



# Engenharia Reversa

asrever ed etnerf

radare2

Functions

entry0

fcn.00402486

fcn.00402136

fcn.00402146

fcn.00402156

fcn.00402166

fcn.00402176

fcn.00402186

fcn.00402496

fcn.004024a6

fcn.004024b6

fcn.004024c6

fcn.004024d6

fcn.004024e6

fcn.004024f6

fcn.00404870

fcn.004048b0

fcn.004048f0

fcn.00404910

fcn.00404940

fcn.00404950

fcn.00404970

fcn.00404990

fcn.00404a60

fcn.00404a70

Symbols

Relocs

Imports

Flags

Disassembler

Hex Dump

Strings

Entropy

Settings

0x404795

0x40479c

0x4047a1

0x4047a6

0x4047ab

0x4047ae

0x4047b4

0x4047bb

0x4047bf

0x4047c4

0x4047c8

0x4047cd

0x4047cf

0x4047d1

0x4047d4

0x4047d7

0x4047dc

0x4047e1

0x4047e4

0x4047e9

0x4047ed

0x4047f2

0x4047f4

0x4047f8

0x404804

0x404806

0x404808

0x40480e

0x404816

0x40481b

0x40481d

mov rax, qword [rip + 0x218304]

mov rdi, qword [rsp + 0x28]

lea rsi, qword [rsp + 0x38]

mov edx, 1

mov rcx, r13

add qword [rsp + 0x38], 1

mov qword [rip + 0x2182e5], r13

mov qword [r13 + 0x20], rax

mov rax, qword [rsp + 0x40]

mov qword [r13 + 8], rax

call 0x404a70

cmp al, 1

sub edx, edx

and edx, 2

add edx, 3

jmp 0x404362

JMP XREF from 0x404763

mov rax, qword [rsp + 0x40]

mov rcx, r13

mov rdi, qword [rsp + 0x28]

shl rcx

lea rsi, qword [rsp + 0x38]

xor edx, edx

add rax, 0x61bc80

mov r14, qword [rsp + 0x40]

call 0x404a70

xor edx, edx

test al, al

jne 0x40436b

JMP XREF from 0x404775

lea rdi, qword [rsp + 0xf0]

call 0x404a70

xor edi, edi

mov r14, rax

file /bin/ls

type EXEC (Executable)

pic false

canary true

nx true

crypto false

va true

root elf

class ELF64

lang c

arch x86

bits 64

machine AMD x86-64 arch

os linux

subsys linux

endian little

strip true

static false

linenum false

lsyms false

relocs false

rpath NONE

type EXEC (Executable)

os linux

arch AMD x86-64 arch

bits 64

endian little

file /bin/ls

fd 6

size 0x1c6f8

mode r--

...

Sections

> ar

r15 0x00000000

r12 0x00000000

r11 0x00000000

r8 0x00000000

rdx 0x00000000

orax 0x00000000

rsp 0x00000000

r14 0x00000000

rbp 0x00000000

r10 0x00000000

rax 0x00000000

rsi 0x00000000

rip 0x00000000

r13 0x00000000

rbx 0x00000000

r9 0x00000000

rcx 0x00000000

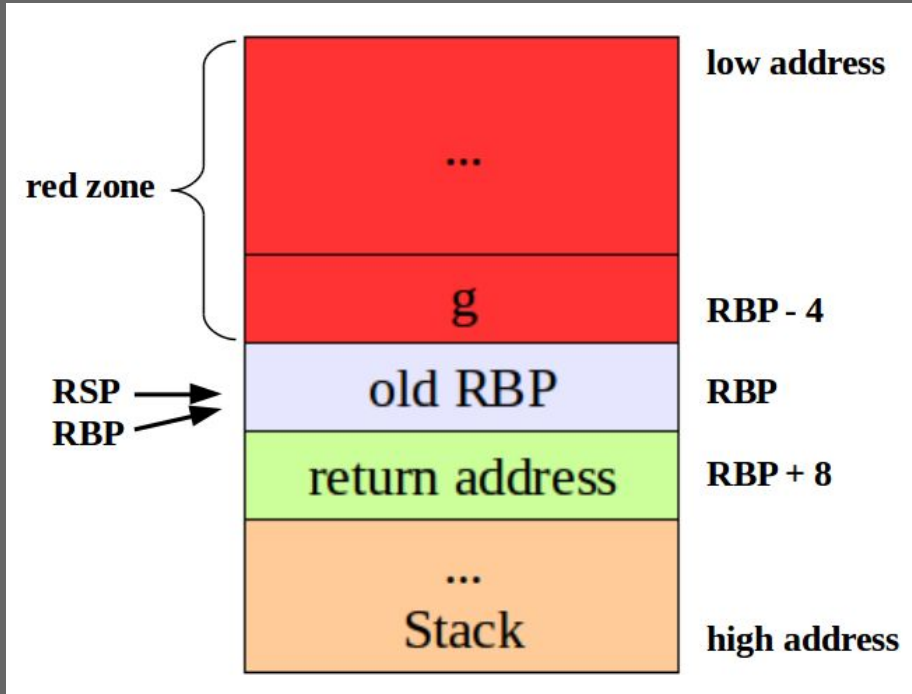
rdi 0x00000000

rflags =

Pilha - O que é?

2

# Funcionamento da pilha



- Pilha

- rsp: stack pointer (topo)
- rbp: base pointer (base)

# Como manipular?



- Push / Pop
- add / sub
- lea

```
push rbp
```

```
pop rbp
```

```
sub rsp, 0x10
```

```
add rsp, 0x10
```

```
lea eax, DWORD PTR[rbp - 0xa]
```

# Por que é importante

---



- Declaração de variáveis
- Chamada de funções
  - retorno
  - rbp

# Exemplo



```
#include <stdio.h>

int main(){
    printf("Hello, world!\n");
    return 0;
}
```

```
00000000000001139 <main>:
   1139: 55                push    rbp
   113a: 48 89 e5          mov     rbp, rsp
   113d: 48 8d 3d c0 0e 00 00 lea     rdi, [rip+0xec0]          # 2004 <_IO_stdin_used+0x4>
   1144: e8 e7 fe ff ff    call    1030 <puts@plt>
   1149: b8 00 00 00 00    mov     eax, 0x0
   114e: 5d                pop     rbp
   114f: c3                ret
```

# Ferramentas úteis



- objdump
- Hexdump (xxd)
- gdb [+ peda]
- nasm

```
$ objdump -Intel -d prog  
  
$ hexdump prog  
  
$ nasm -f elf64 prog.asm  
$ ld -s -o prog prog.o -m elf_x86_64  
$ ./prog ; echo $?
```



- GDB

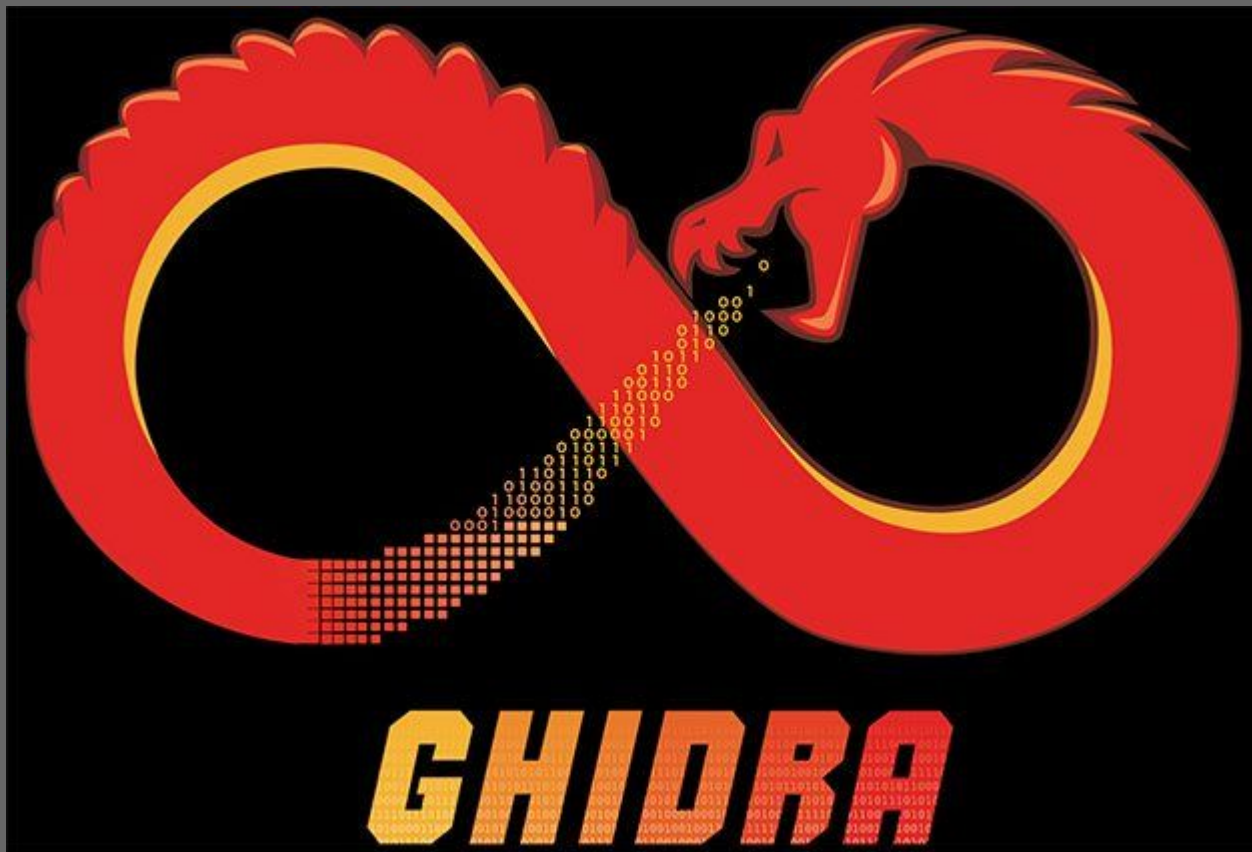




# Ferramenta OP



- Ghidra



# GANESH

Grupo de Segurança da Informação  
ICMC / USP - São Carlos, SP  
<http://ganesh.icmc.usp.br/>  
[ganesh@icmc.usp.br](mailto:ganesh@icmc.usp.br)

