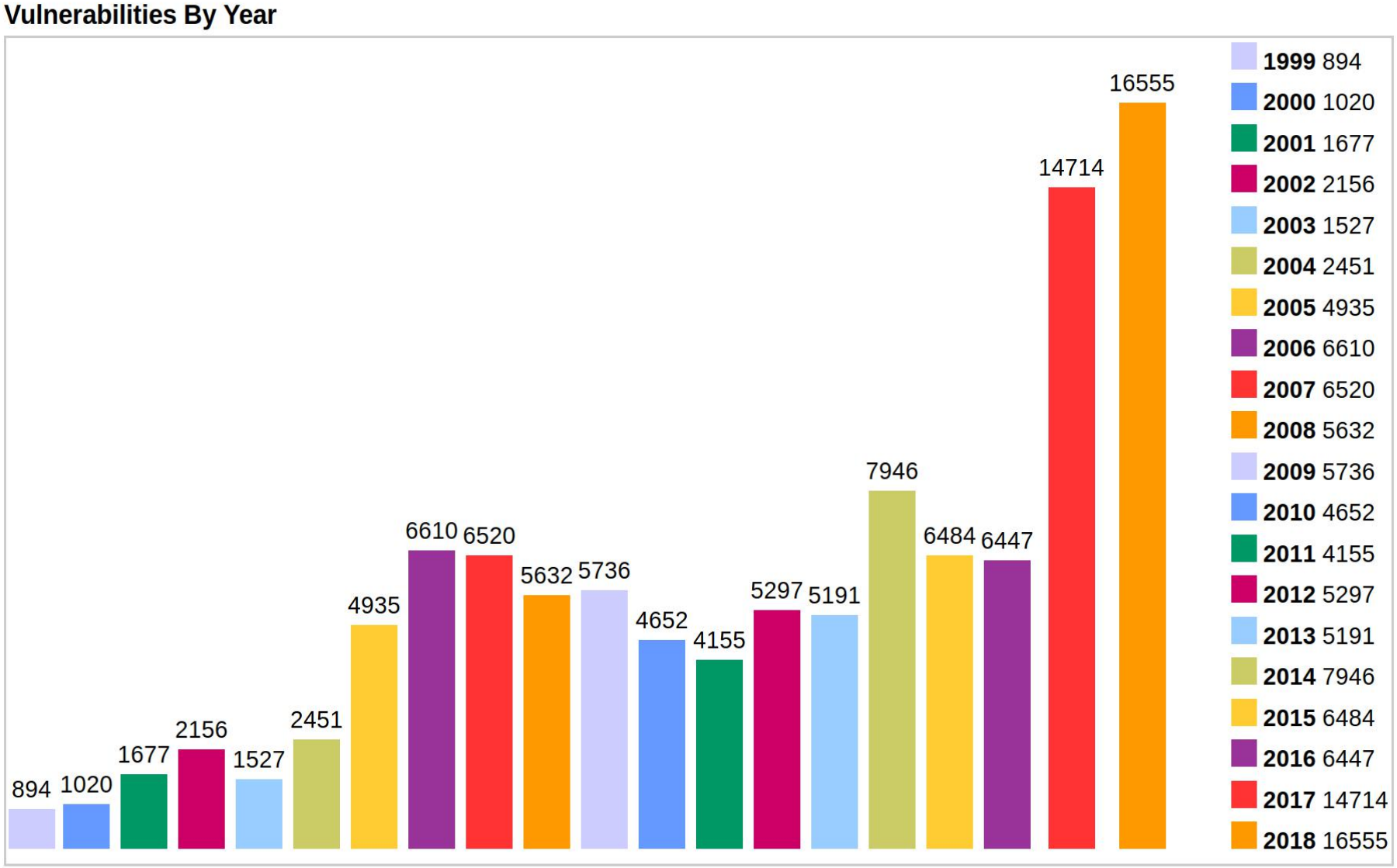
毕业设计答辩

基于二进制动态翻译的ROP攻击检测方法研究与实现

答辩人：刘天祺　　指导教师：詹静

背景：近20年CVE漏洞数量

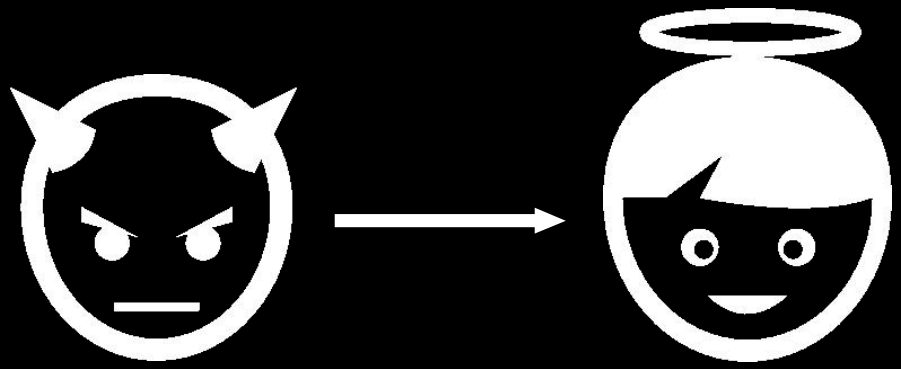


背景：ROP攻击发展现状

代码注入攻击

背景：ROP攻击发展现状

代码注入攻击　　数据执行保护



背景：ROP攻击发展现状

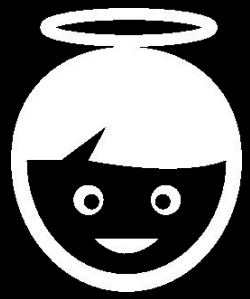
代码注入攻击　　数据执行保护　　代码复用攻击

Return-into-libc攻击
ROP攻击

背景：ROP攻击发展现状

ROP攻击

地址空间布局随机化
(ASLR)



控制流完整性
(CFI)

二进制动态插桩检测
ROPdefender/DROP

返回导向编程(ROP)

```
sub    rsp,0x8
mov    rax, [rip+0x200bb5]
test   rax,rax
je     40044a <_init+0x12>
call   rax
add    rsp,0x8
ret
mov    esi,0x601038
push   rbp
sub    rsi,0x601038
mov    rbp,rsp
sar    rsi,0x3
mov    rax,rsi
shr    rax,0x3f
add    rsi,rax
sar    rsi,1
je     400528
mov    eax,0x0
test   rax,rax
je     400528
```

```
pop    rbp
mov    edi,0x601038
jmp    rax
nop    DWORD PTR [rax]
pop    rbp
ret
push   r15
push   r14
mov    r15,rdx
push   r13
push   r12
lea    r12,[rip+0x2007c6]
push   rbp
lea    rbp,[rip+0x2007c6]
push   rbx
mov    r13d,edi
mov    r14,rsi
sub    rbp,r12
sub    rsp,0x8
sar    rbp,0x3
```

```
test   rbp,rbp
je     400686
xor    ebx,ebx
nop
mov    rdx,r15
mov    rsi,r14
mov    edi,r13d
call   [r12+rbx*8]
add    rbx,0x1
cmp    rbp,rbx
jne    400670
add    rsp,0x8
pop    rbx
pop    rbp
pop    r12
pop    r13
pop    r14
pop    r15
ret
nop
```

返回导向编程(ROP)

```
sub    rsp,0x8
mov    rax, [rip+0x200bb5]
test   rax,rax
je     40044a <_init+0x12>
call   rax
add    rsp,0x8
ret
mov    esi,0x601038
push   rbp
sub    rsi,0x601038
mov    rbp,rsp
sar    rsi,0x3
mov    rax,rsi
shr    rax,0x3f
add    rsi,rax
sar    rsi,1
je     400528
mov    eax,0x0
test   rax,rax
je     400528
```

```
pop    rbp
mov    edi,0x601038
jmp    rax
nop    DWORD PTR [rax]
pop    rbp
ret
push   r15
push   r14
mov    r15,rdx
push   r13
push   r12
lea    r12,[rip+0x2007c6]
push   rbp
lea    rbp,[rip+0x2007c6]
push   rbx
mov    r13d,edi
mov    r14,rsi
sub    rbp,r12
sub    rsp,0x8
sar    rbp,0x3
```

```
test   rbp,rbp
je     400686
xor    ebx,ebx
nop
mov    rdx,r15
mov    rsi,r14
mov    edi,r13d
call   [r12+rbx*8]
add    rbx,0x1
cmp    rbp,rbx
jne    400670
add    rsp,0x8
pop    rbx
pop    rbp
pop    r12
pop    r13
pop    r14
pop    r15
ret
nop
```

返回导向编程(ROP)

```
sub    rsp,0x8
mov    rax, [rip+0x200bb5]
test   rax,rax
je     40044a <_init+0x12>
call   rax
add    rsp,0x8
ret
mov    esi,0x601038
push   rbp
sub    rsi,0x601038
mov    rbp,rsp
sar    rsi,0x3
mov    rax,rsi
shr    rax,0x3f
add    rsi,rax
sar    rsi,1
je     400528
mov    eax,0x0
test   rax,rax
je     400528
```

```
pop    rbp
mov    edi,0x601038
jmp    rax
nop    DWORD PTR [rax]
pop    rbp
ret
push   r15
push   r14
mov    r15,rdx
push   r13
push   r12
                0x2007c6]
                0x2007c6]
push   rbx
mov    r13d,edi
mov    r14,rsi
sub    rbp,r12
sub    rsp,0x8
sar    rbp,0x3
```

Gadget

```
test   rbp,rbp
je     400686
xor    ebx,ebx
nop
mov    rdx,r15
mov    rsi,r14
mov    edi,r13d
call   [r12+rbx*8]
add    rbx,0x1
cmp    rbp,rbx
jne    400670
add    rsp,0x8
pop    rbx
pop    rbp
pop    r12
pop    r13
pop    r14
pop    r15
ret
nop
```

返回导向编程(ROP)

```
sub    rsp,0x8
mov    rax, [rip+0x200bb5]
test   rax,rax
je     40044a <_init+0x12>
call   rax
add    rsp,0x8
ret
mov    esi,0x601038
push   rbp
sub    rsi,0x601038
mov    rbp,rsp
sar    rsi,0x3
mov    rax,rsi
shr    rax,0x3f
add    rsi,rax
sar    rsi,1
je     400528
mov    eax,0x0
test   rax,rax
je     400528
```

```
pop    rbp
mov    edi,0x601038
jmp    rax
nop    DWORD PTR [rax]
pop    rbp
ret
push   r15
push   r14
mov    r15,rdx
push   r13
push   r12
[illegible]0x2007c6]
[illegible]0x2007c6]
push   rbx
mov    r13d,edi
mov    r14,rsi
sub    rbp,r12
sub    rsp,0x8
sar    rbp,0x3
```

变种Gadget

Gadget

```
[illegible]p
[illegible]6
xor    ebx,ebx
nop
mov    rdx,r15
mov    rsi,r14
mov    edi,r13d
call   [r12+rbx*8]
add    rbx,0x1
cmp    rbp,rbx
jne    400670
add    rsp,0x8
pop    rbx
pop    rbp
pop    r12
pop    r13
pop    r14
pop    r15
ret
nop
```

检测策略——指令特征

如果能够识别出攻击代码的特征

...就能检测出ROP攻击

检测策略——指令特征

ROP攻击代码的特征：

1. 连续执行gadget指令序列
2. 大量利用返回指令

检测策略——连续gadget识别

检测策略(1)：
连续执行的以返回指令为结尾的短指令序列不得超过一定范围

检测策略——连续gadget识别

候选gadget指令数长度阈值T0：7

候选gadget连续执行次数阈值T1：4

检测策略——连续gadget识别

指令长度阈值T0　7

Gadget次数阈值T1　4

指令计数器	1
gadget计数器	0

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret
```

```
mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall
```

检测策略——连续gadget识别

指令长度阈值T0　7

Gadget次数阈值T1　4

指令计数器	2
gadget计数器	0

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret
```

```
mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall
```

检测策略——连续gadget识别

指令长度阈值T0　7

Gadget次数阈值T1　4

指令计数器	3
gadget计数器	0

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret
```

```
mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall
```

检测策略——连续gadget识别

指令长度阈值T0　7

Gadget次数阈值T1　4

指令计数器	4
gadget计数器	0

< T0

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret

mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall
```

检测策略——连续gadget识别

指令长度阈值T0　7

Gadget次数阈值T1　4

指令计数器	0
gadget计数器	**1**

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret
```

```
mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall
```

检测策略——连续gadget识别

指令长度阈值T0　7

Gadget次数阈值T1　4

指令计数器	1
gadget计数器	1

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret
```

```
mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall
```

检测策略——连续gadget识别

指令长度阈值T0　7

Gadget次数阈值T1　4

指令计数器	2
gadget计数器	1

< T0

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret

mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall
```

检测策略——连续gadget识别

指令长度阈值T0　7

Gadget次数阈值T1　4

指令计数器	0
gadget计数器	**2**

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret

mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall
```

检测策略——连续gadget识别

指令长度阈值T0　7

Gadget次数阈值T1　4

指令计数器	1
gadget计数器	2

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret
```

```
mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall
```

检测策略——连续gadget识别

指令长度阈值T0　7

Gadget次数阈值T1　4

指令计数器	2
gadget计数器	2

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret
```

```
mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall
```

检测策略——连续gadget识别

指令长度阈值T0　7

Gadget次数阈值T1　4

指令计数器	3
gadget计数器	2

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret
```

```
mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall
```

检测策略——连续gadget识别

指令长度阈值T0　7

Gadget次数阈值T1　4

指令计数器	4
gadget计数器	2

< T0

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret
```

```
mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall
```

检测策略——连续gadget识别

指令长度阈值T0　7

Gadget次数阈值T1　4

指令计数器	0
gadget计数器	**3**

```
pop     r13
pop     r14
pop     r15
ret

                        mov     rdx,r15
                        mov     rsi,r14
                        mov     rdi,r13
pop     rdx             call    rax
pop     rsi
ret
                        pop     rax
                        ret

                        syscall
```

检测策略——连续gadget识别

指令长度阈值T0　7

Gadget次数阈值T1　4

指令计数器	1
gadget计数器	3

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret
```

```
mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall
```

检测策略——连续gadget识别

指令长度阈值T0　7

Gadget次数阈值T1　4

指令计数器	2
gadget计数器	3

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret
```

```
mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall
```

检测策略——连续gadget识别

指令长度阈值T0　7

Gadget次数阈值T1　4

指令计数器	3
gadget计数器	3

< T0

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret
```

```
mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall
```

检测策略——连续gadget识别

指令长度阈值T0　7

Gadget次数阈值T1　4

指令计数器	0
gadget计数器	**4**

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret
```

```
mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall
```

检测策略——连续gadget识别

指令长度阈值T0　7

Gadget次数阈值T1　4

指令计数器	1
gadget计数器	4

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret
```

```
mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall
```

检测策略——连续gadget识别

指令长度阈值T0　7

Gadget次数阈值T1　4

指令计数器	2
gadget计数器	4

< T0

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret
```

```
mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall
```

检测策略——连续gadget识别

指令长度阈值T0　7

Gadget次数阈值T1　4

指令计数器	0
gadget计数器	5

> T1

被攻击

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret

mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall
```

检测策略——连续gadget识别

策略缺陷：

如果攻击者使用的gadget突破阈值

...将无法检测到ROP攻击

检测策略——连续gadget识别

指令长度阈值T0　7

Gadget次数阈值T1　4

指令计数器	2
gadget计数器	3

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret
```

```
mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall
```

检测策略——连续gadget识别

指令长度阈值T0　7

Gadget次数阈值T1　4

指令计数器	3
gadget计数器	3

< T0

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret
```

```
mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall
```

检测策略——连续gadget识别

指令长度阈值T0 7

Gadget次数阈值T1 4

指令计数器	0
gadget计数器	4

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret
```

```
mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall
```

```
add     rsp,8
pop     rbx
pop     rbp
pop     r12
pop     r13
pop     r14
pop     r15
ret
```

检测策略——连续gadget识别

指令长度阈值T0 7

Gadget次数阈值T1 4

指令计数器	1
gadget计数器	4

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret

mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall

add     rsp,8
pop     rbx
pop     rbp
pop     r12
pop     r13
pop     r14
pop     r15
ret
```

检测策略——连续gadget识别

指令长度阈值T0 7

Gadget次数阈值T1 4

指令计数器	2
gadget计数器	4

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret

mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall

add     rsp,8
pop     rbx
pop     rbp
pop     r12
pop     r13
pop     r14
pop     r15
ret
```

检测策略——连续gadget识别

指令长度阈值T0　7

Gadget次数阈值T1　4

指令计数器	3
gadget计数器	4

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret

mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall

add     rsp,8
pop     rbx
pop     rbp
pop     r12
pop     r13
pop     r14
pop     r15
ret
```

检测策略——连续gadget识别

指令长度阈值T0 7

Gadget次数阈值T1 4

指令计数器	4
gadget计数器	4

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret

mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall

add     rsp,8
pop     rbx
pop     rbp
pop     r12
pop     r13
pop     r14
pop     r15
ret
```

检测策略——连续gadget识别

指令长度阈值T0 7

Gadget次数阈值T1 4

指令计数器	5
gadget计数器	4

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret

mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall

add     rsp,8
pop     rbx
pop     rbp
pop     r12
pop     r13
pop     r14
pop     r15
ret
```

检测策略——连续gadget识别

指令长度阈值T0 7

Gadget次数阈值T1 4

指令计数器	6
gadget计数器	4

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret

mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall

add     rsp,8
pop     rbx
pop     rbp
pop     r12
pop     r13
pop     r14
pop     r15
ret
```

检测策略——连续gadget识别

指令长度阈值T0 7

Gadget次数阈值T1 4

指令计数器	7
gadget计数器	4

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret

mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall

add     rsp,8
pop     rbx
pop     rbp
pop     r12
pop     r13
pop     r14
pop     r15
ret
```

检测策略——连续gadget识别

指令长度阈值T0　7

Gadget次数阈值T1　4

指令计数器	8
gadget计数器	4

> T0

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret

mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall

add     rsp,8
pop     rbx
pop     rbp
pop     r12
pop     r13
pop     r14
pop     r15
ret
```

检测策略——连续gadget识别

指令长度阈值T0 7

Gadget次数阈值T1 4

指令计数器	0
gadget计数器	0

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret

mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall

add     rsp,8
pop     rbx
pop     rbp
pop     r12
pop     r13
pop     r14
pop     r15
ret
```

检测策略——连续gadget识别

指令长度阈值T0　7

Gadget次数阈值T1　4

指令计数器	1
gadget计数器	0

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret
```

```
mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall
```

检测策略——连续gadget识别

指令长度阈值T0　7

Gadget次数阈值T1　4

指令计数器	2
gadget计数器	0

< T0

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret
```

```
mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall
```

检测策略——连续gadget识别

指令长度阈值T0　7

Gadget次数阈值T1　4

指令计数器	0
gadget计数器	**1**

```
pop     r13
pop     r14
pop     r15
ret

pop     rdx
pop     rsi
ret
```

```
mov     rdx,r15
mov     rsi,r14
mov     rdi,r13
call    rax

pop     rax
ret

syscall
```

攻击完成

检测策略——返回指令数特征

检测策略(2)：
被执行的调用指令数
不得小于返回指令数

检测策略——返回指令数特征

调用指令计数器	0
返回指令计数器	0

```
mov     rdi, 0
call    func1
nop
ret
```

```
func1:
    ...
    call    func2
    ...
    ret
```

```
func2:
    ...
    call    func3
    ...
    ret
```

```
func3:
    ...
    ...
    ret
```

检测策略——返回指令数特征

```
mov     rdi, 0
call    func1
nop
ret
```

调用指令计数器	1
返回指令计数器	0

```
func1:
    ...
    call    func2
    ...
    ret
```

```
func2:
    ...
    call    func3
    ...
    ret
```

```
func3:
    ...
    ...
    ret
```

检测策略——返回指令数特征

```
mov     rdi, 0
call    func1
nop
ret
```

调用指令计数器	1
返回指令计数器	0

```
func1:
    ...
    call    func2
    ...
    ret
```

```
func2:
    ...
    call    func3
    ...
    ret
```

```
func3:
    ...
    ...
    ret
```

检测策略——返回指令数特征

```
mov     rdi, 0
call    func1
nop
ret
```

调用指令计数器	2
返回指令计数器	0

```
func1:
    ...
    call    func2
    ...
    ret
```

```
func2:
    ...
    call    func3
    ...
    ret
```

```
func3:
    ...
    ...
    ret
```

检测策略——返回指令数特征

```
mov     rdi, 0
call    func1
nop
ret
```

调用指令计数器	2
返回指令计数器	0

```
func1:
    ...
    call    func2
    ...
    ret
```

```
func2:
    ...
    call    func3
    ...
    ret
```

```
func3:
    ...
    ...
    ret
```

检测策略——返回指令数特征

```
mov     rdi, 0
call    func1
nop
ret
```

调用指令计数器	3
返回指令计数器	0

```
func1:
    ...
    call    func2
    ...
    ret
```

```
func2:
    ...
    call    func3
    ...
    ret
```

```
func3:
    ...
    ...
    ret
```

检测策略——返回指令数特征

调用指令计数器	3
返回指令计数器	0

```
mov     rdi, 0
call    func1
nop
ret
```

```
func1:
    ...
    call    func2
    ...
    ret
```

```
func2:
    ...
    call    func3
    ...
    ret
```

```
func3:
    ...
    ...
    ret
```

检测策略——返回指令数特征

```
mov     rdi, 0
call    func1
nop
ret
```

调用指令计数器	4
返回指令计数器	0

```
func1:
    ...
    call    func2
    ...
    ret
```

```
func2:
    ...
    call    func3
    ...
    ret
```

```
func3:
    ...
    ...
    ret
```

检测策略——返回指令数特征

```
mov     rdi, 0
call    func1
nop
ret
```

调用指令计数器	4
返回指令计数器	0

```
func1:
    ...
    call    func2
    ...
    ret

func2:
    ...
    call    func3
    ...
    ret

func3:
    ...
    ...
    ret
```

检测策略——返回指令数特征

```
mov     rdi, 0
call    func1
nop
ret
```

调用指令计数器	4
返回指令计数器	1

```
func1:
    ...
    call    func2
    ...
    ret
```

```
func2:
    ...
    call    func3
    ...
    ret
```

```
func3:
    ...
    ...
    ret
```

检测策略——返回指令数特征

```
mov     rdi, 0
call    func1
nop
ret
```

调用指令计数器	4
返回指令计数器	1

```
func1:
    ...
    call    func2
    ...
    ret
```

```
func2:
    ...
    call    func3
    ...
    ret
```

```
func3:
    ...
    ...
    ret
```

检测策略——返回指令数特征

```
mov     rdi, 0
call    func1
nop
ret
```

调用指令计数器	4
返回指令计数器	2

```
func1:
    ...
    call    func2
    ...
    ret
```

```
func2:
    ...
    call    func3
    ...
    ret
```

```
func3:
    ...
    ...
    ret
```

检测策略——返回指令数特征

调用指令计数器	4
返回指令计数器	2

```
mov     rdi, 0
call    func1
nop
ret
```

```
func1:
    ...
    call    func2
    ...
    ret
```

```
func2:
    ...
    call    func3
    ...
    ret
```

```
func3:
    ...
    ...
    ret
```

检测策略——返回指令数特征

调用指令计数器	4
返回指令计数器	3

```
mov     rdi, 0
call    func1
nop
ret
```

```
func1:
    ...
    call    func2
    ...
    ret
```

```
func2:
    ...
    call    func3
    ...
    ret
```

```
func3:
    ...
    ...
    ret
```

检测策略——返回指令数特征

调用指令计数器	4
返回指令计数器	3

```
mov     rdi, 0
call    func1
nop
ret
```

```
func1:
    ...
    call    func2
    ...
    ret
```

```
func2:
    ...
    call    func3
    ...
    ret
```

```
func3:
    ...
    ...
    ret
```

检测策略——返回指令数特征

调用指令计数器	4
返回指令计数器	4

```
mov     rdi, 0
call    func1
nop
ret
```

```
func1:
    ...
    call    func2
    ...
    ret
```

```
func2:
    ...
    call    func3
    ...
    ret
```

```
func3:
    ...
    ...
    ret
```

检测策略(2)：
ROP攻击会打破调用指令数量与返回指令数量的动态平衡

检测策略——返回指令数特征

```
mov     rdi, 0
call    func1
nop
ret
```

调用指令计数器	0
返回指令计数器	0

```
func1:
    ...
    ret
```

```
gadget1:
    ...
    ret
```

```
gadget2:
    ...
    ret
```

检测策略——返回指令数特征

```
mov     rdi, 0
call    func1
nop
ret
```

调用指令计数器	1
返回指令计数器	0

```
func1:
    ...
    ret
```

```
gadget1:
    ...
    ret
```

```
gadget2:
    ...
    ret
```

检测策略——返回指令数特征

```
mov     rdi, 0
call    func1
nop
ret
```

调用指令计数器	1
返回指令计数器	0

```
func1:
    ...
    ret
```

```
gadget1:
    ...
    ret
```

```
gadget2:
    ...
    ret
```

检测策略——返回指令数特征

```
mov     rdi, 0
call    func1
nop
ret
```

调用指令计数器	2
返回指令计数器	0

```
func1:
    ...
    ret

gadget1:
    ...
    ret

gadget2:
    ...
    ret
```

检测策略——返回指令数特征

```
mov     rdi, 0
call    func1
nop
ret
```

调用指令计数器	2
返回指令计数器	0

程序控制流被劫持

```
func1:
  ...
  ret

gadget1:
  ...
  ret

gadget2:
  ...
  ret
```

检测策略——返回指令数特征

调用指令计数器	2
返回指令计数器	1

```
mov     rdi, 0
call    func1
nop
ret
```

```
func1:
    ...
    ret
```

```
gadget1:
    ...
    ret
```

```
gadget2:
    ...
    ret
```

检测策略——返回指令数特征

```
mov     rdi, 0
call    func1
nop
ret
```

调用指令计数器	2
返回指令计数器	1

```
func1:
    ...
    ret
```

```
gadget1:
    ...
    ret
```

```
gadget2:
    ...
    ret
```

检测策略——返回指令数特征

```
mov     rdi, 0
call    func1
nop
ret
```

调用指令计数器	2
返回指令计数器	2

```
func1:
    ...
    ret
```

```
gadget1:
    ...
    ret
```

```
gadget2:
    ...
    ret
```

检测策略——返回指令数特征

```
mov     rdi, 0
call    func1
nop
ret
```

调用指令计数器	2
返回指令计数器	2

```
func1:
    ...
    ret
```

```
gadget1:
    ...
    ret
```

```
gadget2:
    ...
    ret
```

检测策略——返回指令数特征

```
mov     rdi, 0
call    func1
nop
ret
```

调用指令计数器	2
返回指令计数器	3

被攻击

```
func1:
    ...
    ret
```

```
gadget1:
    ...
    ret
```

```
gadget2:
    ...
    ret
```

检测策略——返回指令数特征

策略缺陷：
程序控制流被劫持时，已执行的调用指令数量比返回指令数量多

检测策略——小结

基于ROP攻击指令特征：

检测策略(1)——阈值识别连续gadget

检测策略(2)——调用返回指令数对比

其他特征？

检测策略——控制流特征

如果能够发现程序控制流的异常

...就能检测出ROP攻击

检测策略——控制流特征

劫持控制流的方式：

1.篡改返回地址

2.修改函数指针

检测策略——检查返回地址完整性

检测策略(3)：

被调函数的返回地址保存在栈中

...值为调用指令的下条指令地址

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
    push    rbp
    mov     rbp, rsp
    sub     rsp, 0x80
    lea     rax, [rbp-0x80]
    mov     edx, 0x200
    mov     rsi, rax
    mov     edi, 0
    call    read
    leave
    ret
```

栈

...
...
...
...
...

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
    push    rbp
    mov     rbp, rsp
    sub     rsp, 0x80
    lea     rax, [rbp-0x80]
    mov     edx, 0x200
    mov     rsi, rax
    mov     edi, 0
    call    read
    leave
    ret
```

栈

...
...
...
...
...

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
    push    rbp
    mov     rbp, rsp
    sub     rsp, 0x80
    lea     rax, [rbp-0x80]
    mov     edx, 0x200
    mov     rsi, rax
    mov     edi, 0
    call    read
    leave
    ret
```

栈

…
…
…
…
…

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
    push    rbp
    mov     rbp, rsp
    sub     rsp, 0x80
    lea     rax, [rbp-0x80]
    mov     edx, 0x200
    mov     rsi, rax
    mov     edi, 0
    call    read
    leave
    ret
```

栈

…
…
…
…
…

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
push    rbp
mov     rbp, rsp
sub     rsp, 0x80
lea     rax, [rbp-0x80]
mov     edx, 0x200
mov     rsi, rax
mov     edi, 0
call    read
leave
ret
```

栈

…
…
…
…
…

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
  push    rbp
  mov     rbp, rsp
  sub     rsp, 0x80
  lea     rax, [rbp-0x80]
  mov     edx, 0x200
  mov     rsi, rax
  mov     edi, 0
  call    read
  leave
  ret
```

栈

返回地址
…
…
…
…

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
  push    rbp
  mov     rbp, rsp
  sub     rsp, 0x80
  lea     rax, [rbp-0x80]
  mov     edx, 0x200
  mov     rsi, rax
  mov     edi, 0
  call    read
  leave
  ret
```

栈

旧rbp
返回地址
...
...
...

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
  push    rbp
  mov     rbp, rsp
  sub     rsp, 0x80
  lea     rax, [rbp-0x80]
  mov     edx, 0x200
  mov     rsi, rax
  mov     edi, 0
  call    read
  leave
  ret
```

栈

旧rbp
返回地址
...
...
...

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
    push    rbp
    mov     rbp, rsp
    sub     rsp, 0x80
    lea     rax, [rbp-0x80]
    mov     edx, 0x200
    mov     rsi, rax
    mov     edi, 0
    call    read
    leave
    ret
```

栈

...
...
旧rbp
返回地址
...

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
    push    rbp
    mov     rbp, rsp
    sub     rsp, 0x80
    lea     rax, [rbp-0x80]
    mov     edx, 0x200
    mov     rsi, rax
    mov     edi, 0
    call    read
    leave
    ret
```

栈

...
buf[128]
旧rbp
返回地址
...

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
    push    rbp
    mov     rbp, rsp
    sub     rsp, 0x80
    lea     rax, [rbp-0x80]
    mov     edx, 0x200
    mov     rsi, rax
    mov     edi, 0
    call    read
    leave
    ret
```

栈

...
buf[128]
旧rbp
返回地址
...

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
    push    rbp
    mov     rbp, rsp
    sub     rsp, 0x80
    lea     rax, [rbp-0x80]
    mov     edx, 0x200
    mov     rsi, rax
    mov     edi, 0
    call    read
    leave
    ret
```

栈

...
buf[128]
旧rbp
返回地址
...

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
    push    rbp
    mov     rbp, rsp
    sub     rsp, 0x80
    lea     rax, [rbp-0x80]
    mov     edx, 0x200
    mov     rsi, rax
    mov     edi, 0
    call    read
    leave
    ret
```

栈

...
buf[128]
旧rbp
返回地址
...

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
    push    rbp
    mov     rbp, rsp
    sub     rsp, 0x80
    lea     rax, [rbp-0x80]
    mov     edx, 0x200
    mov     rsi, rax
    mov     edi, 0
    call    read
    leave
    ret
```

栈

...
buf[128]
旧rbp
返回地址
...

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
    push    rbp
    mov     rbp, rsp
    sub     rsp, 0x80
    lea     rax, [rbp-0x80]
    mov     edx, 0x200
    mov     rsi, rax
    mov     edi, 0
    call    read
    leave
    ret
```

栈

返回地址
…
…
…
…

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad63
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

0x400567：

先前调用返回地址

```
sub     rsp, 0x80
lea     rax, [rbp-0x80]
mov     edx, 0x200
mov     rsi, rax
mov     edi, 0
call    read
leave
ret
```

栈

返回地址
…
…
…
…

检测策略——检查返回地址完整性

检测策略(3)：

调用时事先保存先前调用返回地址

返回时检查返回地址与之是否一致

检测策略(3)：
构造影子栈实现

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
    push    rbp
    mov     rbp, rsp
    sub     rsp, 0x80
    lea     rax, [rbp-0x80]
    mov     edx, 0x200
    mov     rsi, rax
    mov     edi, 0
    call    read
    leave
    ret
```

栈

...
...
...
...
...

影子栈

...
...
...
...
...

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
    push    rbp
    mov     rbp, rsp
    sub     rsp, 0x80
    lea     rax, [rbp-0x80]
    mov     edx, 0x200
    mov     rsi, rax
    mov     edi, 0
    call    read
    leave
    ret
```

栈

...
...
...
...
...

影子栈

...
...
...
...
...

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
    push    rbp
    mov     rbp, rsp
    sub     rsp, 0x80
    lea     rax, [rbp-0x80]
    mov     edx, 0x200
    mov     rsi, rax
    mov     edi, 0
    call    read
    leave
    ret
```

栈

...
...
...
...
...

影子栈

...
...
...
...
...

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
    push    rbp
    mov     rbp, rsp
    sub     rsp, 0x80
    lea     rax, [rbp-0x80]
    mov     edx, 0x200
    mov     rsi, rax
    mov     edi, 0
    call    read
    leave
    ret
```

栈

...
...
...
...
...

影子栈

...
...
...
...
...

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
push    rbp
mov     rbp, rsp
sub     rsp, 0x80
lea     rax, [rbp-0x80]
mov     edx, 0x200
mov     rsi, rax
mov     edi, 0
call    read
leave
ret
```

栈

...
...
...
...
...

影子栈

...
...
...
...
...

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
    push    rbp
    mov     rbp, rsp
    sub     rsp, 0x80
    lea     rax, [rbp-0x80]
    mov     edx, 0x200
    mov     rsi, rax
    mov     edi, 0
    call    read
    leave
    ret
```

栈

返回地址
…
…
…
…

影子栈

返回地址
…
…
…
…

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
    push    rbp
    mov     rbp, rsp
    sub     rsp, 0x80
    lea     rax, [rbp-0x80]
    mov     edx, 0x200
    mov     rsi, rax
    mov     edi, 0
    call    read
    leave
    ret
```

栈

返回地址
...
...
...
...

影子栈

返回地址
...
...
...
...

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
  push    rbp
  mov     rbp, rsp
  sub     rsp, 0x80
  lea     rax, [rbp-0x80]
  mov     edx, 0x200
  mov     rsi, rax
  mov     edi, 0
  call    read
  leave
  ret
```

栈

旧rbp
返回地址
...
...
...

影子栈

返回地址
...
...
...
...

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
    push    rbp
    mov     rbp, rsp
    sub     rsp, 0x80
    lea     rax, [rbp-0x80]
    mov     edx, 0x200
    mov     rsi, rax
    mov     edi, 0
    call    read
    leave
    ret
```

栈

旧rbp
返回地址
...
...
...

影子栈

返回地址
...
...
...
...

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
    push    rbp
    mov     rbp, rsp
    sub     rsp, 0x80
    lea     rax, [rbp-0x80]
    mov     edx, 0x200
    mov     rsi, rax
    mov     edi, 0
    call    read
    leave
    ret
```

栈

...
...
旧rbp
返回地址
...

影子栈

返回地址
...
...
...
...

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
    push    rbp
    mov     rbp, rsp
    sub     rsp, 0x80
    lea     rax, [rbp-0x80]
    mov     edx, 0x200
    mov     rsi, rax
    mov     edi, 0
    call    read
    leave
    ret
```

栈

...
buf[128]
旧rbp
返回地址
...

影子栈

返回地址
...
...
...
...

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
    push    rbp
    mov     rbp, rsp
    sub     rsp, 0x80
    lea     rax, [rbp-0x80]
    mov     edx, 0x200
    mov     rsi, rax
    mov     edi, 0
    call    read
    leave
    ret
```

栈

...
buf[128]
旧rbp
返回地址
...

影子栈

返回地址
...
...
...
...

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
    push    rbp
    mov     rbp, rsp
    sub     rsp, 0x80
    lea     rax, [rbp-0x80]
    mov     edx, 0x200
    mov     rsi, rax
    mov     edi, 0
    call    read
    leave
    ret
```

栈

...
buf[128]
旧rbp
返回地址
...

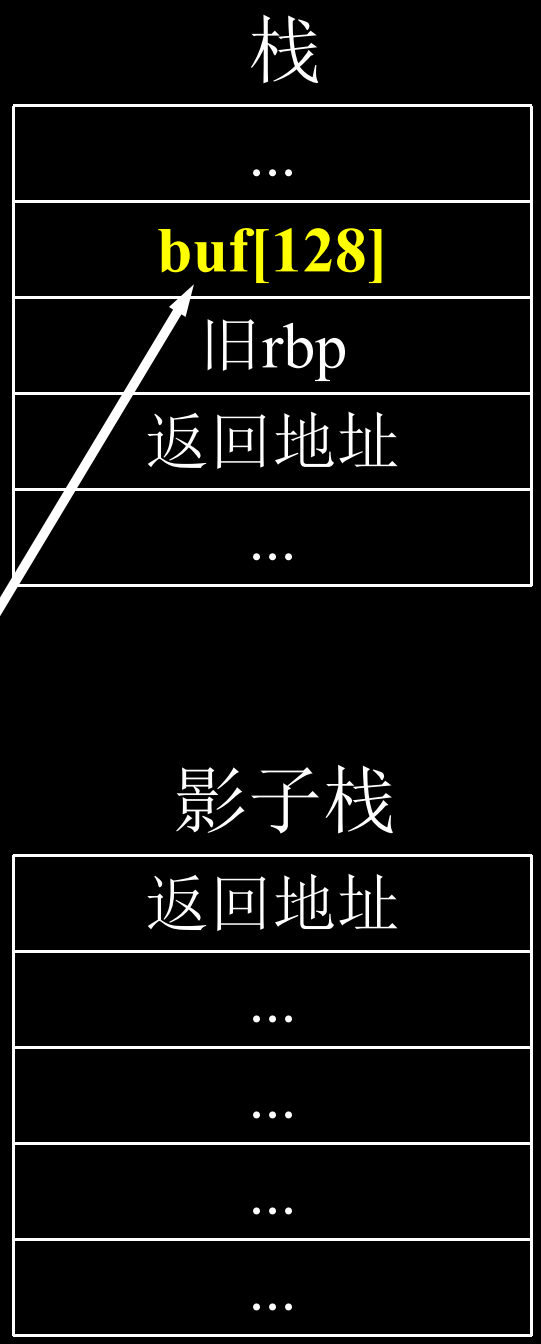
影子栈

返回地址
...
...
...
...

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
    push    rbp
    mov     rbp, rsp
    sub     rsp, 0x80
    lea     rax, [rbp-0x80]
    mov     edx, 0x200
    mov     rsi, rax
    mov     edi, 0
    call    read
    leave
    ret
```



检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
    push    rbp
    mov     rbp, rsp
    sub     rsp, 0x80
    lea     rax, [rbp-0x80]
    mov     edx, 0x200
    mov     rsi, rax
    mov     edi, 0
    call    read
    leave
    ret
```

栈

...
buf[128]
(被覆盖)
Gadget地址
(恶意数据...)

影子栈

返回地址
...
...
...
...

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
    push    rbp
    mov     rbp, rsp
    sub     rsp, 0x80
    lea     rax, [rbp-0x80]
    mov     edx, 0x200
    mov     rsi, rax
    mov     edi, 0
    call    read
    leave
    ret
```

栈

Gadget地址
(恶意数据)
...
...
...

影子栈

返回地址
...
...
...
...

检测策略——检查返回地址完整性

```
nop
mov     edx,0x
mov     esi,0x4
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
Gadget:
    mov     eax,0
    ret
```

```
67:
    rbp
    rbp, rsp
    sub     rsp, 0x80
    lea     rax, [rbp-0x80]
    mov     edx, 0x200
    mov     rsi, rax
    mov     edi, 0
    call    read
    leave
    ret
```

栈

Gadget地址
(恶意数据)
...
...
...

影子栈

返回地址
...
...
...
...

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

不一致

```
0x400567:
        rbp
        rbp, rsp
sub     rsp, 0x80
lea     rax, [rbp-0x80]
mov     edx, 0x200
mov     rsi, rax
mov     edi, 0
call    read
leave
ret
```

栈

Gadget地址
(恶意数据)
...
...
...

影子栈

返回地址
...
...
...
...

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add
mov
ret
```

被攻击

不一致

```
0x400567:
                rbp
                rbp, rsp
sub             rsp, 0x80
lea             rax, [rbp-0x80]
mov             edx, 0x200
mov             rsi, rax
mov             edi, 0
call            read
leave
ret
```

栈

Gadget地址
(恶意数据)
...
...
...

影子栈

返回地址
...
...
...
...

检测策略——检查返回地址完整性

策略缺陷：

大量库函数使用setjmp/longjmp

破坏了影子栈中调用/返回一一对应的关系

检测策略(4)：
建立先前调用返回地址表
函数返回地址不得在表外

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad63
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

0x400567：

先前调用返回地址

```
sub     rsp, 0x80
lea     rax, [rbp-0x80]
mov     edx, 0x200
mov     rsi, rax
mov     edi, 0
call    read
leave
ret
```

栈

返回地址
…
…
…
…

检测策略——检查返回地址完整性

```
nop
mov     edx,0x354
mov     esi,0x4ad632
mov     eax,0
call    0x400567
mov     eax,0
add     rsb, 0x8
mov     rsp,rbp
ret
```

```
0x400567:
    push    rbp
    mov     rbp, rsp
                        80]
    mov     edx, 0x200
    mov     rsi, rax
    mov     edi, 0
    call    read
    leave
    ret
```

非先前调用返回地址

栈

异常地址
…
…
…
…

检测策略——检查返回地址完整性

策略缺陷：
无法检测不依赖栈的JOP攻击

跳转导向编程(JOP)

dispatcher:

```
add edx,4
jmp [edx]
```

Dispatch Table

loader
adder
storer
...
...

loader:

```
mov eax,[eax]
jmp esi
```

adder:

```
add  eax,[ebx]
jmp  [edi]
```

storer:

```
add  [ecx],eax
jmp  [edi]
```

跳转导向编程(JOP)

```
dispatcher:
    add edx,4
    jmp [edx]
```

Dispatch Table

loader
adder
storer
...
...

```
loader:
    mov eax,[eax]
    jmp esi

adder:
    add  eax,[ebx]
    jmp  [edi]

storer:
    add  [ecx],eax
    jmp  [edi]
```

跳转导向编程(JOP)

```
dispatcher:
    add edx,4
    jmp [edx]
```

Dispatch Table

loader
adder
storer
...
...

```
loader:
    mov eax,[eax]
    jmp esi

adder:
    add  eax,[ebx]
    jmp  [edi]

storer:
    add  [ecx],eax
    jmp  [edi]
```

跳转导向编程(JOP)

dispatcher:
```
add edx,4
jmp [edx]
```

Dispatch Table

loader
adder
storer
...
...

loader:
```
mov eax,[eax]
jmp esi
```

adder:
```
add  eax,[ebx]
jmp  [edi]
```

storer:
```
add  [ecx],eax
jmp  [edi]
```

跳转导向编程(JOP)

dispatcher:

```
add edx,4
jmp [edx]
```

Dispatch Table

loader
adder
storer
...
...

loader:

```
mov eax,[eax]
jmp esi
```

adder:

```
add  eax,[ebx]
jmp  [edi]
```

storer:

```
add  [ecx],eax
jmp  [edi]
```

跳转导向编程(JOP)

```
dispatcher:
    add edx,4
    jmp [edx]
```

Dispatch Table

loader
adder
storer
...
...

```
loader:
    mov eax,[eax]
    jmp esi

adder:
    add  eax,[ebx]
    jmp  [edi]

storer:
    add  [ecx],eax
    jmp  [edi]
```

跳转导向编程(JOP)

```
dispatcher:
    add edx,4
    jmp [edx]
```

Dispatch Table

loader
adder
storer
...
...

```
loader:
    mov eax,[eax]
    jmp esi

adder:
    add  eax,[ebx]
    jmp  [edi]

storer:
    add  [ecx],eax
    jmp  [edi]
```

跳转导向编程(JOP)

```
dispatcher:
    add edx,4
    jmp [edx]
```

Dispatch Table

loader
adder
storer
...
...

```
loader:
    mov eax,[eax]
    jmp esi

adder:
    add  eax,[ebx]
    jmp  [edi]

storer:
    add  [ecx],eax
    jmp  [edi]
```

跳转导向编程(JOP)

dispatcher:

```
add edx,4
jmp [edx]
```

Dispatch Table

loader
adder
storer
...
...

loader:

```
mov eax,[eax]
jmp esi
```

adder:

```
add  eax,[ebx]
jmp  [edi]
```

storer:

```
add  [ecx],eax
jmp  [edi]
```

跳转导向编程(JOP)

```
dispatcher:
    add edx,4
    jmp [edx]
```

Dispatch Table

loader
adder
storer
...
...

```
loader:
    mov eax,[eax]
    jmp esi

adder:
    add  eax,[ebx]
    jmp  [edi]

storer:
    add  [ecx],eax
    jmp  [edi]
```

跳转导向编程(JOP)

```
dispatcher:
    add edx,4
    jmp [edx]
```

Dispatch Table

loader
adder
storer
...
...

```
loader:
    mov eax,[eax]
    jmp esi

adder:
    add  eax,[ebx]
    jmp  [edi]

storer:
    add  [ecx],eax
    jmp  [edi]
```

跳转导向编程(JOP)

```
dispatcher:
    add edx,4
    jmp [edx]
```

Dispatch Table

loader
adder
storer
..
...

```
loader:
    mov eax,[eax]
    jmp esi

adder:
    add  eax,[ebx]
    jmp  [edi]

storer:
    add  [ecx],eax
    jmp  [edi]
```

跳转导向编程(JOP)

dispatcher:

```
add edx,4
jmp [edx]
```

Dispatch Table

loader
adder
storer
...
...

loader:

```
mov eax,[eax]
jmp esi
```

adder:

```
add  eax,[ebx]
jmp  [edi]
```

storer:

```
add  [ecx],eax
jmp  [edi]
```

跳转导向编程(JOP)

```
dispatcher:
    add edx,4
    jmp [edx]
```

Dispatch Table

loader
adder
storer
...
...

```
loader:
    mov eax,[eax]
    jmp esi

adder:
    add  eax,[ebx]
    jmp  [edi]

storer:
    add  [ecx],eax
    jmp  [edi]
```

检测策略——检查函数指针完整性

不依赖栈的控制流劫持：

1.修改函数指针

2.覆写setjmp缓冲区

检测策略——检查函数指针完整性

检测策略(5)：

如果函数指针的值不合法

...则说明程序受到攻击

检测策略——检查函数指针完整性

```
lea     rdi, [rip + 0xdf]
call    0x4004a0
mov     eax, 0
leave
ret
```

```
0x4004a0 <puts@plt>:
    jmp    [rip+0x200b72]
    push   0x0
    jmp    0x400490
```

检测策略——检查函数指针完整性

```
lea     rdi, [rip + 0xdf]
call    0x4004a0
mov     eax, 0
leave
ret
```

```
0x4004a0 <puts@plt>:
    jmp   [rip+0x200b72]
    push  0x0
    jmp   0x400490
```

检测策略——检查函数指针完整性

lea rdi, [rip + 0xdf]
call
mov
leave
ret

函数指针

GOT表

...	...
...	...
0x601018	puts地址
...	...
...	...

```
0x4004a0 <puts@plt>:
  jmp   [rip+0x200b72]
  push  0x0
  jmp   0x400490
```

检测策略——检查函数指针完整性

```
lea     rdi, [rip + 0xdf]
call    0x4004a0
mov     eax, 0
leave
ret
```

GOT表

...	...
...	...
0x601018	puts地址
...	...
...	...

正常转移

```
0x4004a0 <puts@plt>:
    jmp   [rip+0x200b72]
    push  0x0
    jmp   0x400490
```

```
puts:
    push   r14
    push   r13
        ...
    ret
```

检测策略——检查函数指针完整性

```
lea     rdi, [rip + 0xdf]
call    0x4004a0
mov     eax, 0
leave
ret
```

```
0x4004a0 <puts@plt>:
  jmp   [rip+0x200b72]
  push  0x0
  jmp   0x400490
```

GOT表

…	…
…	…
0x601018	**Gadget地址**
…	…
…	…

检测策略——检查函数指针完整性

```
lea     rdi, [rip + 0xdf]
call    0x4004a0
mov     eax, 0
leave
ret
```

GOT表

...	...
...	...
0x601018	**Gadget地址**
...	...
...	...

异常转移

```
0x4004a0 <puts@plt>:
    jmp   [rip+0x200b72]
    push  0x0
    jmp   0x400490
```

```
gadget:
    mov   eax,0
    mov   ebx,0
          ...
    ret
```

检测策略——检查函数指针完整性

策略缺陷：
难以监测程序中所有的函数指针

检测策略——小结

基于ROP攻击指令特征：

检测策略(1)——阈值识别连续gadget

检测策略(2)——调用返回指令数对比

基于ROP攻击控制流特征：

检测策略(3)——影子栈验证返回地址

检测策略(4)——查表验证返回地址

检测策略(5)——验证GOT指针合法性

检测策略(6)：
应用多维度的检测策略
将极大地提高攻击难度

检测策略——多维度综合检测策略

必须使用先前调用返回gadget

检测策略——多维度综合检测策略

必须使用先前调用返回gadget

必须使用长gadget突破阈值

检测策略——多维度综合检测策略

必须使用先前调用返回gadget

必须使用长gadget突破阈值

恶意代码中返回指令不能过多

检测策略——多维度综合检测策略

必须使用先前调用返回gadget

必须使用长gadget突破阈值

恶意代码中返回指令不能过多

无法修改GOT表

检测策略——多维度综合检测策略

可检测的攻击类型：

- Return-into-libc攻击
- ROP攻击
- 使用长gadget的ROP攻击
- 部分JOP攻击

测试界面

视频展示

谢谢！

答辩人：刘天祺　　指导教师：詹静