容器场景下的内核安全

申文博

浙江大学网络空间安全学院

2021-08-05

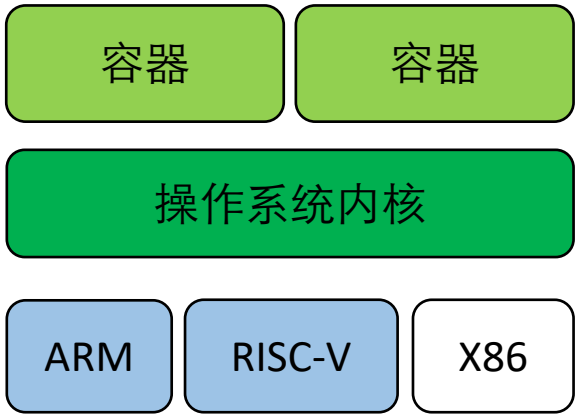
浙江大学 网络空间安全学院

SCHOOL OF CYBER SCIENCE AND TECHNOLOGY

ZHEJIANG UNIVERSITY

个人介绍

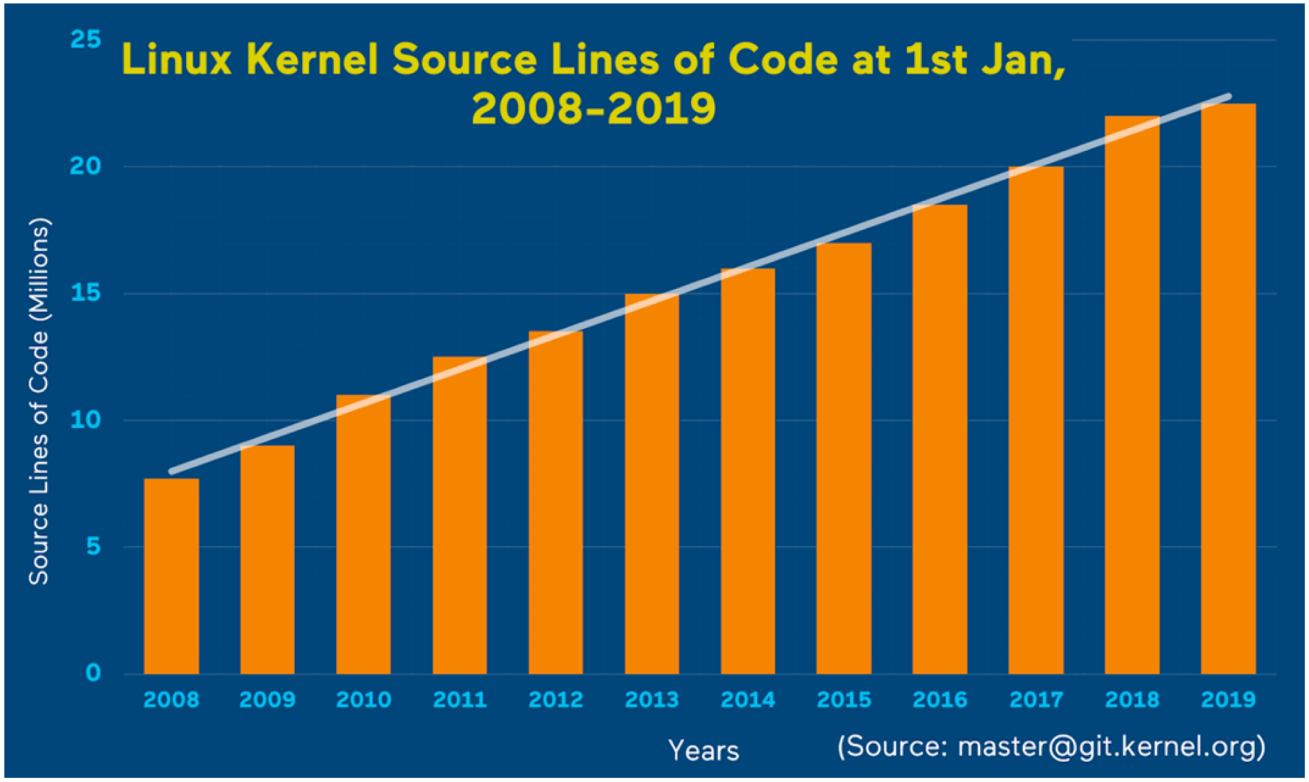
- 浙江大学百人计划研究员，博士生导师
  - 网络空间安全学院，计算机科学与技术学院
  - 研究方向：软硬件一体化保护，操作系统安全，容器安全，程序分析
- 内核安全技术负责人-三星美国研究院(硅谷)， 2015-2019
  - 根据实际攻击，设计、实现内核安全机制，部署超过亿部旗舰手机
  - 包含内核代码保护；内核控制流保护（2016年部署前向后向，移动内核首次）；内核数据保护
  - 论文发表于四大国际安全顶会: IEEE S&P, CCS, USENIX SEC, NDSS
- 美国北卡罗莱纳州立大学计算机博士，2015
- 哈尔滨工业大学学士，2010



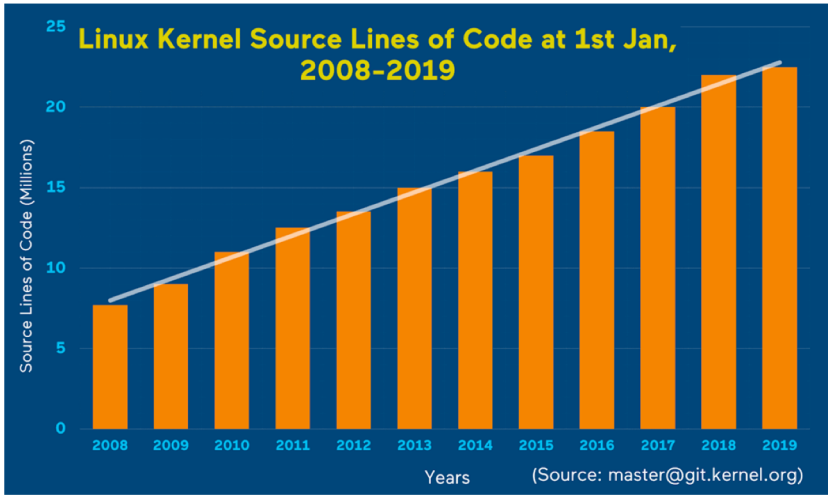
提纲

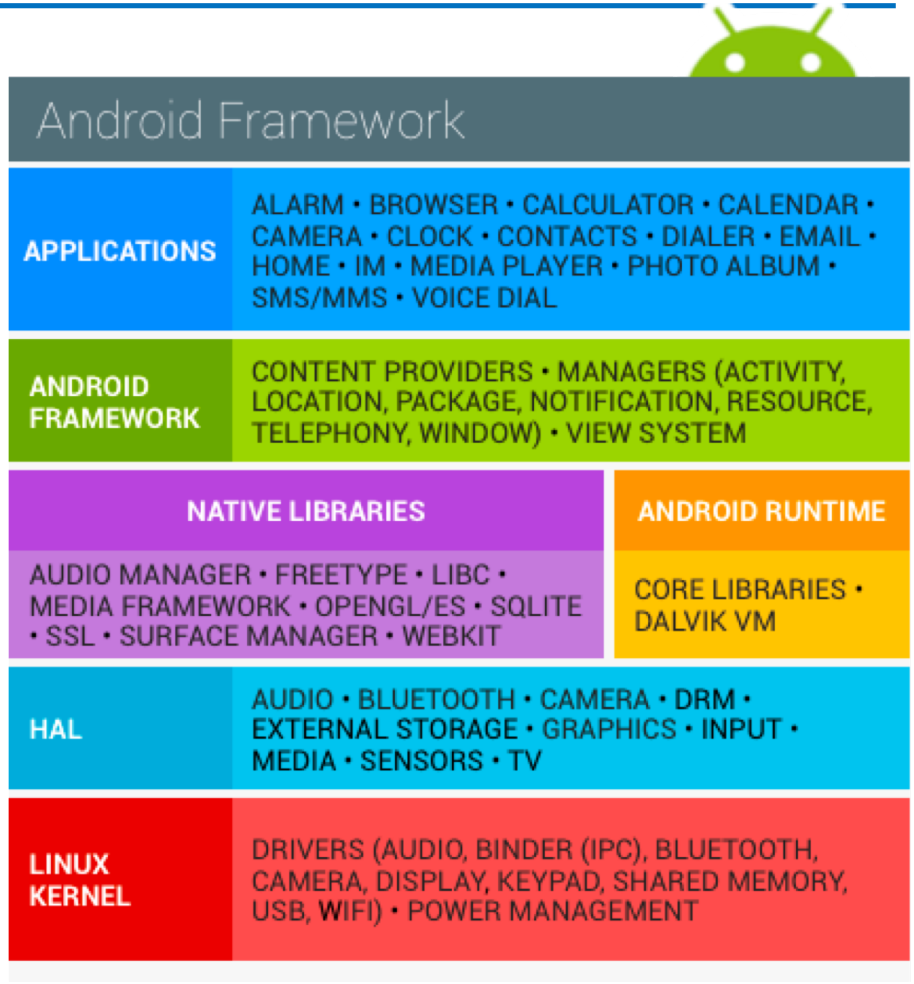
- 内核传统攻击和防护演化
  - 代码注入攻防
  - 代码重用攻防
  - 数据攻防
- 内核在容器场景下的安全问题
  - 抽象资源攻击
  - 内存计数问题

系统安全现状

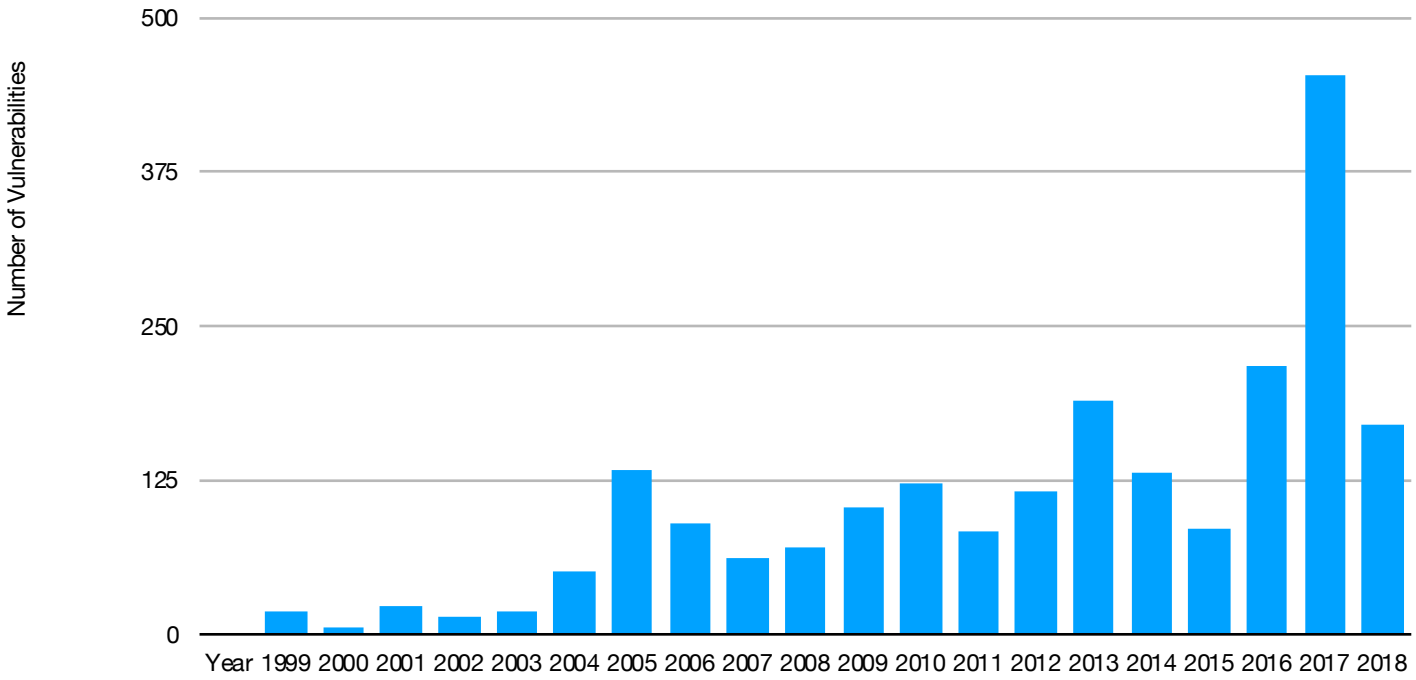

Linux Kernel Source Lines of Code at 1st Jan, 2008-2019
Source Lines of Code (Millions)
25
20
15
10
5
0
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
Years
(Source: master@git.kernel.org)

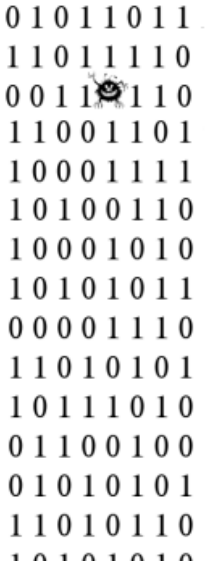
系统安全现状


Linux Kernel Source Lines of Code at 1st Jan, 2008-2019
Source Lines of Code (Millions)
25
20
15
10
5
0
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
Years
(Source: master@git.kernel.org)


Android Framework
APPLICATIONS
ALARM • BROWSER • CALCULATOR • CALENDAR • CAMERA • CLOCK • CONTACTS • DIALER • EMAIL • HOME • IM • MEDIA PLAYER • PHOTO ALBUM • SMS/MMS • VOICE DIAL
ANDROID FRAMEWORK
CONTENT PROVIDERS • MANAGERS (ACTIVITY, LOCATION, PACKAGE, NOTIFICATION, RESOURCE, TELEPHONY, WINDOW) • VIEW SYSTEM
NATIVE LIBRARIES
AUDIO MANAGER • FREETYPE • LIBC • MEDIA FRAMEWORK • OPENGL/ES • SQLITE • SSL • SURFACE MANAGER • WEBKIT
ANDROID RUNTIME
CORE LIBRARIES • DALVIK VM
HAL
AUDIO • BLUETOOTH • CAMERA • DRM • EXTERNAL STORAGE • GRAPHICS • INPUT • MEDIA • SENSORS • TV
LINUX KERNEL
DRIVERS (AUDIO, BINDER (IPC), BLUETOOTH, CAMERA, DISPLAY, KEYPAD, SHARED MEMORY, USB, WIFI) • POWER MANAGEMENT

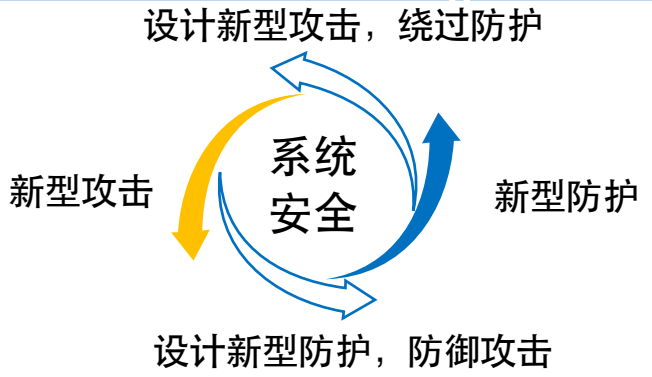
Linux内核现状

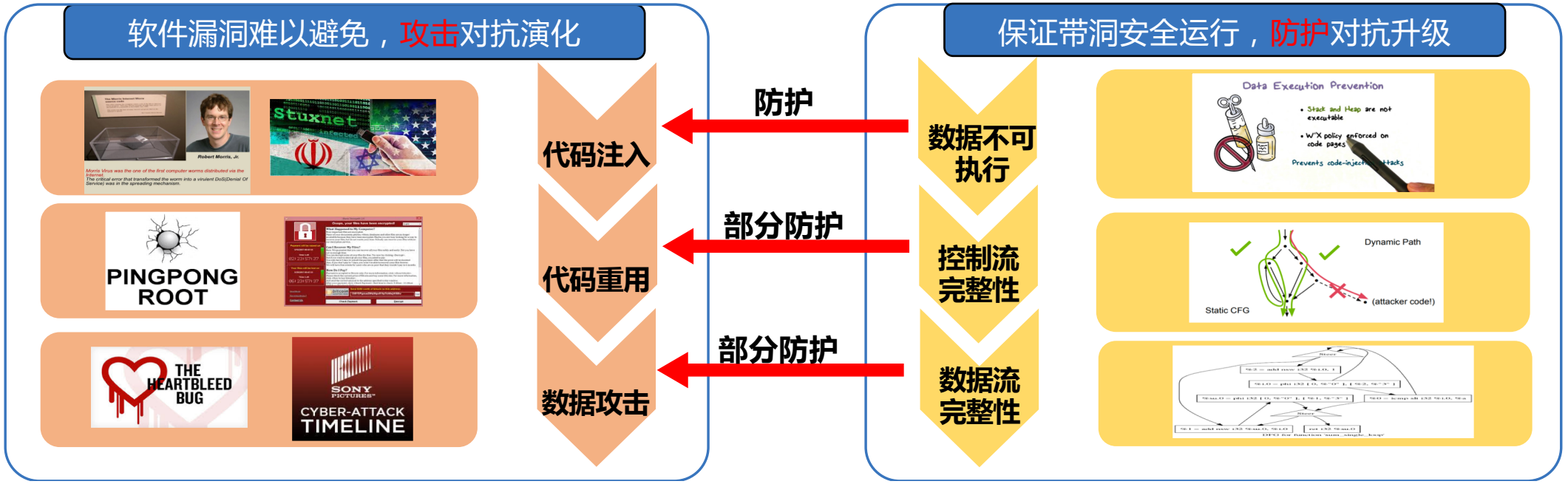

Number of Vulnerabilities
500
375
250
125
0
Year 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018



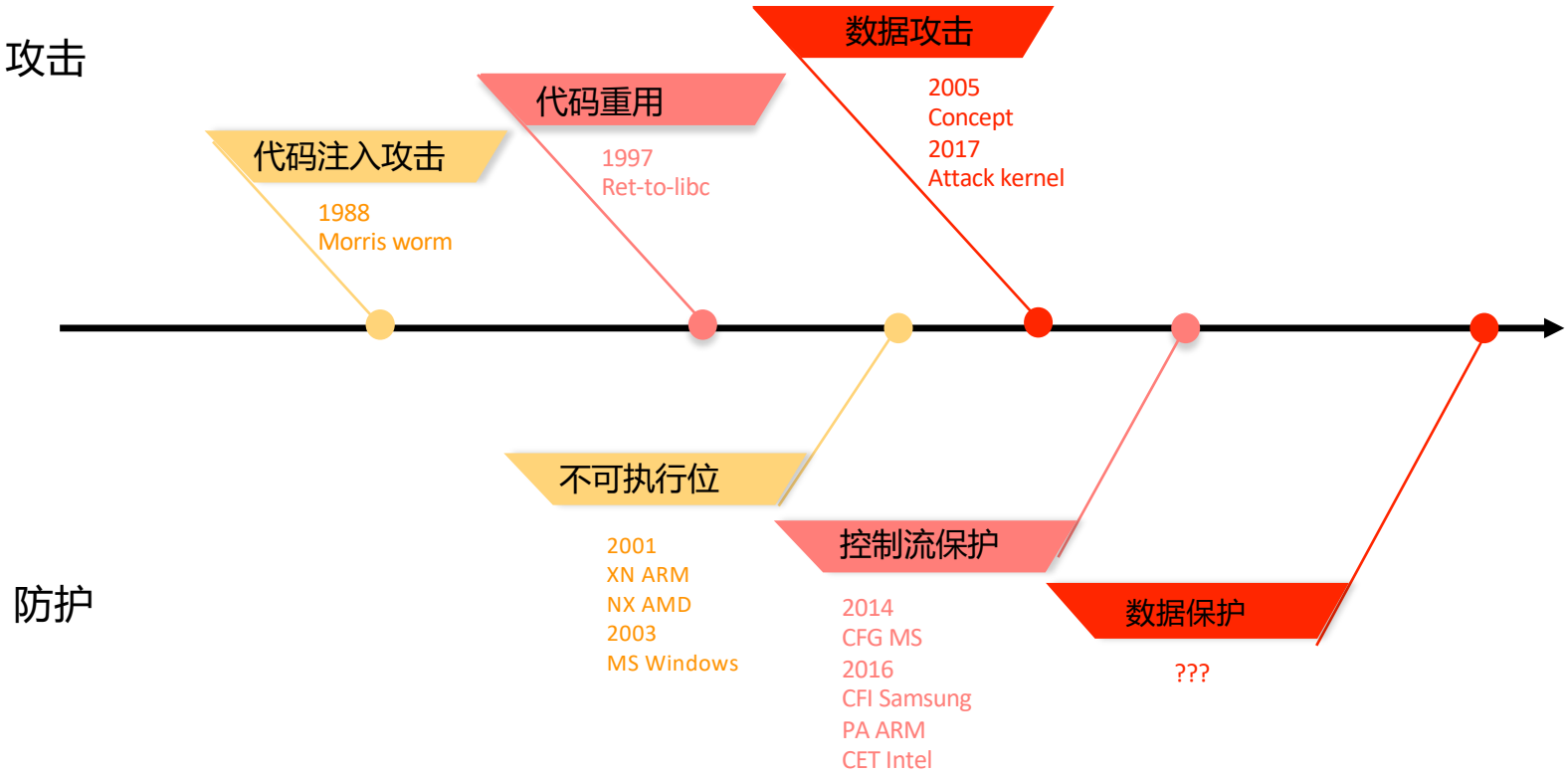
攻防演化

- 代码量巨大 → 软件漏洞无法避免
  → 新型攻击
- 新型攻击→新型防护
  - 内核攻防在不断的对抗中演化、升级



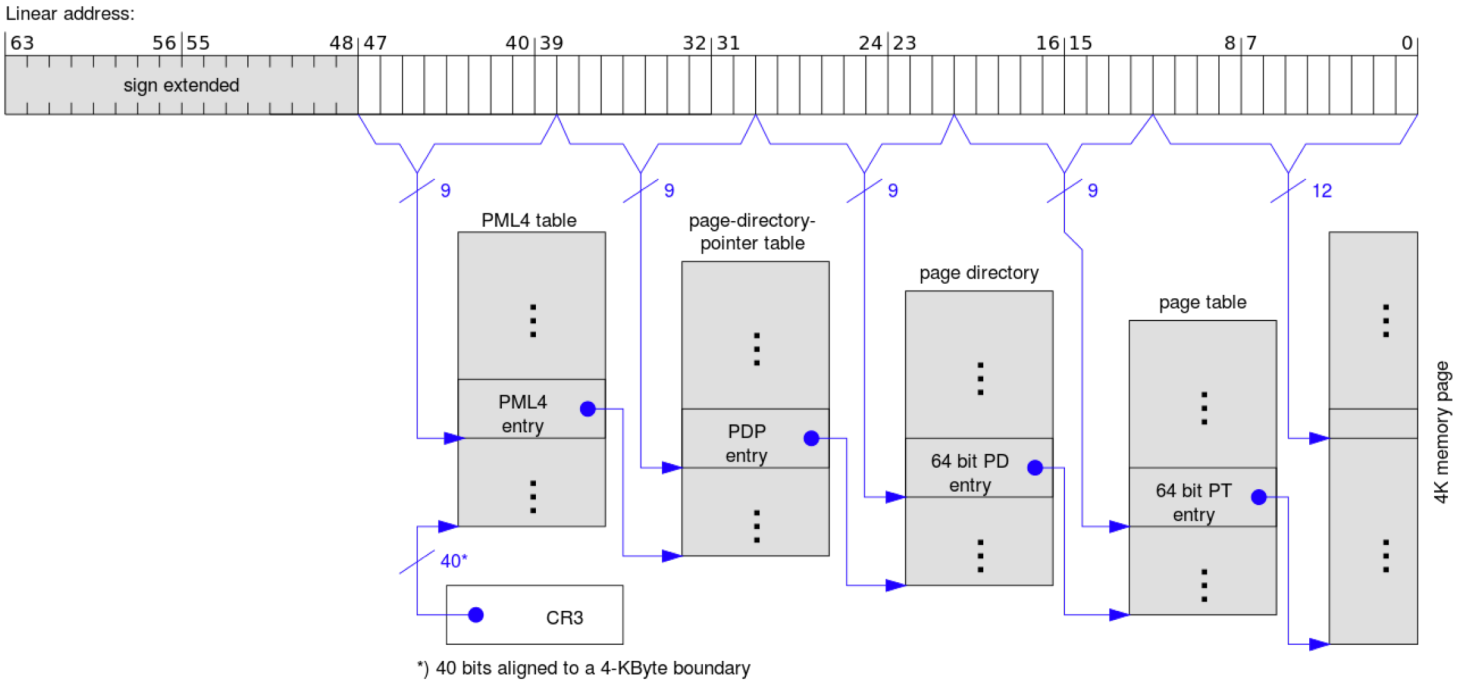


内核攻击和防护演化

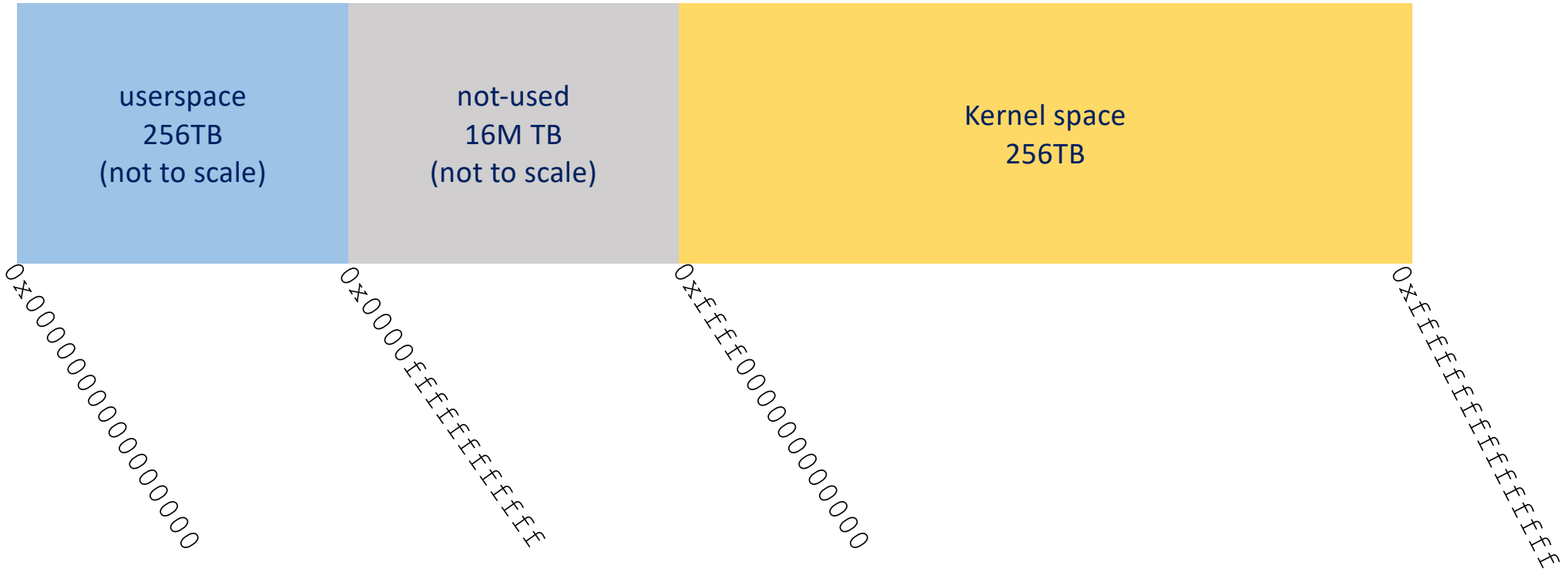

攻击
防护
代码注入攻击
1988
Morris worm
代码重用
1997
Ret-to-libc
数据攻击
2005
Concept
2017
Attack kernel
不可执行位
2001
XN ARM
NX AMD
2003
MS Windows
控制流保护
2014
CFG MS
2016
CFI Samsung
PA ARM
CET Intel
数据保护
???

内核页表Page Table

- 48-bit virtual address
  - 9+9+9+9+12

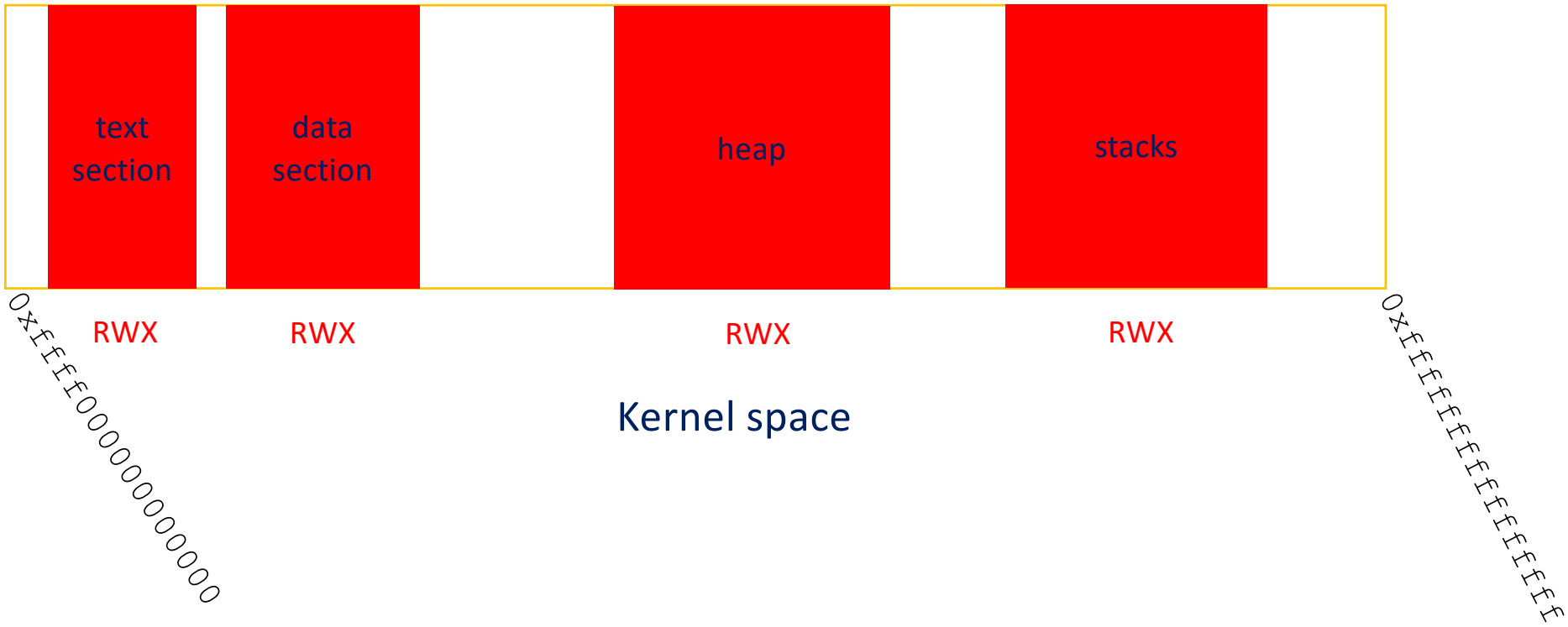


内核内存布局


userspace
256TB
(not to scale)
not-used
16M TB
(not to scale)
Kernel space
256TB
0x0000000000000
0x0000ffffffffffff
0xffff000000000000
0xffffffffffffffff

内核内存布局

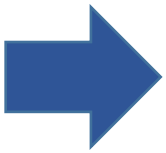
内核代码注入 内核代码重用 内核数据攻击



代码注入攻防

内核代码注入 | 内核代码重用 | 内核数据攻击

- 通过内核漏洞
  - 篡改已有代码text section
  - 注入新的代码
  - 或者跳到用户代码，比如jump-to-user
- 系统控制能力大
  - 可执行新代码
  - 危害大
- 多见于早期Linux
  - Kernel Text RWX (Android 2013)



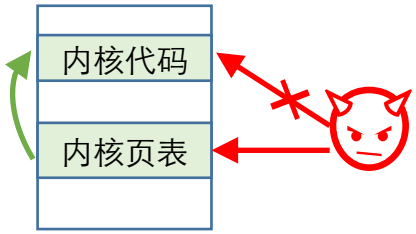
- 内核代码注入防护
  - 保护已有代码W^X
  - 硬件支持，杜绝注入
    - 数据不可执行(2001 XN ARM; NX AMD)
    - 特权不可执行(2011 SMEP Intel; PXN ARM)
- 通过内核页表来实现
  - 在内核页表设置相应的保护位，实现保护
  - 多数Android设备，包含Google Pixel
  - 防御性弱，内核页表被改掉即失效

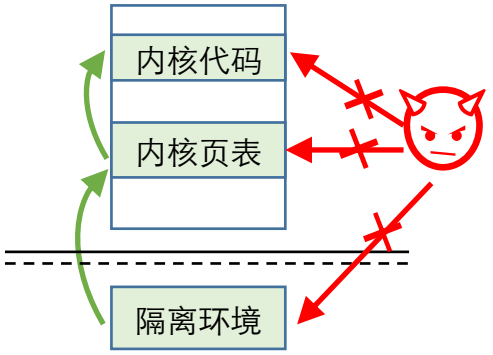
代码注入攻防

内核代码注入 | 内核代码重用 | 内核数据攻击

- 通过内核页表来实现
  - 对内核页表没有保护
  - 攻击者可篡改页表，去掉保护
  - 进而篡改代码

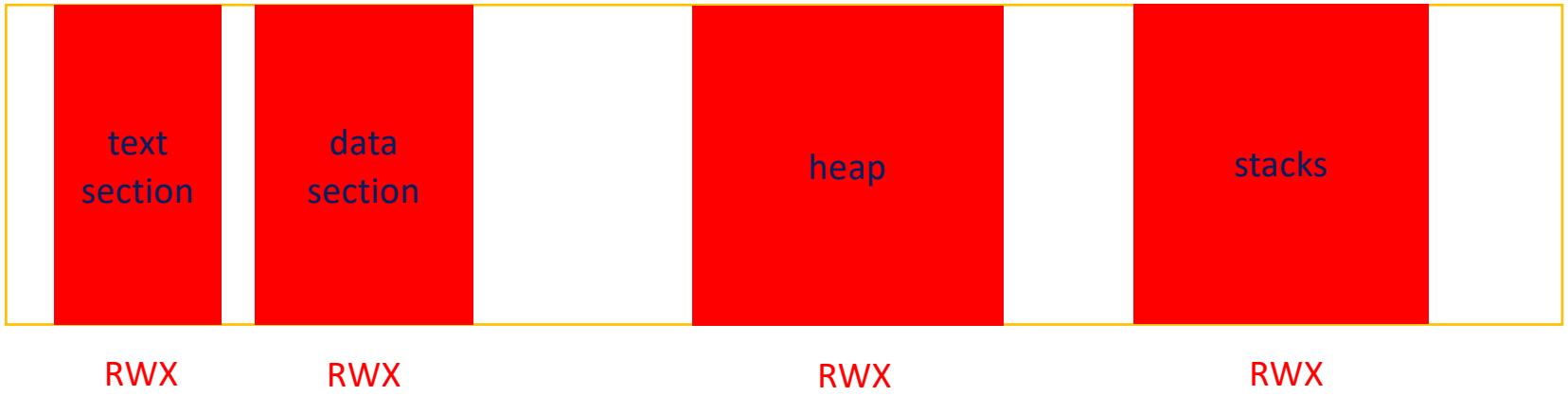
- 通过隔离环境保护内核页表
  - 通过隔离环境，避免内核漏洞影响
    - Intel VT, SGX; RISCV PMP
    - ARM TrustZone [ACM CCS 14]
    - SKEE [NDSS 16 Distinguished Paper Award]
  - 实现了纵深防御defense-in-depth





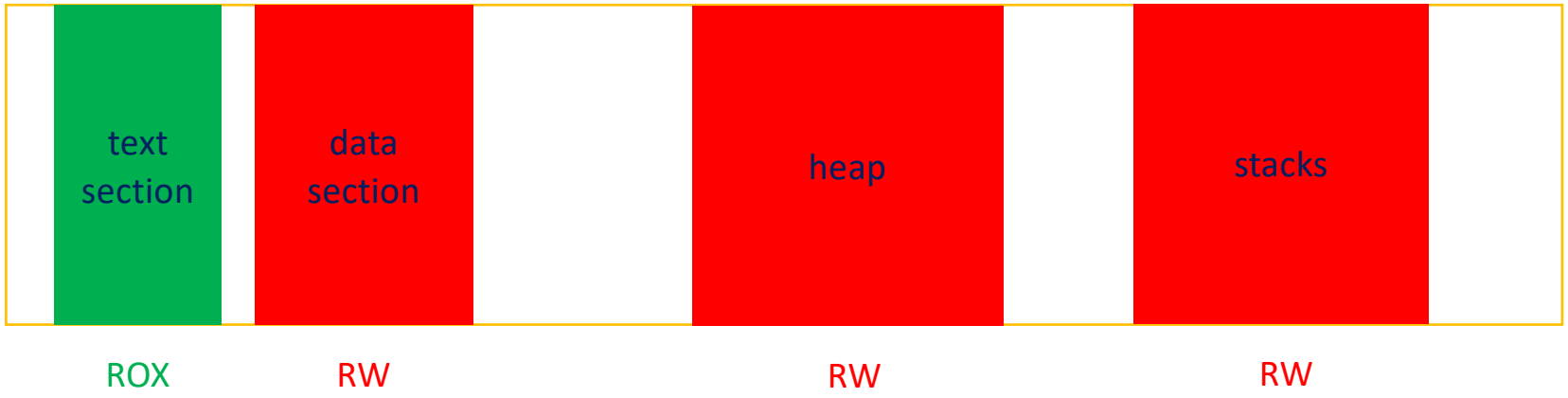
内核内存布局

内核代码注入 内核代码重用 内核数据攻击



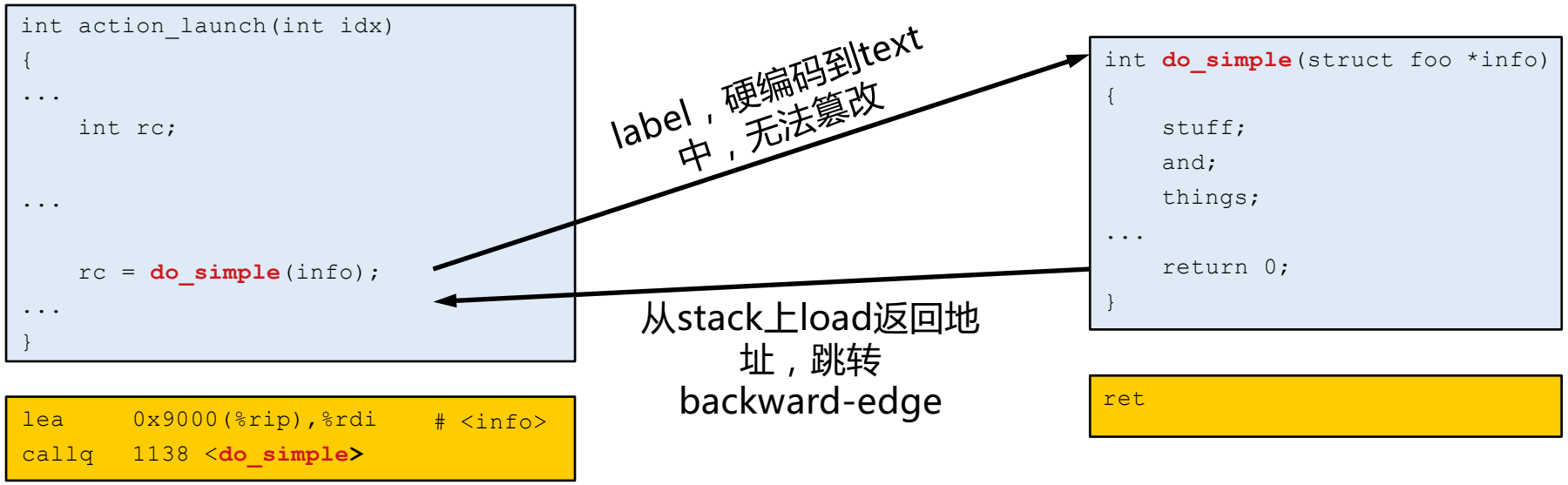
内核内存布局

内核代码注入 内核代码重用 内核数据攻击



内核代码重用攻击

- 程序的控制流转移
  - 函数调用，call graph
  - 直接跳转



Source: https://static.sched.com/hosted_files/ossna2020/b2/CFI%20in%20the%20Linux%20Kernel.pdf

内核代码重用攻击

- 程序的控制流转移
  - 间接跳转

```
typedef int (*func_ptr)(struct foo *);

func_ptr saved_actions[] = {
     do_simple,
     do_fancy,
     ...
};

int action_launch(int idx)
{
    func_ptr action;
    int rc;
...
    action = saved_actions[idx];
...
    rc = action(info);
...
}
```

```
lea    0x2ea6(%rip),%rax  # <saved_actions>
mov    (%rax,%rdi,8),%rax
lea    0x9000(%rip),%rdi  # <info>
callq  *%rax
```

从内存中load的func_ptr，跳转forward-edge

```
int do_simple(struct foo *info)
{
    stuff;
    and;
    things;
...
    return 0;
```

Source: https://static.sched.com/hosted_files/ossna2020/b2/CFI%20in%20the%20Linux%20Kernel.pdf

内核代码重用攻击

- 程序的控制流转移
  - 间接跳转

Global data section

Stack

```
typedef int (*func_ptr)(struct foo *);

func_ptr saved_actions[] = {
     do_simple,
     do_fancy,
     ...
};

int action_launch(int idx)
{
     func_ptr action;
     int rc;
...
     action = saved_actions[idx];
...
     rc = action(info);
...
}
```

```
lea    0x2ea6(%rip),%rax  # <saved_actions>
mov    (%rax,%rdi,8),%rax
lea    0x9000(%rip),%rdi  # <info>
callq  *%rax
```

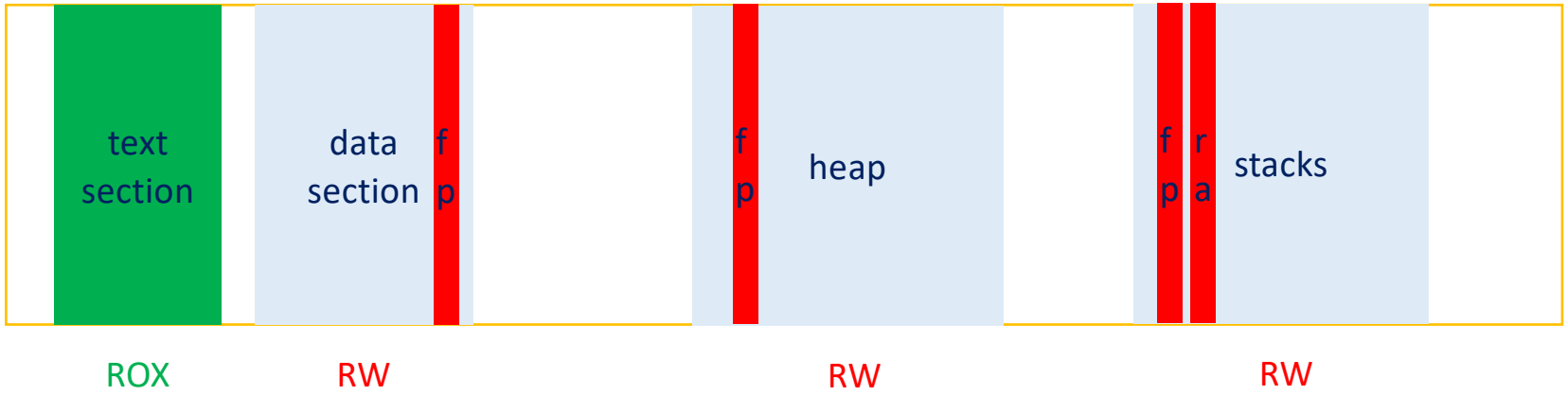
哪块内存，有保护吗

```
int do_simple(struct foo *info)
{
     stuff;
     and;
     things;
...
     return 0;
```

Source: https://static.sched.com/hosted_files/ossna2020/b2/CFI%20in%20the%20Linux%20Kernel.pdf

内核内存布局

内核代码注入 | 内核代码重用 | 内核数据攻击



内核代码重用攻击

内核代码注入 | 内核代码重用 | 内核数据攻击

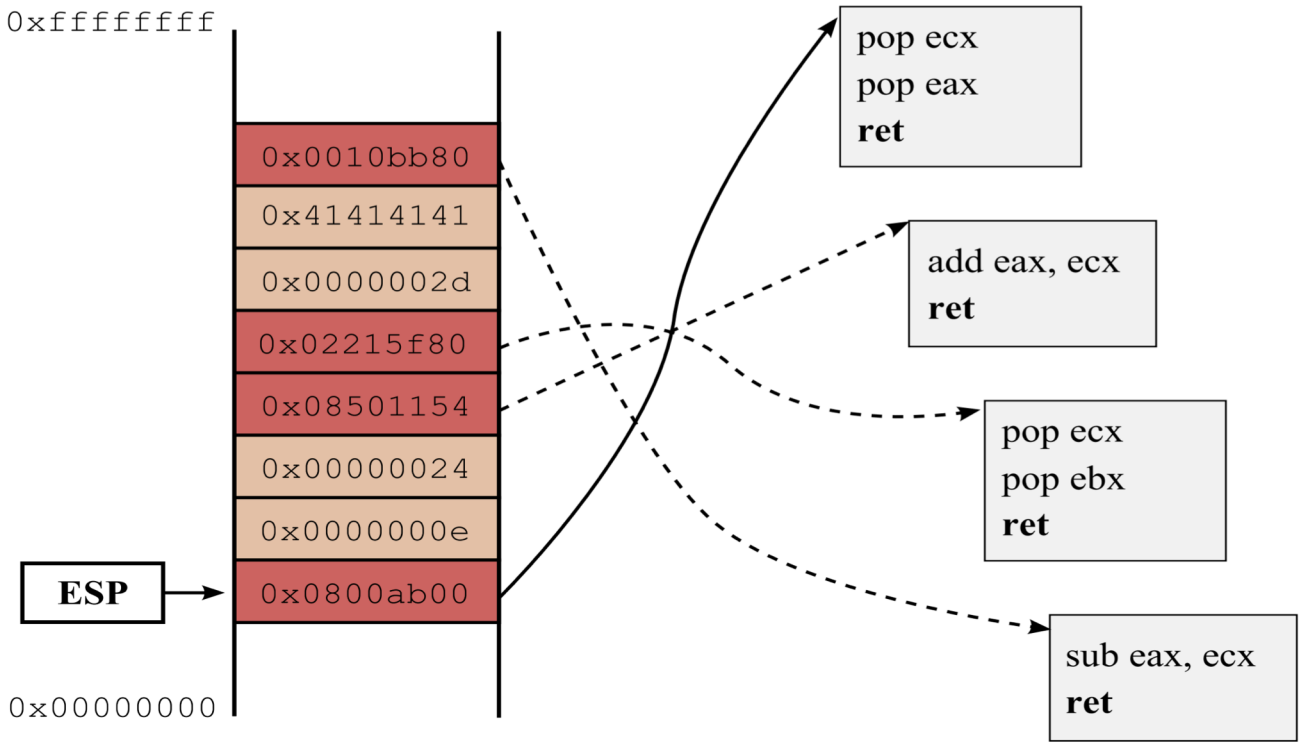
- 无法注入新代码，重用已有代码
- 通过篡改控制流，拼接已有函数片段，实现攻击函数
- 也被称为控制流劫持攻击

- 篡改控制数据
  - 返回地址 -> Return-oriented programming (ROP)
  - 函数指针 -> Jump-oriented programming (JOP)

内核代码重用攻击

内核代码注入 | 内核代码重用 | 内核数据攻击

- Return-oriented programming (ROP)
  - Stack is NX
  - 通过注入恶意返回地址，构造攻击函数
  - 0xe+ 0x24−0x2d

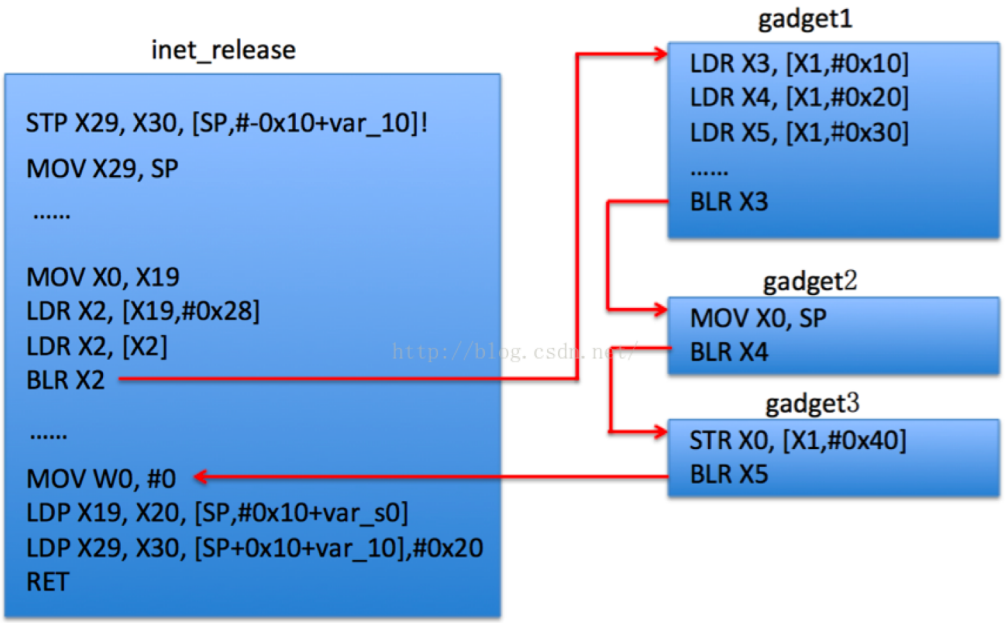


Source: Size Does Matter Why Using Gadget-Chain Length to Prevent Code-Reuse Attacks is Hard

内核代码重用攻击

内核代码注入 | 内核代码重用 | 内核数据攻击

- Jump-oriented programming (JOP)
  - 通过篡改函数指针，构造攻击函数

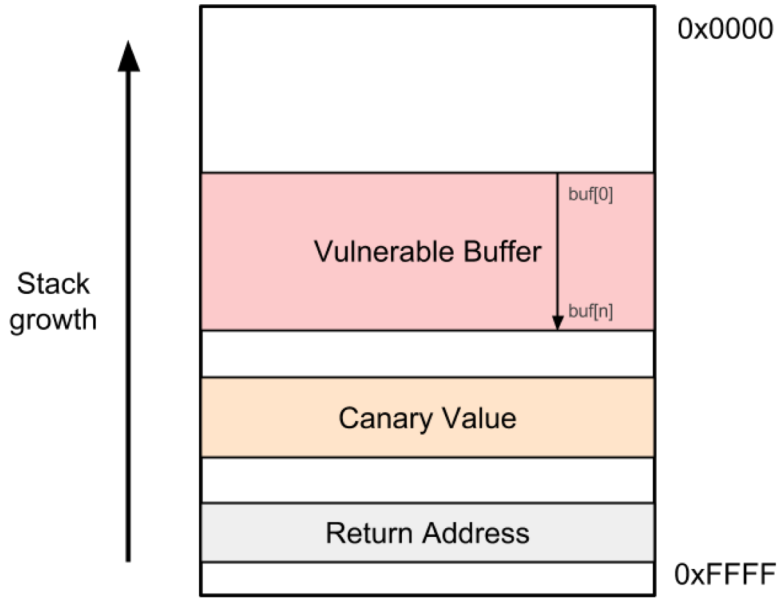


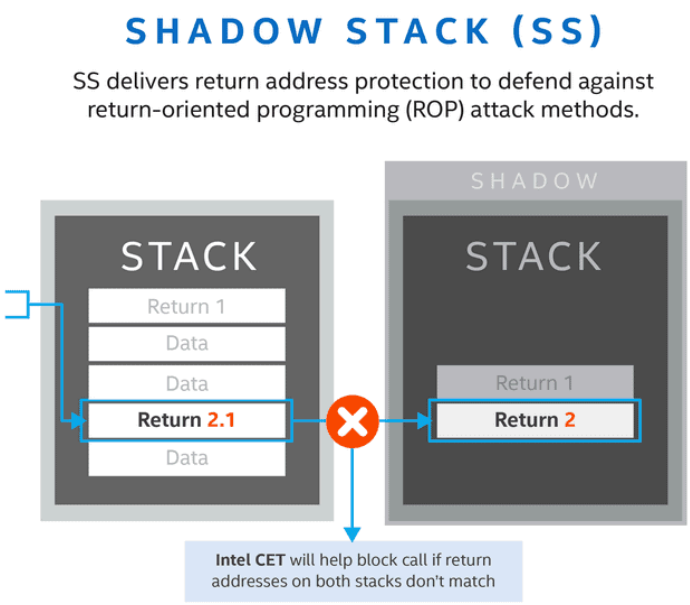
ROP泄露SP的大致思路

内核代码重用攻击

内核代码注入 | 内核代码重用 | 内核数据攻击

- 保护返回地址
  - Stack canary
  - Randomization
  - Shadow stack
- 硬件支持
  - Intel CET 2016

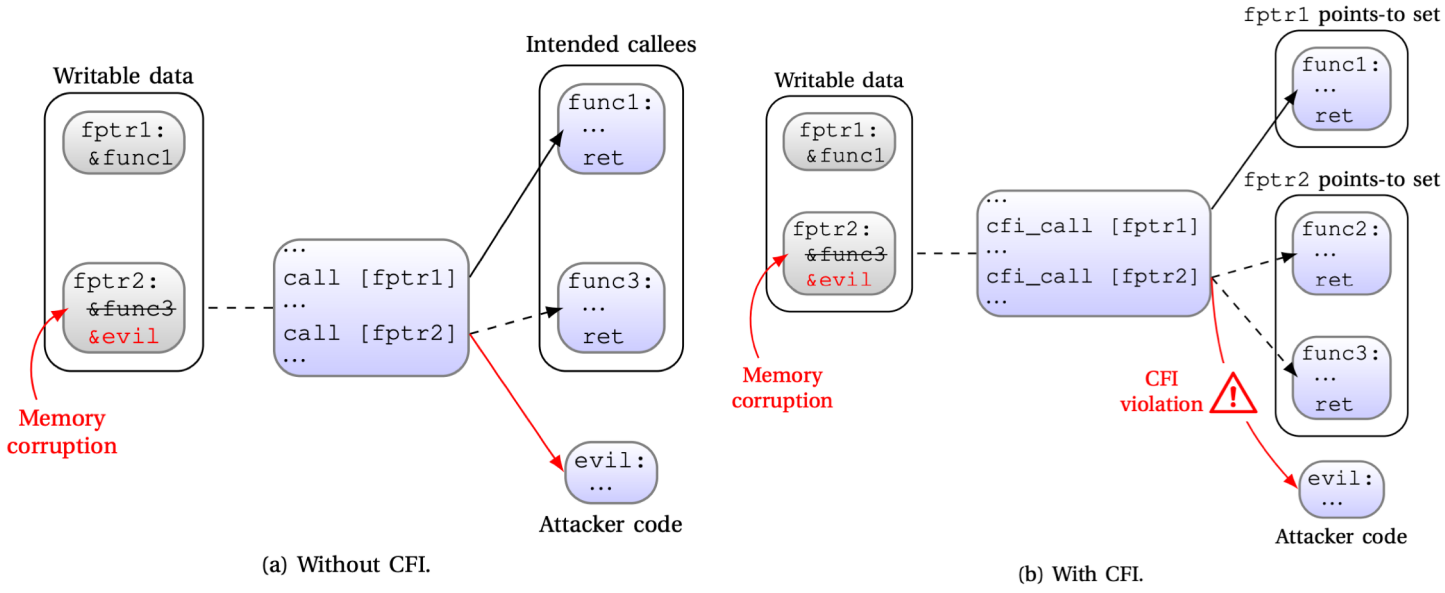




内核代码重用攻击

内核代码注入 | 内核代码重用 | 内核数据攻击

- 保护函数指针
  - 跳转检查
    - MS CFG, LLVM CFI
  - 指针保护
    - CPI, PA based
- 硬件支持
  - Intel CET 2016
  - ARM PA 2016



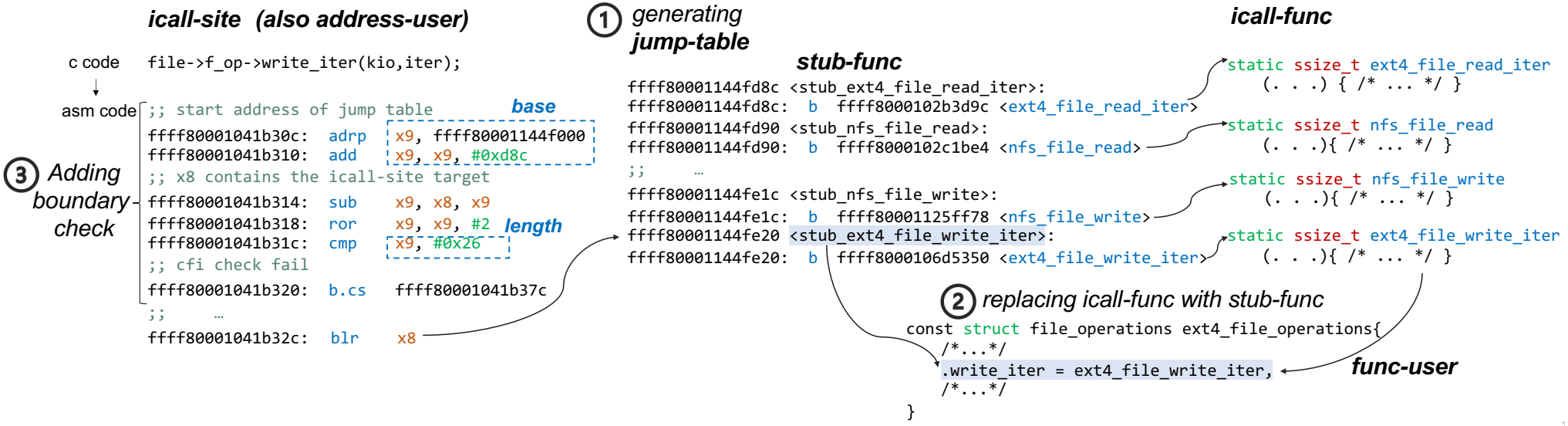
Source: Back To The Epilogue: Evading Control Flow Guard via Unaligned Targets

Source: https://eyalitkin.wordpress.com/2017/08/18/bypassing-return-flow-guard-rfg/

内核代码重用攻击

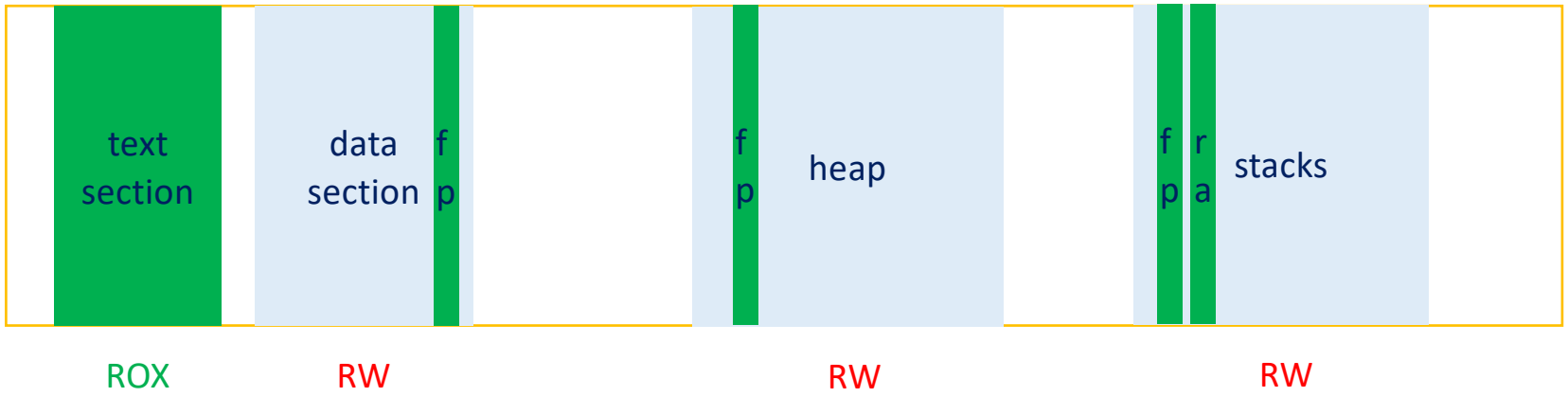
内核代码注入 | 内核代码重用 | 内核数据攻击

- 保护函数指针
  - 跳转检查
    - MS CFG, LLVM CFI
  - 指针保护
    - CPI, PA based
- 硬件支持
  - Intel CET 2016
  - ARM PA 2016



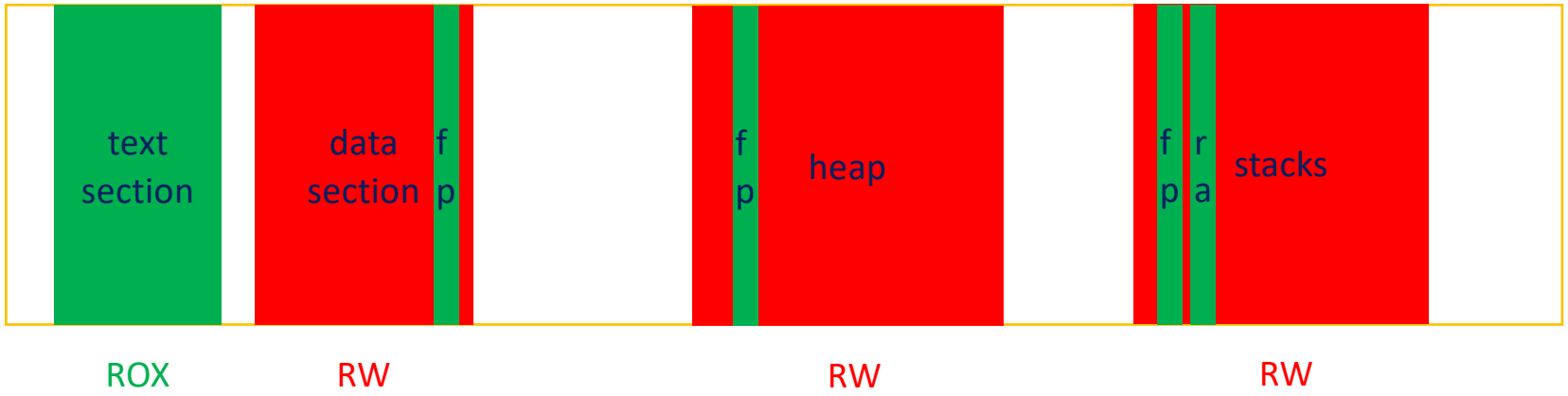
内核代码重用攻击

内核代码注入 → 内核代码重用 → 内核数据攻击



内核数据攻击

内核代码注入 内核代码重用 内核数据攻击



内核数据攻击

内核代码注入 > 内核代码重用 > 内核数据攻击

- 控制数据被保护后，攻击者提出非控制数据攻击
  - 返回地址和函数指针以外的数据
  - Data-oriented programming
  - 影响关键的安全特性
    - 仅利用非控制数据攻击做到内核提权-2017
- 非控制数据防护
  - 种类繁杂，难以实行统一有效保护
  - 主流操作系统均缺乏对数据攻击的有效防护

SELinux绕过

```
int ss_initialized;

void security_compute_av(u32 ssid,
                         u32 tsid,
                         u16 orig_tclass,
                         struct av_decision *avd,
                         struct extended_perms *xperms)
{
        ......
        if (!ss_initialized)
                goto allow;
        .....
}
```

内核数据攻击

SELinux绕过（被攻击v3.18）

```
int ss_initialized;

void security_compute_av(u32 ssid,
                         u32 tsid,
                         u16 orig_tclass,
                         struct av_decision *avd,
                         struct extended_perms *xperms)
{
        ......
        if (!ss_initialized)
                goto allow;
        .....
}
```

SELinux依然可被绕过（最新v5.14）

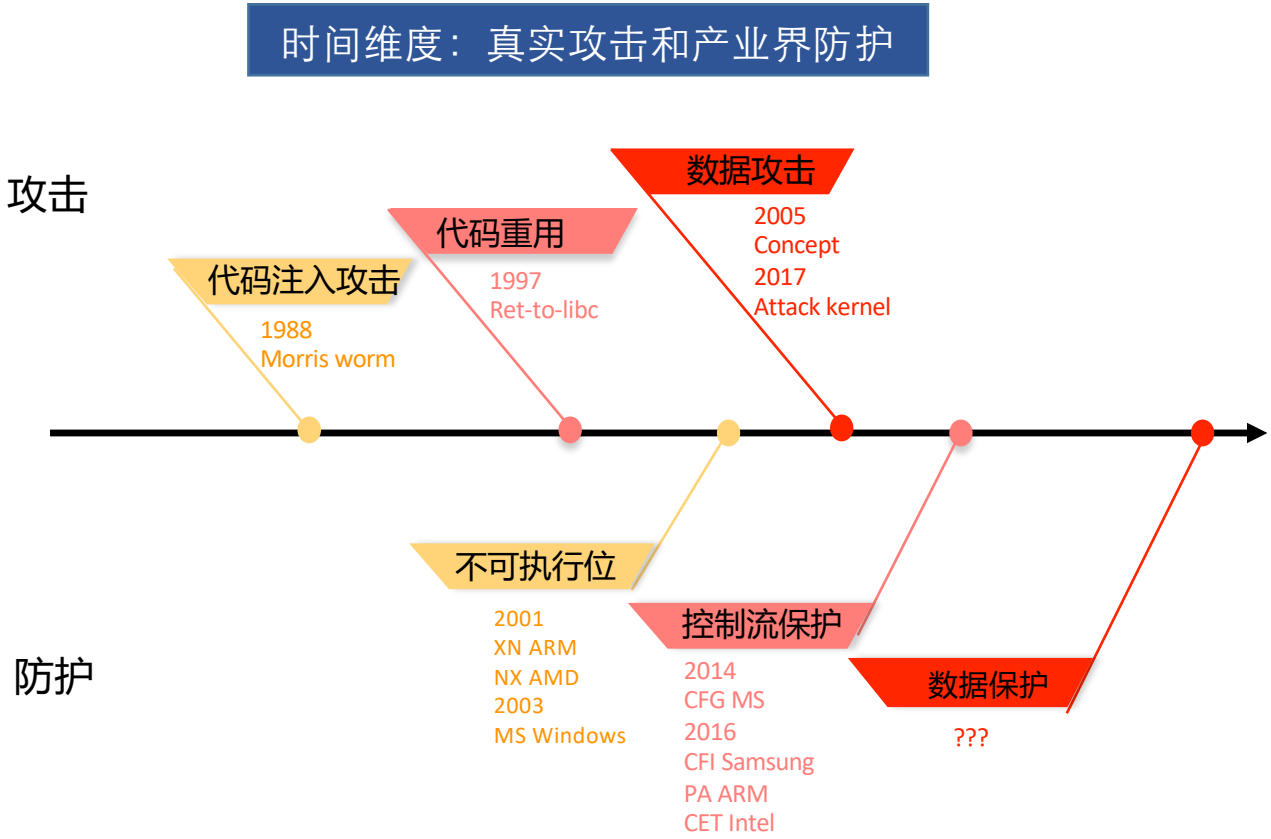
```
struct selinux_state {
        ......
        bool initialized;
        ......
} __randomize_layout;

struct selinux_state selinux_state;

void security_compute_av(struct selinux_state *state,
{
        ......
        if (!selinux_initialized(state))
                goto allow;
        ......
}
```

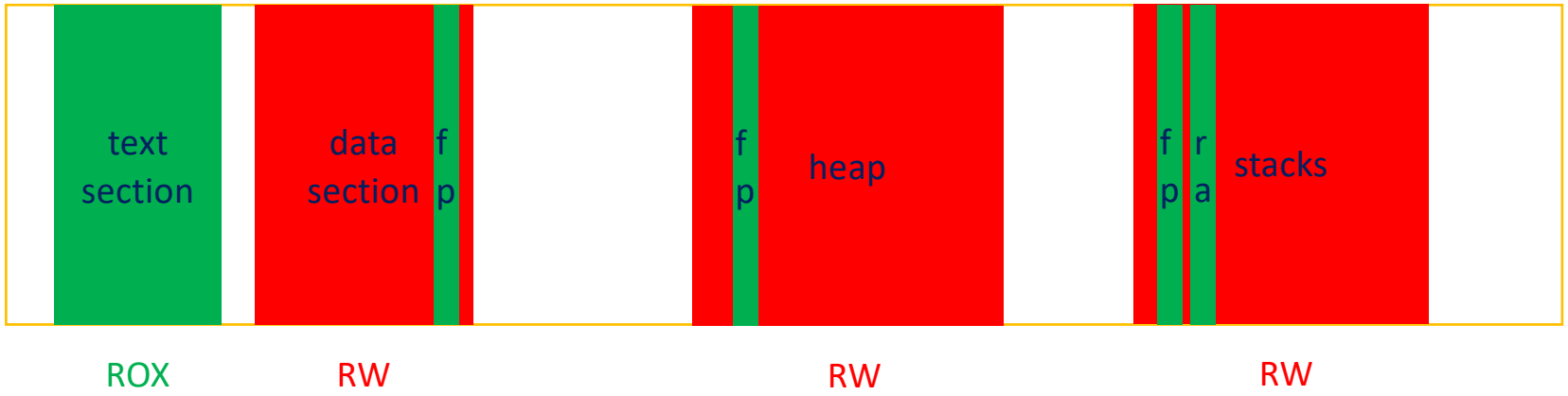
内核攻防演化

- 攻击演化
  - 攻击难度指数级增加
    - 复杂性指数级增加
    - 隐蔽性在增加
  - 控制能力缩小
    - 数据攻击依然能root内核
- 防护演化
  - 软件到硬件
  - 学术界原型到产业界实用方案
  - 有滞后性



内核保护现状

内核代码注入 > 内核代码重用 > **内核数据攻击**

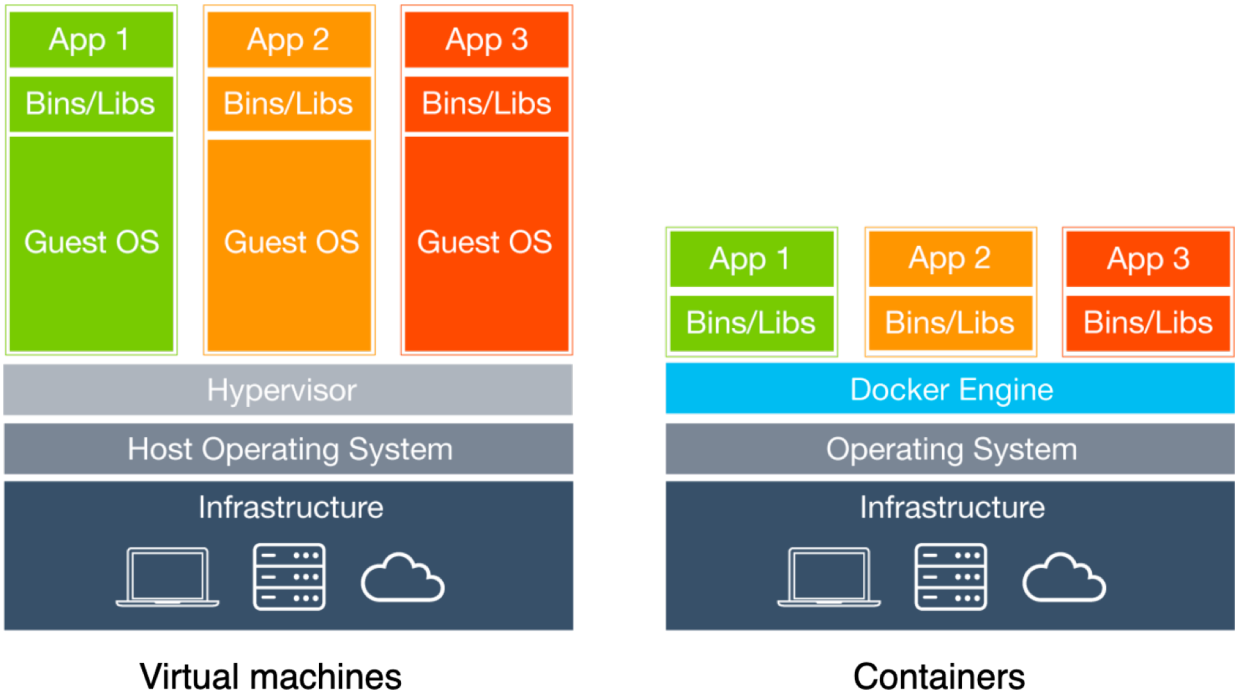


提纲

- 内核传统攻击和防护演化
  - 代码注入攻防
  - 代码重用攻防
  - 数据攻防
- 内核在容器场景下的安全问题
  - 抽象资源攻击
  - 内存计数问题

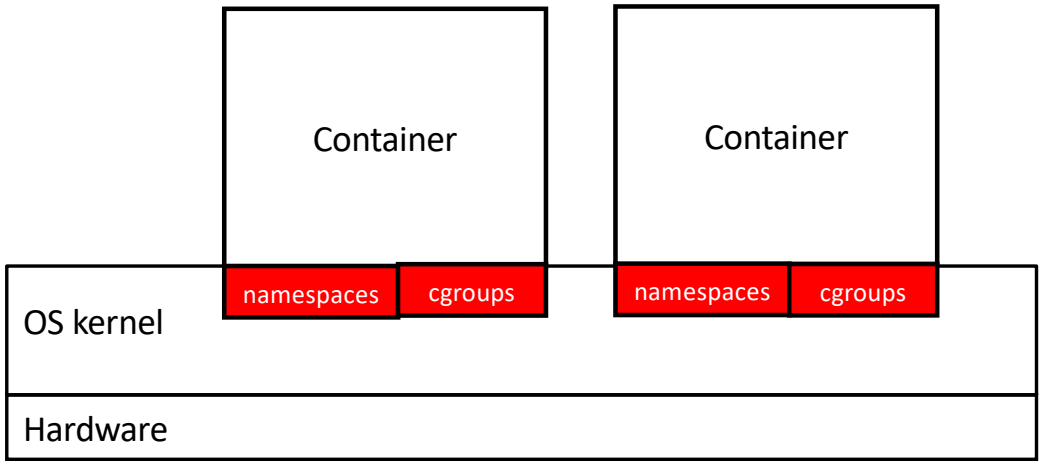
容器介绍

- 操作系统级虚拟化
  - 由同一内核虚拟出多个用户空间实例，无需维护单独内核
- 具有效率高、启动快、配置灵活等特点



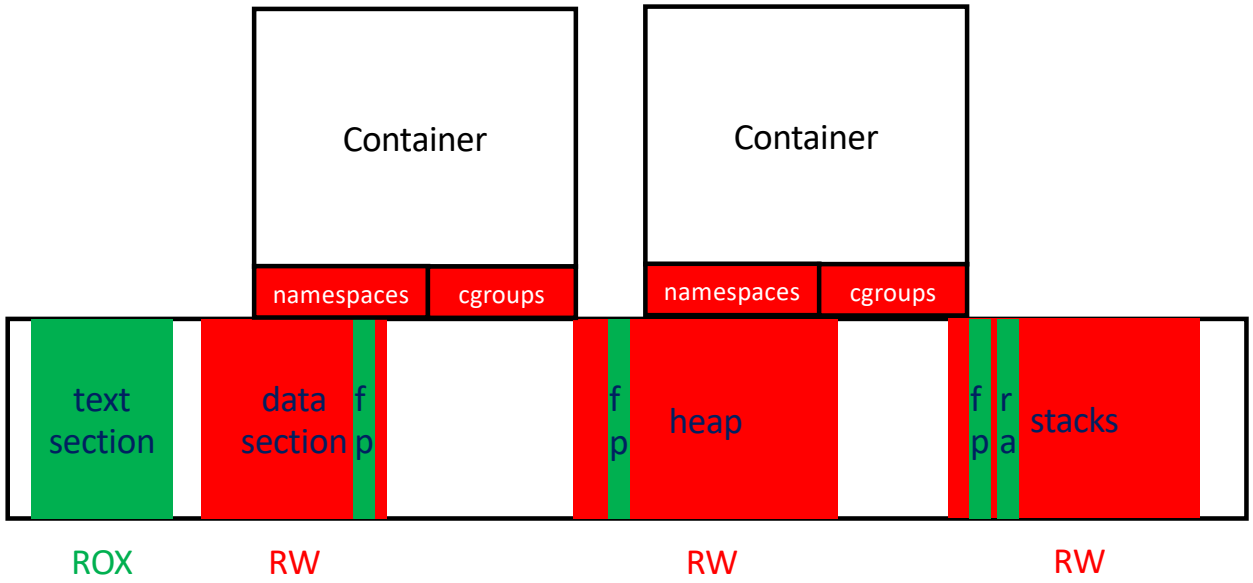
容器介绍

- 由namespaces负责隔离
  - 当前内核支持8种namespaces
  - 包含UTS、IPC、mount、PID、network、user、time、cgroup
- 由control groups进行限制
  - 当前内核支持13种cgroups
  - 主要用于限制CPU、内存和设备资源
- Seccomp, MAC, …



容器资源隔离

- 本质是基于进程的资源隔离
  - +namespaces and cgroups
- 进程能access哪些内核资源
  - Kernel text
    - Ktrace
  - Global data
  - Heap
  - Stack
- Kernel provides 300+ syscalls
  - Has complex data dependencies
  - Introduces new attacks to multi-tenant containers



Takeaway

- 内核安全性在对抗中大幅提升，但对数据攻击防护依然不足
  - 代码注入攻击
  - 代码重用攻击
  - 数据攻击
- 容器场景给内核带来了新的机遇，也带来了新的安全挑战
  - 抽象资源攻击
  - 内存计数问题

谢谢

浙江大学网络空间安全学院
https://icsr.zju.edu.cn

欢迎报考浙大网安

- 招收有志于系统安全学生
  - 本科生
  - 硕士生
  - 博士生
  - 博士后
  - Research assistant
- 邮箱
  - shenwenbo@zju.edu.cn

- 容器系统安全
  - 关键技术安全性分析
  - 新型容器隔离方案
  - 容器横向/纵向防御研究
- 操作系统内核安全
  - 运行前代码分析
  - 运行时内核保护
- 安全硬件
  - 基于已有硬件，构建系统防护
  - 抽象安全需求，设计新型硬件