Bomblab

Yilong LI
13 Oct, 2016

System V calling convention on x86-64

- the de facto standard among Unix and Unix-like operating systems

- RDI, RSI, RDX, RCX, R8, R9 will be used as the first 6 arguments

- If the callee wishes to use registers RBP, RBX, and R12–R15, it must restore their original values before returning control to the caller

Phase 1

```
(gdb) disas
Dump of assembler code for function phase_1:
=> 0x0000000000400fad <+0>:   sub    $0x8,%rsp
   0x0000000000400fb1 <+4>:   mov    $0x402870,%esi
   0x0000000000400fb6 <+9>:   callq  0x401483 <strings_not_equal>
   0x0000000000400fbb <+14>:  test   %eax,%eax
   0x0000000000400fbd <+16>:  je     0x400fc4 <phase_1+23>
   0x0000000000400fbf <+18>:  callq  0x401756 <explode_bomb>
   0x0000000000400fc4 <+23>:  add    $0x8,%rsp
   0x0000000000400fc8 <+27>:  retq

(gdb) x/s 0x402870
0x402870:   "And they have no disregard for human life."
```

Phase 2

- 等比数列
- 1 2 4 8 16 32

```
0x0000000000400fe1 <+24>:    callq  0x40178c <read_six_numbers>
0x0000000000400fe6 <+29>:    cmpl   $0x1,(%rsp)
0x0000000000400fea <+33>:    je     0x400ff1 <phase_2+40>
0x0000000000400fec <+35>:    callq  0x401756 <explode_bomb>
0x0000000000400ff1 <+40>:    mov    $0x1,%ebx
0x0000000000400ff6 <+45>:    jmp    0x401012 <phase_2+73>
0x0000000000400ff8 <+47>:    movslq %ebx,%rdx
0x0000000000400ffb <+50>:    lea    -0x1(%rbx),%eax
0x0000000000400ffe <+53>:    cltq
0x0000000000401000 <+55>:    mov    (%rsp,%rax,4),%eax
0x0000000000401003 <+58>:    add    %eax,%eax
0x0000000000401005 <+60>:    cmp    %eax,(%rsp,%rdx,4)
0x0000000000401008 <+63>:    je     0x40100f <phase_2+70>
0x000000000040100a <+65>:    callq  0x401756 <explode_bomb>
0x000000000040100f <+70>:    add    $0x1,%ebx
0x0000000000401012 <+73>:    cmp    $0x5,%ebx
0x0000000000401015 <+76>:    jle    0x400ff8 <phase_2+47>
0x0000000000401017 <+78>:    mov    0x18(%rsp),%rax
0x000000000040101c <+83>:    xor    %fs:0x28,%rax
0x0000000000401025 <+92>:    je     0x40102c <phase_2+99>
0x0000000000401027 <+94>:    callq  0x400c00 <__stack_chk_fail@plt>
0x000000000040102c <+99>:    add    $0x20,%rsp
0x0000000000401030 <+103>:   pop    %rbx
```

Phase 3

- switch-case
- 0x000000000040106b <+57>: jmpq *0x4028e0(,%rax,8)
- (gdb) x/x 0x4028e0
- 0x4028e0: 0x00401079

- 3 -410

```
0x00000000004010c1 <+143>:   mov    $0x0,%eax
0x00000000004010c6 <+148>:   sub    $0x19a,%eax
0x00000000004010cb <+153>:   jmp    0x4010d7 <phase_3+165>
0x00000000004010cd <+155>:   callq  0x401756 <explode_bomb>
0x00000000004010d2 <+160>:   mov    $0x0,%eax
0x00000000004010d7 <+165>:   cmpl   $0x5,(%rsp)
0x00000000004010db <+169>:   jg     0x4010e3 <phase_3+177>
0x00000000004010dd <+171>:   cmp    0x4(%rsp),%eax
0x00000000004010e1 <+175>:   je     0x4010e8 <phase_3+182>
0x00000000004010e3 <+177>:   callq  0x401756 <explode_bomb>
0x00000000004010e8 <+182>:   mov    0x8(%rsp),%rax
0x00000000004010ed <+187>:   xor    %fs:0x28,%rax
0x00000000004010f6 <+196>:   je     0x4010fd <phase_3+203>
0x00000000004010f8 <+198>:   callq  0x400c00 <__stack_chk_fail@plt>
0x00000000004010fd <+203>:   add    $0x18,%rsp
0x0000000000401101 <+207>:   retq
```

Phase 4

- **Recursion**

- phase 4
- x y
- x < 0 -> bomb
- eax > 14 -> bomb

- edx = 14, esi = 0, edi = x
- call func4 f(x, 0, 14)
- eax != 2 -> bomb

- y != 2 -> bomb

Phase 4

- f(p, q, r)

- eax = r - q + sgn bit (r - q)
- eax >>= 1
- eax += q

- eax > p?
- return 2 * f(p, q, eax-1)
- else
- eax == p?
- return 0
- return 2 * f(p, eax + 1, r) + 1

f(x, 0, 14) = 2, eax = 7
f(x, 0, 6) = 1, eax = 3
eax < x
f(x, 4, 6) = 0
eax = 5
x = 5

x = 5, y = 2

Phase 5

- (gdb) print/x *0x402920@16
- $11 = {0xa, 0x2, 0xe, 0x7, 0x8, 0xc, 0xf, 0xb, 0x0, 0x4, 0x1, 0xd, 0x3, 0x9, 0x6, 0x5}
- 5 115

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A	2	E	7	8	C	F	B	0	4	1	D	3	9	6	5

Phase 6

- 链表，排列组合

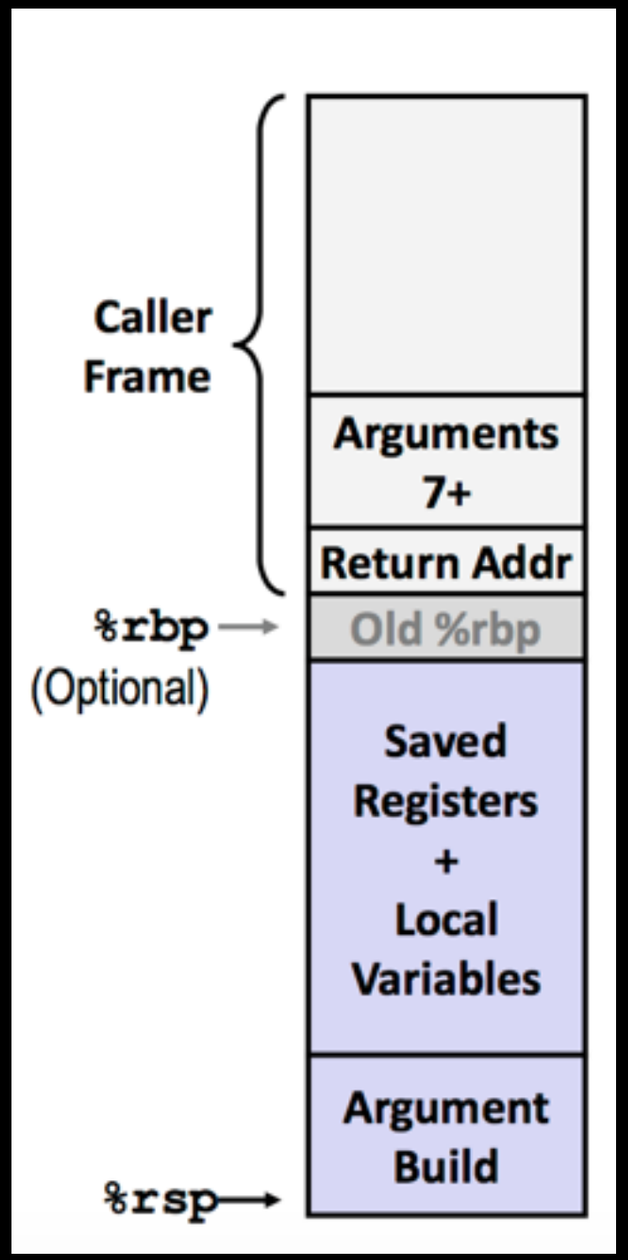
- 3: /x *0x604300@50 =
  {0x1c1, 0x1, 0x604310, 0x0,
  0x2ce, 0x2, 0x604320, 0x0,
  0x171, 0x3, 0x604330, 0x0,
  0xa0, 0x4, 0x604340, 0x0,
  0x28c, 0x5, 0x604350, 0x0,
  0xdd, 0x6, 0x0, 0x0,
  0x626d6f62, 0x62616c, 0x0 <repeats 24 times>}

Secret Phase

- tao@ubuntu64

缓冲区溢出攻击

- stack overflow
- example: gets() function
- — never use!
- — deprecated in C11 std



缓冲区溢出攻击

- 软件安装/网站注册时的用户协议有哪些著名的陷阱条款?
- https://www.zhihu.com/question/26978264/answer/126320570