

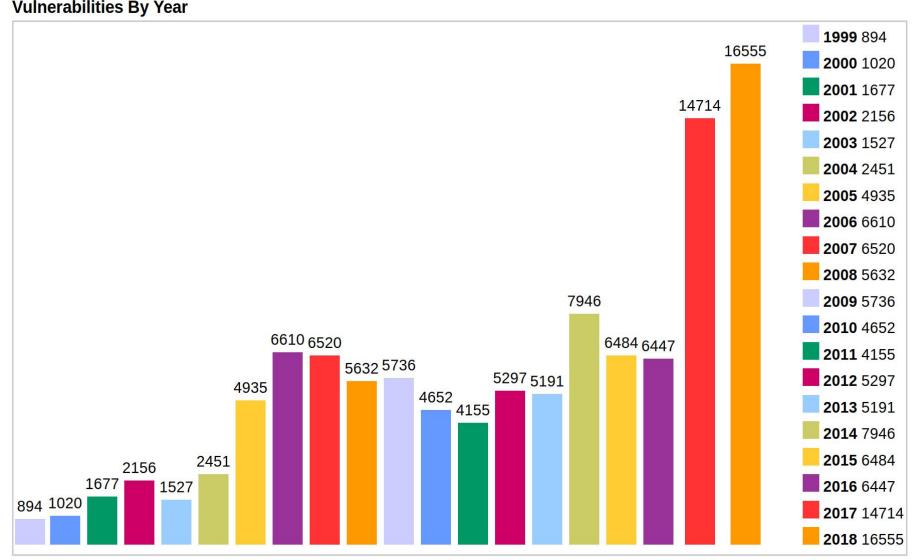
基于二进制动态翻译的 ROP攻击检测方法研究与实现

❸ 答辩人: 刘天祺 ❷ 指导教师: 詹静



背景: 近20年CVE漏洞数量

Vulnerabilities By Year



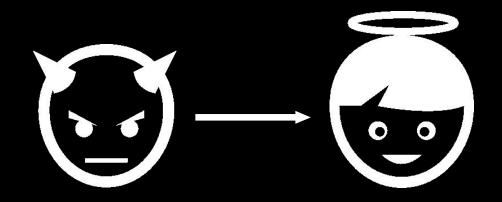


代码注入攻击



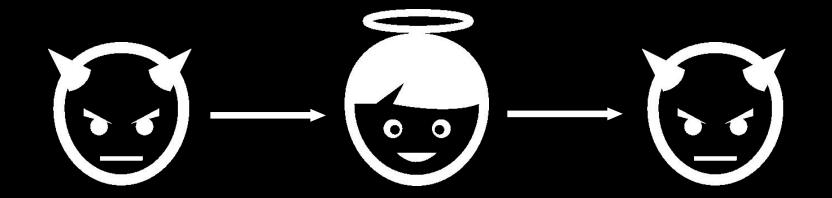


代码注入攻击 数据执行保护



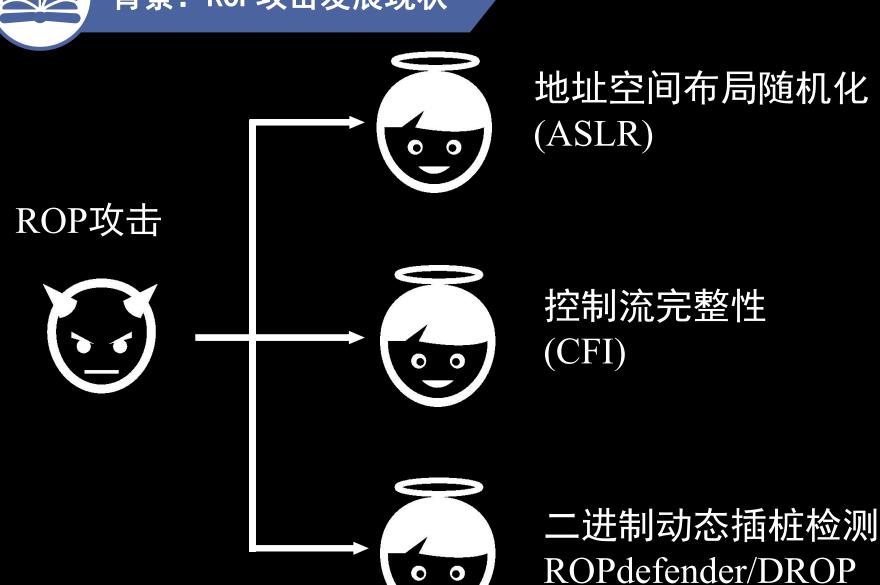


代码注入攻击 数据执行保护 代码复用攻击



Return-into-libc攻击 ROP攻击







sub rsp,0x8	pop rbp	test rbp,rbp
mov rax, [rip+0x200bb5]	mov edi,0x601038	je 400686
test rax,rax	jmp rax	xor ebx,ebx
je $40044a < init + 0x12 >$	nop DWORD PTR [rax]	nop
call rax	pop rbp	mov rdx,r15
add rsp,0x8	ret	mov rsi,r14
ret	push r15	mov edi,r13d
mov esi,0x601038	push r14	call [r12+rbx*8]
push rbp	mov r15,rdx	add rbx,0x1
sub rsi,0x601038	push r13	cmp rbp,rbx
mov rbp,rsp	push r12	jne 400670
sar rsi,0x3	lea r12,[rip+0x2007c6]	add rsp,0x8
mov rax,rsi	push rbp	pop rbx
shr rax,0x3f	lea rbp,[rip+0x2007c6]	pop rbp
add rsi,rax	push rbx	pop r12
sar rsi,1	mov r13d,edi	pop r13
je 400528	mov r14,rsi	pop r14
mov eax,0x0	sub rbp,r12	pop r15
test rax,rax	sub rsp,0x8	ret
je 400528	sar rbp,0x3	nop



sub rsp,0x8	pop rbp	test rbp,rbp
mov rax, [rip+0x200bb5]	mov edi,0x601038	je 400686
test rax,rax	jmp rax	xor ebx,ebx
je $40044a < init + 0x12 >$	nop DWORD PTR [rax]	nop
call rax	pop rbp	mov rdx,r15
add rsp,0x8	ret	mov rsi,r14
ret	push r15	mov edi,r13d
mov esi,0x601038	push r14	call [r12+rbx*8]
push rbp	mov r15,rdx	add rbx,0x1
sub rsi,0x601038	push r13	cmp rbp,rbx
mov rbp,rsp	push r12	jne 400670
sar rsi,0x3	lea r12,[rip+0x2007c6]	add rsp,0x8
mov rax,rsi	push rbp	pop rbx
shr rax,0x3f	lea rbp,[rip+0x2007c6]	pop rbp
add rsi,rax	push rbx	pop r12
sar rsi,1	mov r13d,edi	pop r13
je 400528	mov r14,rsi	pop r14
mov eax,0x0	sub rbp,r12	pop r15
test rax,rax	sub rsp,0x8	ret
je 400528	sar rbp,0x3	nop



```
rbp,rbp
sub
     rsp,0x8
                                                               test
                                      rbp
                                pop
                                                                    400686
                                                               je
      rax, [rip+0x200bb5]
                                      edi,0x601038
                                mov
mov
                                                                     ebx,ebx
     rax,rax
                                jmp
                                                               xor
test
                                      rax
je
    40044a < init+0x12 >
                                      DWORD PTR [rax]
                                                               nop
                                nop
                                                                      rdx,r15
call
                                                               mov
     rax
                                     rbp
                                pop
                                                                      rsi,r14
add
     rsp,0x8
                                                               mov
                                ret
                                                                      edi,r13d
                                                               mov
                                push
                                     r15
ret
                                                                     [r12+rbx*8]
                                                               call
                                push
                                     r14
mov
      esi,0x601038
                                                               add
                                                                     rbx,0x1
push
     rbp
                                      r15,rdx
                                mov
                                                                     rbp,rbx
     rsi,0x601038
                                push
                                     r13
                                                               cmp
sub
                                                                    400670
                                                               jne
                                push r12
      rbp,rsp
mov
                                                                     rsp,0x8
                                                               add
     rsi,0x3
                                              0x2007c6
sar
                             Gadget
                                                                     rbx
      rax,rsi
                                                               pop
mov
                                                                     rbp
shr
     rax,0x3f
                                              0x2007c6
                                                               pop
                                                                     r12
add
     rsi,rax
                                push
                                      rbx
                                                               pop
                                                                     r13
                                      r13d,edi
     rsi,1
                                                               pop
sar
                                mov
                                                                     r14
    400528
                                      r14,rsi
                                                               pop
je
                                mov
                                                                     r15
      eax,0x0
                                     rbp,r12
                                sub
                                                               pop
mov
                                                               ret
                                sub
                                     rsp,0x8
     rax,rax
test
    400528
                                     rbp,0x3
je
                                                               nop
                                sar
```



```
变种Gadget
sub
     rsp,0x8
                                     rbp
                               pop
      rax, [rip+0x200bb5]
                                      edi,0x601050
mov
                               mov
                                                                    ebx,ebx
                                                              XON
     rax,rax
                               jmp
test
                                     rax
je
     40044a < init+0x12 > 
                                     DWORD PTR [rax]
                                                              nop
                               nop
                                                                     rdx,r15
call
                                                              mov
                                     rbp
     rax
                               pop
                                                                     rsi,r14
add
     rsp,0x8
                                                              mov
                               ret
                                                                     edi,r13d
                                                              mov
ret
                               push
                                     r15
                                                              call
                                                                    [r12+rbx*8]
                               push
                                     r14
mov
      esi,0x601038
                                                              add
                                                                    rbx,0x1
push
                                      r15,rdx
     rbp
                               mov
                                                                     rbp,rbx
     rsi,0x601038
                               push
                                     r13
                                                              cmp
sub
                                                                    400670
                                                              jne
                               push r12
      rbp,rsp
mov
                                                                    rsp,0x8
                                                              add
     rsi,0x3
                                              0x2007c6
sar
                             Gadget
                                                                    rbx
      rax,rsi
                                                              pop
mov
                                                                    rbp
shr
     rax,0x3f
                                              0x2007c6
                                                              pop
                                                                    r12
add
     rsi,rax
                               push
                                     rbx
                                                              pop
                                                                    r13
                                      r13d,edi
     rsi,1
                                                              pop
sar
                               mov
                                                                    r14
    400528
                                      r14,rsi
                                                              pop
je
                               mov
                                                                    r15
      eax,0x0
                               sub
                                     rbp,r12
                                                              pop
mov
                                                              ret
                               sub
                                     rsp,0x8
     rax,rax
test
     400528
                                    rbp,0x3
je
                                                              nop
                               sar
```

如果能够识别出攻击代码的特征 …就能检测出ROP攻击

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(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

指令长度阈值T0 7 Gadget长度阈值T1 4



指令长度阈值T0

Gadget长度阈值T1

pop	r13	
pop	r14	
pop	r15	
ret		

ga	ldge	t计	数器	1
1	1 =			

指令计数器

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

pop rax ret



指令长度阈值T0

Gadget长度阈值T1

pop	r13	
pop	r14	
pop	r15	
ret		

指令计数器	2
gadget计数器	0

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

pop rax ret



检测策略一 一连续gadget识别

指令长度阈值T0

Gadget长度阈值T1

r13 pop r14 pop r15 pop ret

rdx pop rsi pop

ret

指令计数器	3
gadget计数器	0

rdx,r15 mov

rsi,r14 mov

rdi,r13 mov

call rax

pop rax

ret



指令长度阈值T0

Gadget长度阈值T1

< T0

pop r13pop r14pop r15

ret

pop rdx pop rsi ret

指令计数器	4
gadget计数器	0

mov rdx,r15

mov rsi,r14

mov rdi,r13

call rax

pop rax

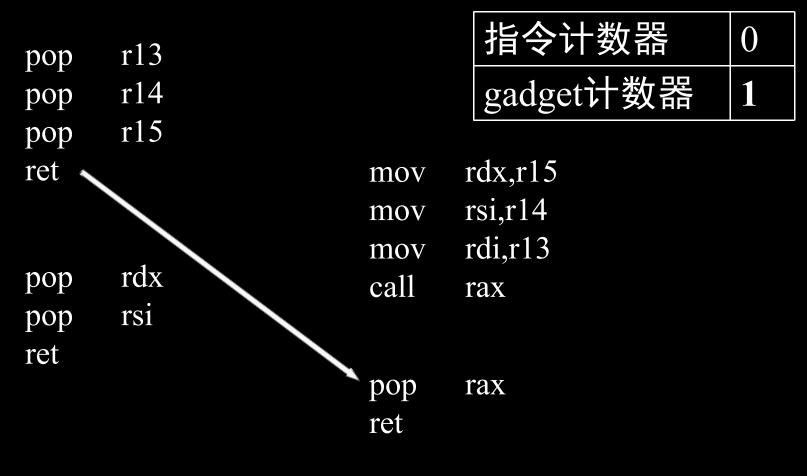
ret



检测策略一 一连续gadget识别

指令长度阈值T0

Gadget长度阈值T1





指令长度阈值T0

Gadget长度阈值T1

pop r13 pop r14 pop r15

ret

pop rdx pop rsi ret

指令计数器	1
gadget计数器	1

mov rdx,r15

mov rsi,r14

mov rdi,r13

call rax

pop rax

ret



指令长度阈值T0

Gadget长度阈值T1

< T0

pop	r13
pop	r14
pop	r15
ret	

mov	rdx.r	15
mov	IUA.I	LO

指令计数器

gadget计数器

mov rsi,r14

mov rdi,r13

call rax

pop rdx pop rsi

ret

pop	rax	
ret		



指令长度阈值T0

Gadget长度阈值T1

4

pop	r13
pop	r14
pop	r15

ret

pop rdx pop rsi ret

指令计数器	0
gadget计数器	2

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

pop rax ret



指令长度阈值T0

Gadget长度阈值T1

pop	r13
pop	r14
pop	r15
4	

mov rdx,r15

指令计数器

gadget计数器

ret

mov rsi,r14

pop rdx pop rsi mov rdi,r13

ret

call rax

pop rax

syscall

ret



指令长度阈值T0

Gadget长度阈值T1

pop r13 pop r14 pop r15 ret

指令计数器	2
gadget计数器	2

pop rdx pop rsi ret mov rdx,r15 mov rsi,r14

mov rdi,r13

call rax

pop rax

ret



检测策略─ 一连续gadget识别

指令长度阈值T0

Gadget长度阈值T1

pop	r13
pop	r14
pop	r15
ret	

指令计数器	3
gadget计数器	2

rdx pop rsi pop ret

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

pop rax ret



指令长度阈值T0

Gadget长度阈值T1_

4

T0

pop r13pop r14pop r15

ret

pop rdx pop rsi ret

指令计数器	4	<
gadget计数器	2	

mov rdx,r15

mov rsi,r14

mov rdi,r13

call rax

pop rax ret



检测策略─ 一连续gadget识别

指令长度阈值T0

Gadget长度阈值T1

0

3

pop	r13		指令计数器
pop	r14		gadget计数器
pop	r15		
ret		mov	rdx,r15
		mov	rsi,r14
	1	mov	rdi,r13
pop	rdx ←	call	rax
pop	rsi		
ret			
		pop	rax
		ret	



指令长度阈值T0

Gadget长度阈值T1

pop	r13
pop	r14
pop	r15
ret	

指令计数器	1
gadget计数器	3

pop	rdx	
pop	rsi	
ret		

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

pop rax ret



指令长度阈值T0

Gadget长度阈值T1

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



指令长度阈值T0

Gadget长度阈值T1 4

pop r13 pop r14 pop r15

ret

pop rdx pop rsi

ret

指令计数器	3	< T
gadget计数器	3	

mov rdx,r15

mov rsi,r14

mov rdi,r13

call rax

pop rax

ret



检测策略一 -连续gadget识别

指令长度阈值T0

Gadget长度阈值T1

4

自令计数器 0 adget计数器 4

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



指令长度阈值T0

Gadget长度阈值T1

pop r13pop r14pop r15

ret

pop rdx pop rsi ret

指令计数器	1
gadget计数器	4

mov rdx,r15

mov rsi,r14

mov rdi,r13

call rax

pop rax

ret



指令长度阈值T0

Gadget长度阈值T1

< T0

pop r13 pop r14 pop r15 ret 指令计数器 2 gadget计数器 4

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

pop rax ret



指令长度阈值T0

Gadget长度阈值T1

pop r13 pop r14 pop r15 ret

pop rdx pop rsi ret

指令计数器	0	
gadget计数器	5	>T1

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

被攻击

pop rax ret syscall

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



指令长度阈值T0

Gadget长度阈值T1

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



检测策略一 一连续gadget识别

指令长度阈值T0

Gadget长度阈值T1

指令计数器 < T0 3 gadget计数器 3

r13 pop r14 pop r15 pop ret

rdx pop rsi pop

ret

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

rax

pop rax ret



指令长度阈值T0

Gadget长度阈值T1

pop	r13		指令计	数器	0
pop	r14		gadgeti	计数器	4
pop	r15				
ret		mov	rdx,r15	add	rsp,8
		mov	rsi r14	pop	rbx
	1	mov	rdi,r13	pop	rbp
pop	rdx	call	rax	pop	r12
pop	rsi			pop	r13
ret		202	POV	pop	r14
		pop	rax	pop	r15
		ret		ret	



ret

检测策略——连续gadget识别

指令长度阈值T0

Gadget长度阈值T1

1 4

pop	r13	指令计数器	1
pop	r14	gadget计数器	4
pop	r15		

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

pop rax
ret

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



指令长度阈值T0

Gadget长度阈值T1

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	add	rsp,8
mov	rsi,r14	pop	rbx
mov	rdi,r13	pop	rbp
call	rax	pop	r12
		pop	r13
non	rax	pop	r14
pop ret	lax	pop	r15
TCt		ret	



指令长度阈值T0

Gadget长度阈值T1

pop	r13			指令计数	器	3
pop	r14			gadget计数	女器	4
pop ret	r15	n	nov 1	dx,r15	add	rsp,
		n		rsi,r14	pop	rbx
	1	n	nov 1	:di,r13	pop	rbp
pop	rdx ·	C	all r	ax	pop	r12
pop	rs1				pop	r13
ret		10	1010	*OV	pop	r14
			1	ax	pop	r15
		Γ	et		ret	



指令长度阈值T0

Gadget长度阈值T1

pop	r13		│指令计数	数器	4	
pop	r14		gadgetì	数器	4	
pop	r15					
ret		mov	rdx,r15	add	rsp,	8
		mov	rsi,r14	pop	rbx	
		mov	rdi,r13	pop	rbp	
pop	rdx ·	call	rax	pop	r12	
pop	rs1			pop	r13	
ret		12.012	40 V	pop	r14	
		pop	rax	pop	r15	
		ret		ret		



指令长度阈值T0

Gadget长度阈值T1

pop	r13		指令计数	活	5
pop	r14		gadgeti+	数器	4
pop	r15				
ret		mov	rdx,r15	add	rsp,
		mov	rsi,r14	pop	rbx
	1	mov	rdi,r13	pop	rbp
pop	rdx	call	rax	pop	r12
pop	rs1			pop	r13
ret		non	rav	pop	r14
		pop 1	rax	pop	r15
		TCt		ret	



指令长度阈值T0

Gadget长度阈值T1

ret

pop	r13
pop	r14
pop	r15
ret	

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

指令计数器

gadget计数器

add	rsp,8
pop	rbx
pop	rbp
pop	r12
pop	r13
pop	r14
pop	r15
4	

6

rdx pop rsi pop

ret

pop rax

ret



指令长度阈值T0

Gadget长度阈值T1

4

pop	r13
pop	r14
pop	r15
ret	

pop	rdx
pop	rsi
ret	

gadget计数器 mov rdx,r15 add mov rsi,r14 pop mov rdi,r13 pop		指令计算	双斋	
mov rsi,r14 pop mov rdi,r13 pop		gadgetì	数器	4
	mov	rsi,r14	pop	

pop rax ret pop r13 pop r14

rsp,8

rbx

rbp

r12

pop r15

ret



指令长度阈值T0

ret

Gadget长度阈值T1

pop	r13		指令计	数器	8	>T0
pop	r14 r15		gadgeti	十数器	4	
pop ret	113	mov	rdx,r15	add	rsp,	8
		mov	rsi,r14	pop	rbx	
	1	mov	rdi,r13	pop	rbp	
pop	rdx	call	rax	pop	r12	
pop	rs1			pop	r13	
ret		non	rov	pop	r14	
		pop	rax	pop	r15	

syscall

ret



指令长度阈值T0

Gadget长度阈值T1

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	add	rsp,8
mov	rsi,r14	pop	rbx
mov	rdi,r13	pop	rbp
call	rax	pop	r12
		pop	r13
202	rov	pop	r14
pop	rax	pop	r15
ret		Tet	



指令长度阈值T0

Gadget长度阈值T1

pop r13pop r14pop r15

ret

pop rdx pop rsi ret

指令计数器	1
gadget计数器	0

mov rdx,r15

mov rsi,r14

mov rdi,r13

call rax

pop rax

ret



指令长度阈值T0

Gadget长度阈值T1

指令计数器 2 < T0

0

r13 r14 gadget计数器

pop r15

ret

pop

pop

mov rdx,r15

mov rsi,r14

mov rdi,r13

call rax

pop rdx

pop rsi

ret

pop rax

ret



指令长度阈值T0

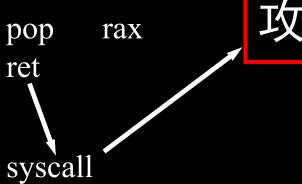
Gadget长度阈值T1

4

pop	r13
pop	r14
pop	r15
ret	

指令计数器	0
gadget计数器	1

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



攻击完成

检测策略(2): 被执行的调用指令数 不得小于返回指令数



mov	rdi, 0
call	func1
nop	
ret	

调用指令计数器	0
返回指令计数器	0

```
func1: func2: func3:
... ... ... ... ... ret
ret ret
```



mov	rdi, 0	
call	func1	
nop		
ret		

调用指令计数器	1
返回指令计数器	0

```
func1: func2: func3:
... ... ... ... ... ret
ret ret
```



mov	rdi, 0
call	func1
nop	
ret	

调用指令计数器	1
返回指令计数器	0

```
func1: func2: func3:

... ... ... ... ... ret

ret ret
```



调用指令计数器	2
返回指令计数器	0

```
func1: func2: func3:

... ... ... ... ... ret
ret ret
```



调用指令计数器	2
返回指令计数器	0

```
func1: func2: func3:
... ... ... ...
call func2 call func3 ...
ret
ret
```



调用指令计数器	3
返回指令计数器	0

```
func1: func2: func3:
... call func2 call func3 ...
ret
ret
```



调用指令计数器	3
返回指令计数器	0

```
func1: func2: func3:
... call func2 call func3 ...
ret
ret
```



调用指令计数器	4
返回指令计数器	0

```
func1: func2: func3: ... call func2 call func3 ... ret ret
```



调用指令计数器	4
返回指令计数器	0

```
func1: func2: func3:
... ... ... ... ... ... ret
ret ret
```

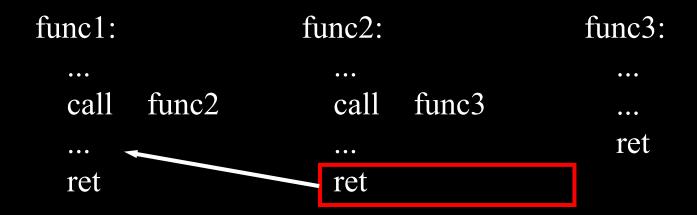


调用指令计数器	4
返回指令计数器	1

```
func1: func2: func3:
... call func2 call func3 ...
ret ret
```



调用指令计数器	4
返回指令计数器	1





调用指令计数器	4
返回指令计数器	2

```
func1: func2: func3:
... call func2 call func3 ...
ret
ret
```



mov rdi, 0 call func1

调用指令计数器 4 返回指令计数器 2

nop

ret

```
func1: func2: func3:
... ... ... ... ...
call func2 call func3 ...
ret
ret
```



ret

检测策略——返回指令数特征

mov rdi, 0 call func1 nop

调用指令计数器	4
返回指令计数器	3

func1: func2: func3:
... call func2 call func3 ...
ret
ret



mov rdi, 0 call func1 nop ret

调用指令计数器	4
返回指令计数器	3

func1: func2: func3:
... call func2 call func3 ...
ret
ret



调用指令计数器	4
返回指令计数器	4

```
func1: func2: func3:
... call func2 call func3 ...
ret
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



mov rdi, 0 call func1 nop ret

调用指令计数器	0
返回指令计数器	0

func1: gadget1: gadget2: ret ret



mov	rdi, 0	
call	func1	
nop		
ret		

调用指令计数器	1
返回指令计数器	0

func1: gadget1: gadget2: ret ret



ret

检测策略——返回指令数特征

mov rdi, 0 call func1 nop

调用指令计数器	1
返回指令计数器	0

func1: gadget1: gadget2: ret ret



调用指令计数器	2
返回指令计数器	0

func1:	gadget1:	gadget2:
•••		•••
ret	ret	ret



ret

检测策略——返回指令数特征

mov rdi, 0 call func1

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: gadget1: gadget2: ret



mov rdi, 0 call func1 nop ret

调用指令计数器	2
返回指令计数器	1

```
func1: gadget1: gadget2: ... ... ... ret ret
```



mov rdi, 0 call func1 nop ret

调用指令计数器	2
返回指令计数器	1

func1: gadget1: gadget2: ... ret ret ret



mov rdi, 0 call func1 nop ret

调用指令计数器	2
返回指令计数器	2

func1: gadget1: gadget2: ... ret ret ret



mov rdi, 0 call func1 nop ret

调用指令计数器	2
返回指令计数器	2

func1: gadget1: gadget2: ret ret ret



rdi, 0 mov func1 call nop ret

调用指令计数器 返回指令计数器

被攻击

func1: gadget1:

ret

ret

gadget2:

ret

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

其他特征?

如果能够发现程序控制流的异常 ...就能检测出ROP攻击

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(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
       edx,0x354
```

esi,0x4ad632 mov

eax,0 mov

0x400567 call

eax,0 mov

rsb, 0x8 add

rsp,rbp mov

ret

mov

0x400567:

push rbp

rbp, rsp mov

rsp, 0x80 sub

rax, [rbp-0x80] lea

edx, 0x200mov

rsi, rax mov

edi, 0 mov

call read

leave

ret



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



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



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



nop 0x400567: edx,0x354mov push rbp esi,0x4ad632 mov rbp, rsp mov eax,0 mov rsp, 0x80 sub 0x400567 call rax, [rbp-0x80] lea eax,0 mov edx, 0x200 mov rsb, 0x8 add rsi, rax mov rsp,rbp mov edi, 0 mov ret call read

leave

ret

...



nop

mov edx,0x354

mov esi,0x4ad632

mov eax,0

call 0x400567

mov eax,0

add rsb, 0x8

mov rsp,rbp

ret

0x400567:

mov

call

ret

leave

push rbp
mov rbp, rsp
sub rsp, 0x80
lea rax, [rbp-0x80]
mov edx, 0x200
mov rsi, rax

edi, 0

read

返回地址

• • •

• • •

• • •

. . .



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

旧rbp

返回地址

• • •

• • •

. . .



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

旧rbp 返回地址 ···

•••

• • •



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

•••

旧rbp 返回地址

. . .



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

... buf[128]

旧rbp

返回地址

. . .



ret

检测策略——检查返回地址完整性

ret

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

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

... buf[128] 旧rbp 返回地址



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

... buf[128] 旧rbp 返回地址

. . .



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

…
buf[128]
旧rbp
返回地址

. . .



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

... buf[128] 旧rbp 返回地址



leave

ret

nop 0x400567: edx,0x354mov push rbp esi,0x4ad632 mov rbp, rsp mov eax,0 mov rsp, 0x80 sub 0x400567 call rax, [rbp-0x80] lea eax,0 mov edx, 0x200 mov rsb, 0x8 add rsi, rax mov rsp,rbp mov edi, 0 mov ret call read

返回地址



nop mov edx,0x354 0x400567: mov esi,0x4ad63 先前调用返回地址

mov eax,0 call 0x400567

mov eax,0

add rsb, 0x8

mov rsp,rbp

ret

sub rsp, 0x80 lea rax, [rbp-0x80] mov edx, 0x200

mov rsi, rax

mov edi, 0

call read

leave

ret

返回地址

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



栈

•••
•••
•••

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

0x40056	7:
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	

影子栈

•••	
•••	
•••	
•••	
•••	



ret

检测策略——检查返回地址完整性

栈

•••

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

0x40056	57:
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	

影子栈



ret

检测策略——检查返回地址完整性

栈

•••

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

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

影子栈



栈

•••

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

ret

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

影子栈



栈

•••

nop edx,0x354mov esi,0x4ad632 mov eax,0 mov 0x400567 call eax,0 mov rsb, 0x8 add rsp,rbp mov ret

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

影子栈

...



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

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

返回地址 ...

影子栈

返回地址 ...





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

返回地址

• • •

• • •

• • •

• • •

影子栈

返回地址

•••

• • •

• •

• • •





nop edx,0x354 mov esi,0x4ad632 mov eax,0 mov $0x4\overline{00567}$ call eax,0 mov add rsb, 0x8 rsp,rbp mov ret

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

旧rbp 返回地址 ...

返回地址
...

影子栈



栈

larbp
返回地址
•••
•••

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

\mathbf{O}	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			

影子栈

返回地址
•••
•••
•••
•••





nop edx,0x354 mov esi,0x4ad632 mov eax,0 mov $0x4\overline{00567}$ call eax,0 mov rsb, 0x8 add rsp,rbp mov ret

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

... 旧rbp 返回地址 ...

影子栈

返回地址



nop edx,0x354 mov esi,0x4ad632 mov eax,0 mov $0x4\overline{00567}$ call eax,0 mov rsb, 0x8 add rsp,rbp mov ret

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

buf[128] 旧rbp 返回地址 …

彩**万**校 返回地址 … …



nop edx,0x354 mov esi,0x4ad632 mov eax,0 mov $0x4\overline{00567}$ call eax,0 mov rsb, 0x8 add rsp,rbp mov ret

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

buf[128] ||目rbp 返回地址 影子栈

影子枝 返回地址



nop edx,0x354 mov esi,0x4ad632 mov eax,0 mov $0x4\overline{00567}$ call eax,0 mov rsb, 0x8 add rsp,rbp mov ret

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

buf[128]
旧rbp
返回地址
...

影子栈 返回地址 ...



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

• • •

buf[128]

旧rbp

返回地址

• • •

影子栈

返回地址

• •

• • •

• • •

•••



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

• • •

buf[128]

(被覆盖)

Gadget地址

(恶意数据...)

影子栈

返回地址

• • •

• • •

• •

• •



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

Gadget地址

(恶意数据)

• • •

• • •

影子栈

返回地址

• • •

• • •

• • •

• • •

Gadget地址

(恶意数据)



nop

mov

mov

mov

call

mov

检测策略——检查返回地址完整性

Gadget: 57: edx,0x eax,0 mov rbp esi,0x4 ret rbp, rsp eax,0 sub rsp, 0x80 0x400567 rax, [rbp-0x80] lea eax,0

mov

mov

mov

call

ret

leave

edx, 0x200

rsi, rax

edi, 0

read

add rsb, 0x8 mov rsp,rbp

ret

影子栈

返回地址 ...



检测策略--检查返回地址完整性

Gadget地址

(恶意数据)

• • •

nop edx,0x354 mov esi,0x4ad632 mov

eax,0 mov

0x400567 call

eax,0 mov

rsb, 0x8 add

rsp,rbp mov

ret

rbp

rbp, rsp rsp. 0x80sub

rax, [rbp-0x80] lea

edx, 0x200 mov

rsi, rax mov

edi, 0 mov

call read

leave

ret

影子栈

返回地址



mov

add

mov

ret

检测策略--检查返回地址完整性

Gadget地址

(恶意数据)

• • •

nop edx,0x354 mov esi,0x4ad632 mov eax,0 mov $0x400\overline{567}$ call

eax,0

rbp rbp, rsp

> rsp. 0x80sub

rax, [rbp-0x80] lea edx, 0x200

rsi, rax mov

edi, 0 mov

call read

leave

mov

ret

被攻击

影子栈

返回地址

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



nop mov edx,0x354 mov esi,0x4ad63 先前调用返回地址

mov eax,0

call 0x400567

mov eax,0

add rsb, 0x8

mov rsp,rbp

ret

sub rsp, 0x80

lea rax, [rbp-0x80]

mov edx, 0x200

mov rsi, rax

mov edi, 0

call read

<u>le</u>ave

ret

返回地址



nop 0x400567: edx,0x354mov push rbp esi,0x4ad632 mov rbp, rsp mov eax,0 mov 0x400567 call 非先前调用返回地址 eax,0 mov edx, 0x200 mov rsb, 0x8 add rsi, rax mov rsp,rbp mov edi, 0 mov ret call read leave ret

异常地址

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
```

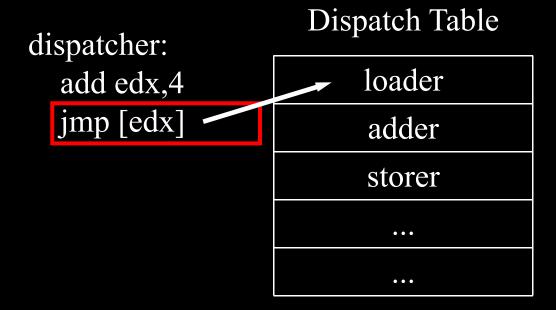
adder:

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

storer:

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





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

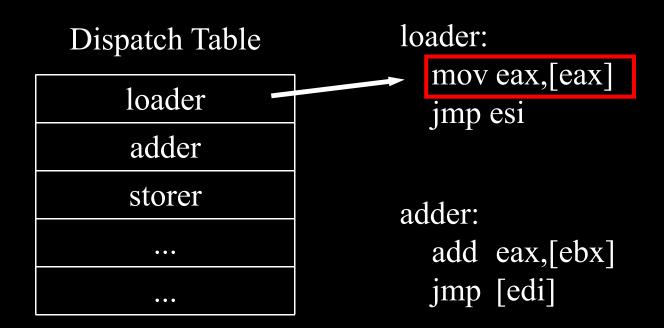
adder:
add eax,[ebx]
jmp [edi]
```

```
storer:

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



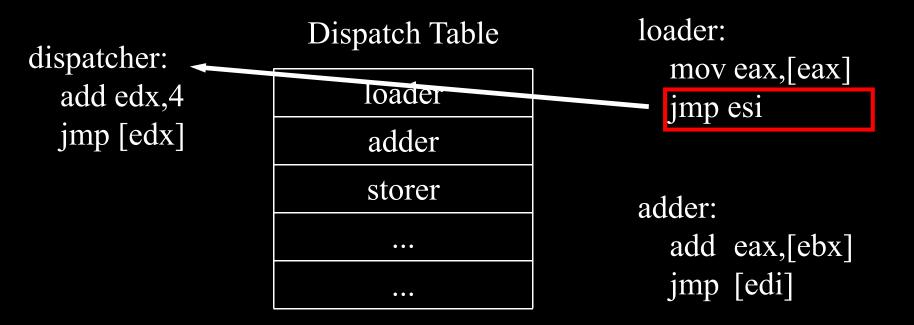
dispatcher: add edx,4 jmp [edx]



```
storer:

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

跳转导向编程(JOP)



```
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
```

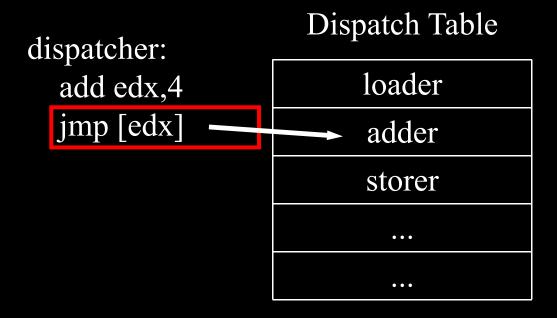
adder:

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

storer:

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





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

adder:
add eax,[ebx]
```

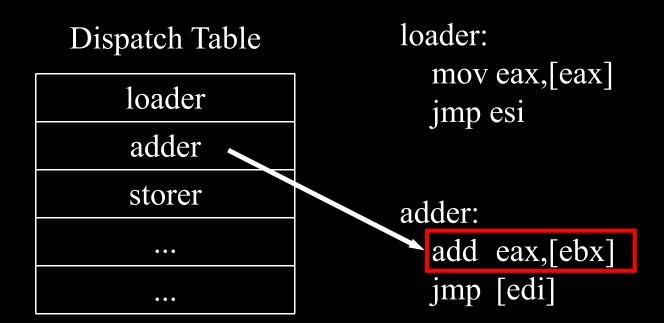
jmp [edi]

```
storer:

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

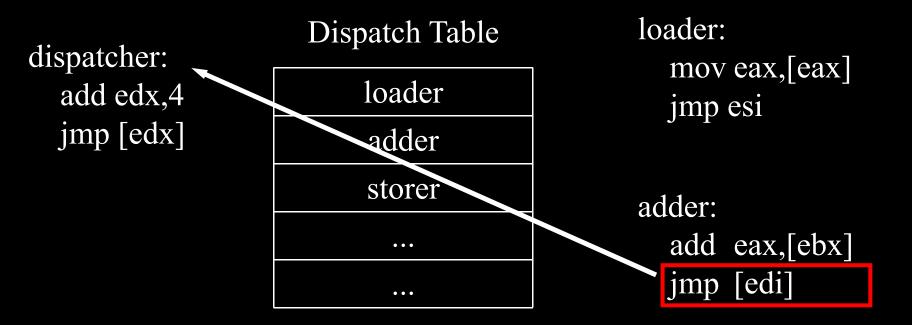


dispatcher: add edx,4 jmp [edx]



```
storer:

add [ecx],eax
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
```

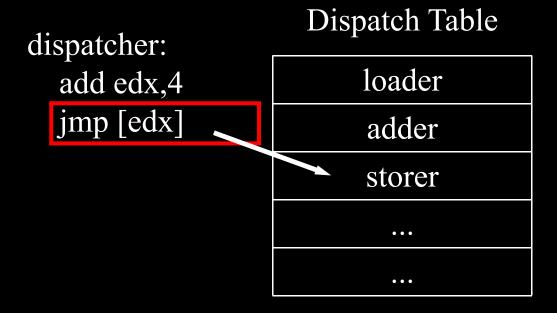
adder:

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

storer:

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





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

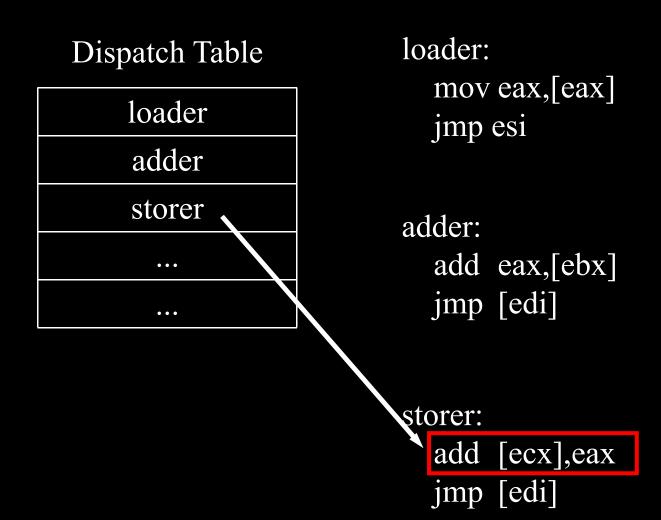
adder:
add eax,[ebx]
jmp [edi]
```

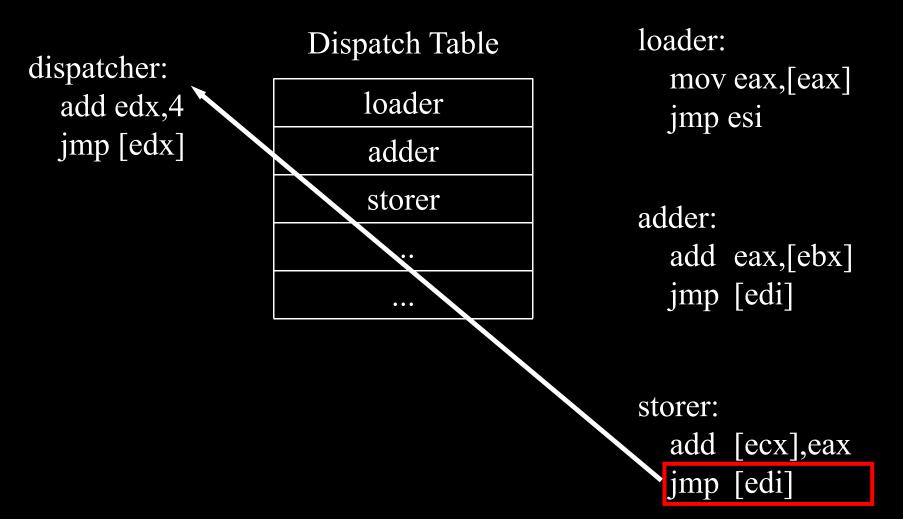
```
storer:

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



dispatcher: add edx,4 jmp [edx]







跳转导向编程(JOP)

dispatcher:

add edx,4

jmp [edx]

Dispatch Table

loader
adder
storer
•••
•••

loader:

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

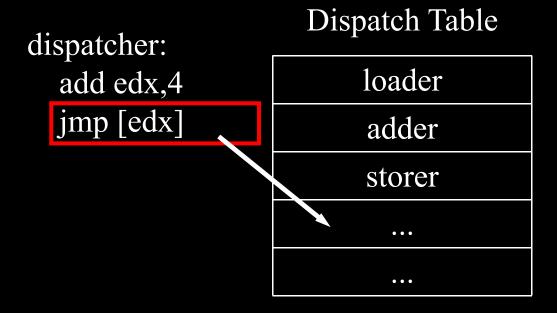
adder:

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

storer:

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





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

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

```
storer:

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



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(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

```
rdi, [rip + 0xdf]
lea
call
       0x4004a0
       eax, 0
mov
leave
ret
  0x4004a0 <puts@plt>:
          [rip+0x200b72]
    jmp
    push
         0x0
          0x400490
    jmp
```



检测策略——检查函数指针完整性

GOT表

lea call mov leave ret

•••	•••
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

0x4004a0 <puts@plt>:

jmp

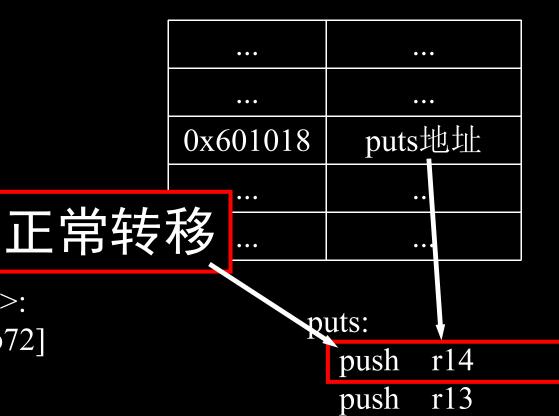
jmp

push 0x0

[rip+0x200b72]

0x400490

GOT表



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

GOT表

•••	•••
•••	•••
0x601018	Gadget地址
•••	•••

0x4004a0 <puts@plt>:

jmp [rip+0x200b72]

push 0x0

jmp 0x400490



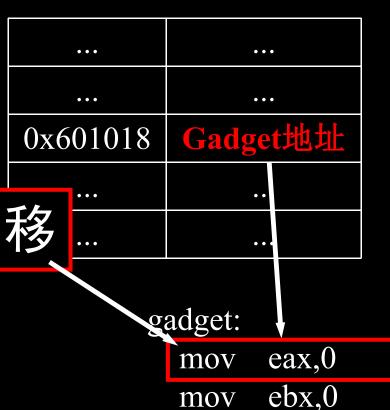
检测策略——检查函数指针完整性

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

leave

ret

GOT表



•••

ret

0x4004a0 <puts@plt>:

jmp [rip+0x200b72]

push 0x0

jmp 0x400490

策略缺陷:

难以监测程序中所有的函数指针

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

必须使用先前调用返回gadget 必须使用长gadget突破阈值

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

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

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



视频展示

谢谢!

🖰 答辩人: 刘天祺 🤷 指导教师: 詹静