hits

SPROBE Type	1	2	3	4
Hit Frequency	N/A	313,836	N/A	85,982
Overhead	0	1.8%	0	6.5%

Q & A