

Return Oriented Programming (ROP)

Vortrag von Enrico

Agenda

- x86 Funktionsaufrufe
- Stack overflows
- DEP (Data Execution Prevention)
- ROP Konzept
- Gadgets und stack pivoting
- Beispiel
- Limitierungen

Funktionsaufruf x86

Was passiert genau?

Status vor einem Funktionsaufruf

Code

0x40000	call 0x40240
0x40008	cmp rax, 0x00
0x4000C	jnz some_address

Register

RSP	0x7FFF8
RIP	0x40000

Stack

0x7FFF8	0xdeadcdedeadebeef
0x80000	0xFF33782229

- Bei einem Call wird die return address auf den Stack gepusht.

Status nach Funktionsaufruf

Code

0x40240	nop
0x40241	nop
0x40242	ret

Register

RSP	0x7FFF0
RIP	0x40240

Stack

0x7FFF0	0x40008
0x7FFF8	0xdeadcdedeadebeef
0x80000	0xFF33782229

- RSP wird kleiner, und enthält return adress zur aufrufenden Funktion.

Status vor Return

Code

0x40240	nop
0x40241	nop
0x40242	ret

Register

RSP	0x7FFF0
RIP	0x40242

Stack

0x7FFF0	0x40008
0x7FFF8	0xdeadc0dedeadbeef
0x80000	0xFF33782229

- Bei einem ret, wird der Wert auf den der Stack Pointer zeigt, in RIP gepoppt.
- Stack Pointer Wert wird größer.

Status nach Return

Code

0x40000	call 0x40240
0x40008	cmp rax, 0x00
0x4000C	jnz some_address

Register

RSP	0x7FFF8
RIP	0x40008

Stack

0x7FFF8	0xdeadC0dedeadbeef
0x80000	0xFF33782229

Klassischer stack overflow

- Payload/Shellcode in Speicher schreiben
- return address auf dem Stack überschreiben

DEP

- Data Execution Prevention
- Keine Ausführung von code aus Pages die als RW markiert sind.
- NX Bit
- Eliminiert die Möglichkeit eigenen Shellcode einzuschleusen.



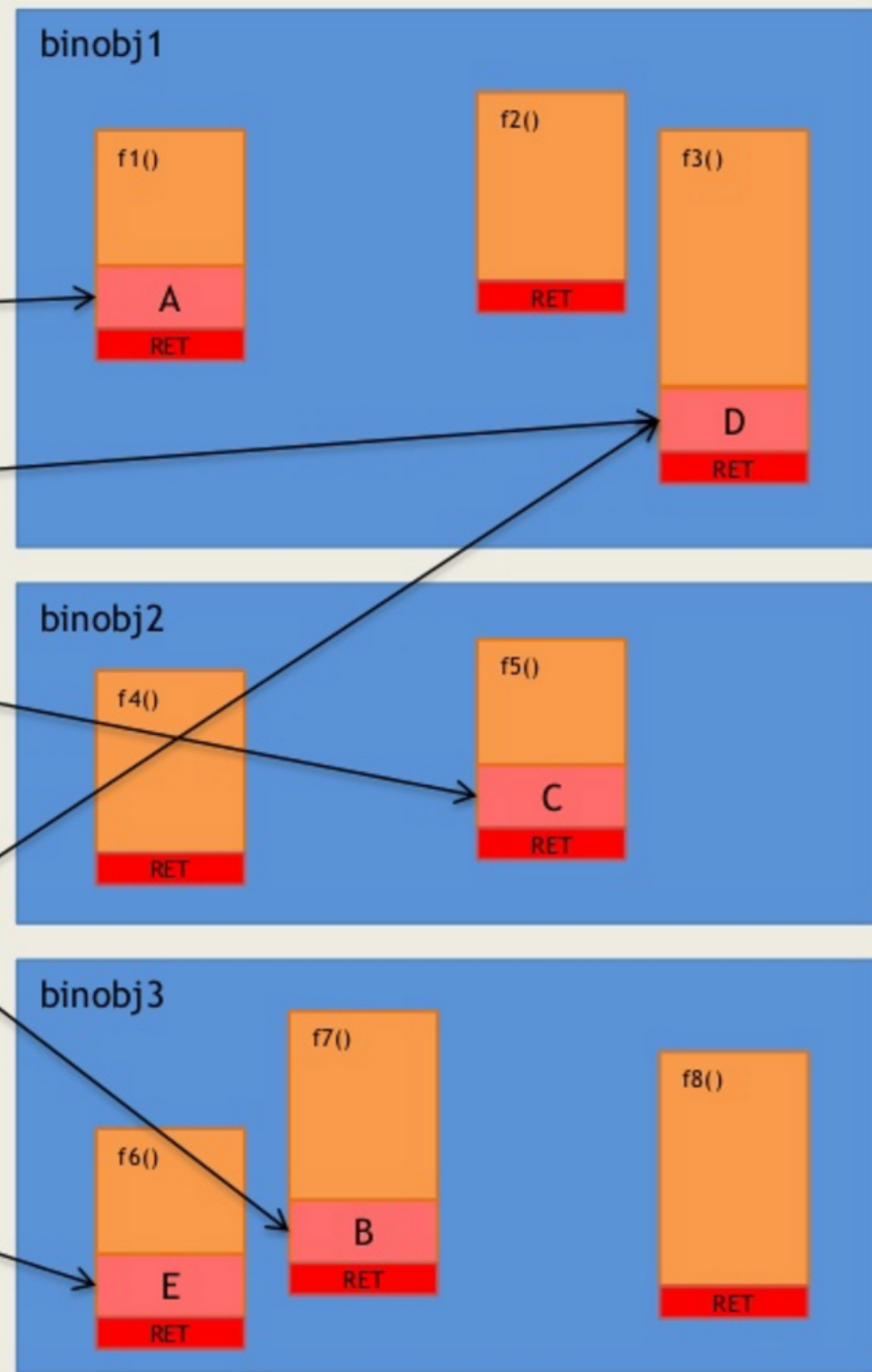
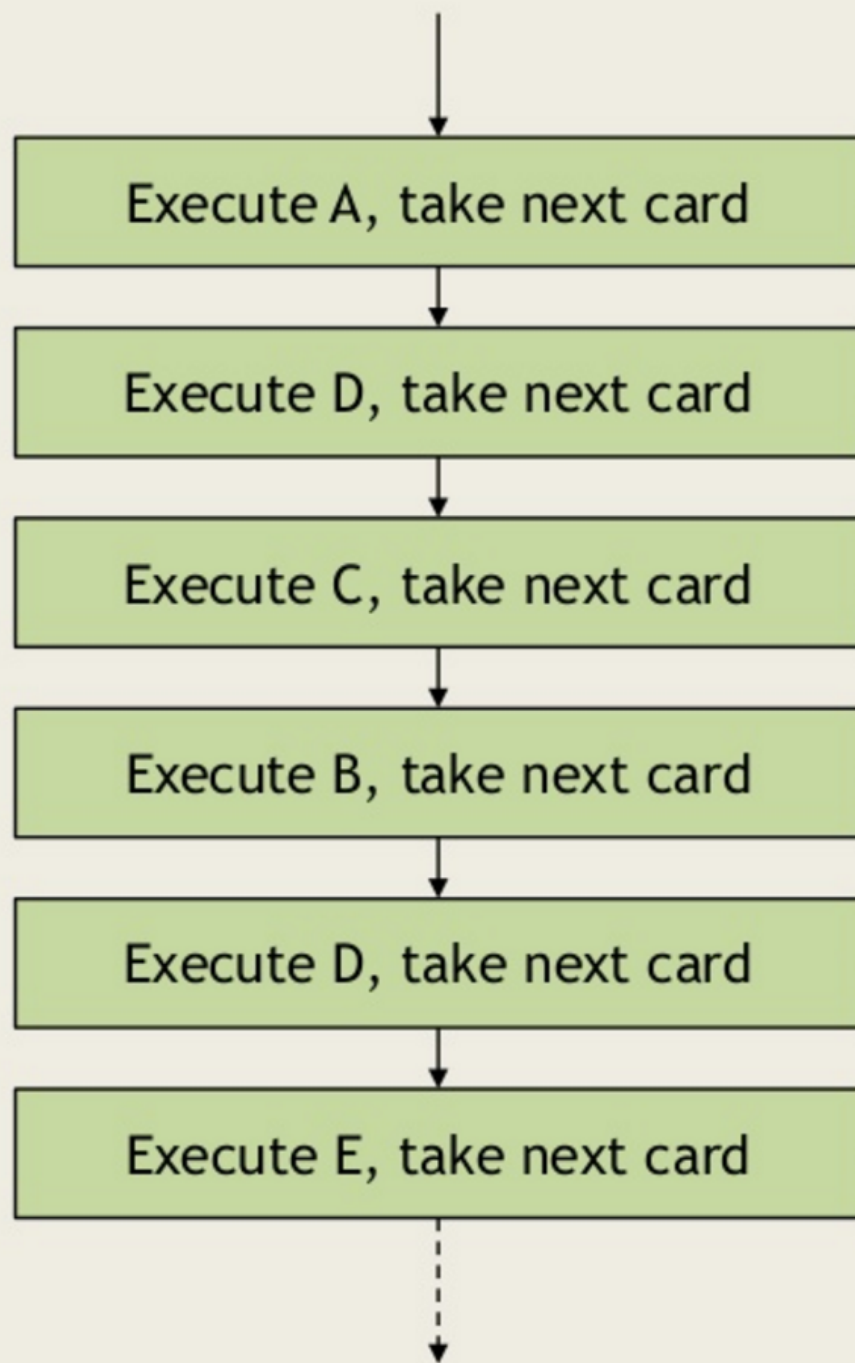
- Wie soll man Shellcode ausführen, wenn man keinen eigenen Code einschleusen kann?



- Man verwendet einfach Code der schon da ist!

ROP Konzept

- Kontrolliere ich den Stack, kontrolliere ich auch alle return Adressen!
 - Kontrolliere ich alle return Adressen, dann kontrolliere ich den gesamten Programmablauf!
- 1) ROP chain in Speicher kopieren (fake stack)
 - 2) Kontrolle über RSP erlangen (stack pivoting)



Gadgets

0x40010: mov rax,0xFFFFFFFF800C3580400

48 B8 00 04 58 C3 00 F8 FF FF

Gadgets

0x40010: mov rax,0xFFFF800C3580400

48 B8 00 04 58 C3 00 F8 FF FF

RIP → 0x40014 → 58 C3 → pop rax ; ret

ROP chain

- Array aus 64 bzw. 32 bit Adressen

0xFFFFFFFF40670290	pop rdi ; ret
0xFFFFFFFFFFFFFFFFFE	
0xFFFFFFFF40273980	pop rsi ; ret
0x0000000000000001	
0xFFFFFFFF4A9A6D8F	pop rax ; ret
0x0000000000000001	
0xFFFFFFFF88BA78FE	syscall

Kontrolle über RSP

- Stack pivoting
 - `mov rsp, [REGISTER] ; ret`
 - `xchg rsp, [REGISTER] ; ret`
- Ein Gadget!

Beispiel

Code

0x40240	nop
0x40241	nop
0x40242	ret

Register

RSP	0x7FFF0
RIP	0x40242
RAX	0x84000

Stack

0x7FFF0	0x40008
0x7FFF8	0xdeadcdedeadebeef
0x80000	0xFF33782229

Normalfall

Code

0x40240	nop
0x40241	nop
0x40242	ret

Register

RSP	0x7FFF0
RIP	0x40242
RAX	0x84000

Stack

0x7FFF0	0x40008
0x7FFF8	0xdeadcdedeadbeef
0x80000	0xFF33782229

- Wir möchten aber RSP unter unsere Kontrolle bringen...

Stack pivoting

Code

0x40240	nop
0x40241	nop
0x40242	ret

Register

RSP	0x7FFF0
RIP	0x40242
RAX	0x84000

Stack

0x7FFF0	0x40008
0x7FFF8	0xdead0dedeadbeef
0x80000	0xFF33782229

Stackpivot gadget

0x40380

mov rsp, rax ; ret

Stack pivoting

Code

0x40240	nop
0x40241	nop
0x40242	ret

Register

RSP	0x7FFF0
RIP	0x40242
RAX	0x84000

Stack

0x7FFF0	0x40380
0x7FFF8	0xdeadc0dedeadbeef
0x80000	0xFF33782229

- Was passiert ret?



Stack pivoting

Code

0x40380	mov rsp, rax
0x40381	ret
0x40382	

Register

RSP	0x7FFF8
RIP	0x40380
RAX	0x84000

Stack

0x7FFF8	0xdeadc0dedeadbeef
0x80000	0xFF33782229

- mov rsp, rax = verschiebt den Wert von RAX in RSP

Stack pivoting

Code

0x40380	mov rsp, rax
0x40381	ret
0x40382	

Register

RSP	0x7FFF8
RIP	0x40380
RAX	0x84000

Stack

0x7FFF8	0xdeadc0dedeadbeef
0x80000	0xFF33782229

- RAX ist in einem Bereich der RW ist.
- ROP Chain genau dort platzieren!!!

Stack pivoting

Code

0x40380	mov rsp, rax
0x40381	ret
0x40382	

Register

RSP	0x84000
RIP	0x40381
RAX	0x84000

Stack

0x84000	0xFFFFFFFF40670290
0x84008	0xFFFFFFFFFFFFFFFFFE
0x84010	0xFFFFFFFF40273980
0x84018	0x01
0x84020	0xFFFFFFFF4A9A6D8F
0x84028	0x01
0x84030	0xFFFFFFFF4A9B99A0

- Was passiert beim ret?

ROP execution

0x84000	0xFFFFFFFF40670290	pop rdi ; ret
0x84008	0xFFFFFFFFFFFFFFFFFE	-2
0x84010	0xFFFFFFFF40273980	pop rsi ; ret
0x84018	0x01	1
0x84020	0xFFFFFFFF4A9A6D8F	pop rax ; ret
0x84028	0x01	1
0x84030	0xFFFFFFFF4A9B99A0	syscall ; ret

ROP execution

0x84000	0xFFFFFFFF40670290	pop rdi ; ret
0x84008	0xFFFFFFFFFFFFFFFFFE	-2
0x84010	0xFFFFFFFF40273980	pop rsi ; ret
0x84018	0x01	1
0x84020	0xFFFFFFFF4A9A6D8F	pop rax ; ret
0x84028	0x01	1
0x84030	0xFFFFFFFF4A9B99A0	syscall ; ret

RDI	0xFFFFFFFFFFFFFFFFFE
-----	----------------------

ROP execution

0x84000	0xFFFFFFFF40670290	pop rdi ; ret
0x84008	0xFFFFFFFFFFFFFFFFFE	-2
0x84010	0xFFFFFFFF40273980	pop rsi ; ret
0x84018	0x01	1
0x84020	0xFFFFFFFF4A9A6D8F	pop rax ; ret
0x84028	0x01	1
0x84030	0xFFFFFFFF4A9B99A0	syscall ; ret

RDI	0xFFFFFFFFFFFFFFFFFE
-----	----------------------

ROP execution

0x84000	0xFFFFFFFF40670290	pop rdi ; ret
0x84008	0xFFFFFFFFFFFFFFFFFE	-2
0x84010	0xFFFFFFFF40273980	pop rsi ; ret
0x84018	0x01	1
0x84020	0xFFFFFFFF4A9A6D8F	pop rax ; ret
0x84028	0x01	1
0x84030	0xFFFFFFFF4A9B99A0	syscall ; ret

RDI	0xFFFFFFFFFFFFFFFFFE
RSI	0x01

ROP execution

0x84000	0xFFFFFFFF40670290	pop rdi ; ret
0x84008	0xFFFFFFFFFFFFFFFFFE	-2
0x84010	0xFFFFFFFF40273980	pop rsi ; ret
0x84018	0x01	1
0x84020	0xFFFFFFFF4A9A6D8F	pop rax ; ret
0x84028	0x01	1
0x84030	0xFFFFFFFF4A9B99A0	syscall ; ret

RDI	0xFFFFFFFFFFFFFFFFFE
RSI	0x01

ROP execution

0x84000	0xFFFFFFFF40670290	pop rdi ; ret
0x84008	0xFFFFFFFFFFFFFFFFFE	-2
0x84010	0xFFFFFFFF40273980	pop rsi ; ret
0x84018	0x01	1
0x84020	0xFFFFFFFF4A9A6D8F	pop rax ; ret
0x84028	0x01	1
0x84030	0xFFFFFFFF4A9B99A0	syscall ; ret

RDI	0xFFFFFFFFFFFFFFFFFE
RSI	0x01
RAX	0x01

ROP execution

0x84000	0xFFFFFFFF40670290	pop rdi ; ret
0x84008	0xFFFFFFFFFFFFFFFFFE	-2
0x84010	0xFFFFFFFF40273980	pop rsi ; ret
0x84018	0x01	1
0x84020	0xFFFFFFFF4A9A6D8F	pop rax ; ret
0x84028	0x01	1
0x84030	0xFFFFFFFF4A9B99A0	syscall ; ret

RDI	0xFFFFFFFFFFFFFFFFFE
RSI	0x01
RAX	0x01

syscall(1,-2,1);

Limitierungen

- Anzahl brauchbarer Gadgets
- Keine jumps/compares (nur bedingt)
- Lange chains werden schnell unübersichtlich
- ASLR (Address Space Layout Randomization)

Vielen Dank!