

# Building an alternative to IDA

Radare2

# History

- radare in 2006



# History



- radare in 2006
- radare2 in 2009

# History

---



- radare in 2006
- radare2 in 2009
- written in pure C

# History



- radare in 2006
- radare2 in 2009
- written in pure C
- 350k LoC

# History



- radare in 2006
- radare2 in 2009
- written in pure C
- 350k LoC
- several contributors

# History



- radare in 2006
- radare2 in 2009
- written in pure C
- 350k LoC
- several contributors
- RSoC



# Platforms

## Runs on

- Windows
- Linux
- BSD
- OSX
- Android
- iOS
- Smartwatch
- Browser

## Handles

- PE/COFF
- ELF
- Mach0
- DEX/JAVA
- BIOS/TE
- GB/GBA/DS
- XBOX
- Plan9

# Architectures

- 8051
- arc
- arm
- avr
- brainfuck
- cr16
- csr
- dalvik
- dcpu16
- ebc
- gb
- h8300
- i8080
- java
- m68k
- malbolge
- mips
- msil
- msp430
- nios2
- powerpc
- rar
- sh
- sparc
- spc700
- sysz
- tms320
- v850
- whitespace
- x86
- xcore
- z80
- propeller
- snes
- psosvm
- 6502

# What about a GUI?

The screenshot displays the 'Bokken' debugger interface. The title bar reads 'Bokken, a GUI for pyw and radare2!'. The menu bar includes 'File', 'Edit', 'View', and 'Help'. Below the menu is a toolbar with icons for home, refresh, search, and other functions. A search box contains the text 'String' and 'Text to search'. The main window is divided into several panes:

- Functions:** A list of functions including 'entry0', 'fcn.00401080', 'fcn.004012e0', 'fcn.004013c0', 'fcn.00401560', 'fcn.004015f0', 'fcn.00401820', 'fcn.00402a90', 'fcn.00402d00', 'fcn.00402e00', 'fcn.00404ca0', 'fcn.00404d00', 'fcn.00404f30', 'fcn.00405b50', 'loc.004011d7', 'loc.004011e0', 'loc.0040126a', 'loc.00401278', 'loc.004012d0', 'loc.00401336', 'loc.004013f8', 'loc.004013ff', 'loc.00401418', 'loc.00401449', 'loc.004014b8', 'loc.00401540', 'loc.00401600', 'loc.0040160f', 'loc.00401620', and 'loc.00401635'.
- Imports:** A list of imported symbols.
- Exports:** A list of exported symbols.
- Code:** The main pane showing assembly code for the selected function 'fcn.00402000'. The code includes instructions like 'jmp', 'push', 'call', 'xor', 'mov', 'pop', 'and', 'push', 'push', 'mov', 'mov', 'mov', 'call', 'hlt', 'nop', 'nop', 'sub', 'mov', 'test', 'jz', 'call', 'add', and 'ret'. Comments and labels like 'section: .text', 'section: .text', and 'lib: start\_main' are also visible.
- Find:** A search box at the bottom of the code pane.

The status bar at the bottom indicates 'Processor: AMD x86-64 architecture | Name: /bin/true | Format: elf' and 'Bokken 1.5-dev'.

# What about a GUI?

---

- We don't like GUI.

# What about a GUI?

---

- We don't like GUI.
- We don't use GUI.

## What about a GUI?

---

- We don't like GUI.
- We don't use GUI.
- There are unmaintained ones

# What about a GUI?

---

- We don't like GUI.
- We don't use GUI.
- There are unmaintained ones
- We're busy.

## What about a GUI?

---

- We don't like GUI.
- We don't use GUI.
- There are unmaintained ones
- We're busy.
- R2 is open source, write one.



## What about a GUI?

---

- We don't like GUI.
- We don't use GUI.
- There are unmaintained ones
- We're busy.
- R2 is open source, write one.
- Pay us to write one?

## But what about graphs?

```
int main(int argc, char* argv[]){  
    if (argc > 1 && !strcmp(argv[1], "1337"))
```

```
    puts ("Lose");
```

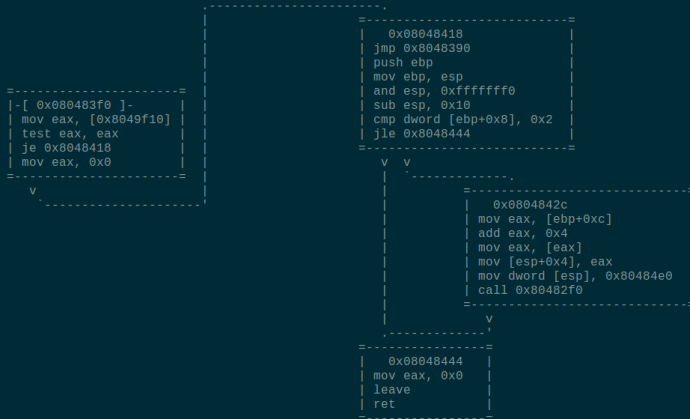
```
    puts ("Win");
```

```
    return 0;  
}
```

```
graph TD; A["int main(int argc, char* argv[]){  
    if (argc > 1 && !strcmp(argv[1], \"1337\"))"] -- Green --> B["puts (\"Lose\");"]; A -- Red --> C["puts (\"Win\");"]; B -- Green --> D["return 0;  
}"]; C -- Red --> D;
```

# But what about graphs?

```
[0x080483f0]> VV @ sym.frame_dummy (nodes 4)
```



# Analysis

- Functions detection
- Local var detection
- FLIRT integration
- (X)REF
- DWARF and PDB

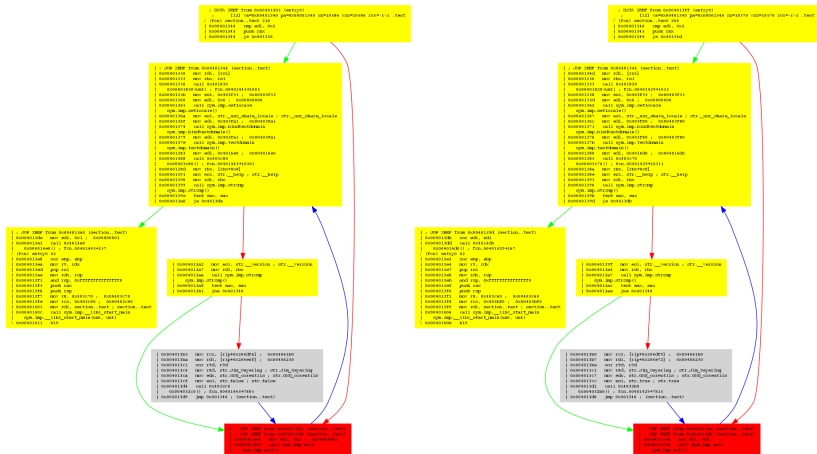
```
[0x00048390]> pdf @ main
+ (fcn) main 95
0x000485c0 8d4c2404 lea ecx, [esp+0x4]
0x000485c4 83e4f0 and esp, 0xffffffff
0x000485c7 ff71fc push_dword [ecx-0x4]
0x000485ca 55 push ebp
0x000485cb 89e5 mov ebp, esp
0x000485cd 51 push ecx
0x000485ce 83ec84 sub esp, 0x4
0x000485d1 8b4104 mov eax, [ecx+0x4]
0x000485d4 833081 cmc dword [ecx], 0x1
0x000485d7 7f10 jg 0x000485f2
0x000485d9 83ec88 sub esp, 0x8
0x000485dc ff30 push_dword [eax]
0x000485de 60a6860408 push str.Usage__s_key_n ; str.Usage__s_key_n
0x000485e3 e840fdffff call sym.imp.printf ; (fcn.0804832c)
          fcn.0804832c(unk, unk, unk, unk) ; sym.imp.printf
0x000485e8 83c410 add esp, 0x10
0x000485eb b8ffffff mov eax, 0xffffffff ; -1
0x000485f0 eb25 jmp 0x00048617 ; (main)
0x000485f2 8b4084 mov eax, [eax+0x4]
0x000485f5 a3b890408 mov [0x00499b0], eax
0x000485fa ba00010000 mov edx, 0x100 ; 0x00000100
0x000485ff eb0a jmp 0x004860b ; (main)
0x00048601 83e801 sub eax, 0x1
0x00048604 75fb jne 0x0048601
0x00048606 83ea01 sub edx, 0x1
0x00048609 7407 je 0x0048612
0x0004860b b800010000 mov eax, 0x100 ; 0x00000100
0x00048610 ebef jmp 0x0048601 ; (main)
0x00048612 b800000000 mov eax, 0x0
0x00048617 8b40fc mov ecx, [ebp-0x4] ; 0x000000fc
0x0004861a c9 leave
0x0004861b 8d61fc lea esp, [ecx-0x4] ; 0x000000fc
0x0004861e c3 ret

[0x00048390]> axt main
d 0x00483a7 push 0x00485c0
[0x00048390]> □
```

# Binary diffing with radiff2

/bin/true

/bin/false



# Assembling and disassembling

```
jvoisin@kaa 2:30 ~ ragg2 -i exec -a x86 -b 32 -z
"\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\x99\xb0\b\xcd\x80"
jvoisin@kaa 2:30 ~ ragg2 -i exec -a x86 -b 32 | rasm2 -d -
xor eax, eax
push eax
push 0x68732f2f
push 0x6e69622f
mov ebx, esp
push eax
push ebx
mov ecx, esp
cdq
mov al, 0xb
int 0x80
zsh: done      ragg2 -i exec -a x86 -b 32 |
zsh: exit 24   rasm2 -d -
jvoisin@kaa 2:30 ~ rasm2 -a z80 'ld hl,$1000;add a,(hl);inc hl'
218623
jvoisin@kaa 2:42 ~ rasm2 -L | wc -l
46
zsh: exit 1   rasm2 -L |
zsh: done    wc -l
jvoisin@kaa 2:43 ~ □
```

# Structures

```
[0x00000000]> pf.pe_nt_image_headers32 @ pe_nt_image_headers32
signature : 0x00000108 = PE
(pe_image_file_header)fileHeader : <struct>
  (pe_machine)machine : 0x0000010c = machine (enum) = 0x14c ; IMAGE_FILE_MACHINE_I386
  numberOfSections : 0x0000010e = 0x0005
  timeDateStamp : 0x00000110 = 0x52d829f8
  pointerToSymbolTable : 0x00000114 = 0x00000000
  numberOfSymbols : 0x00000118 = 0x00000000
  sizeOfOptionalHeader : 0x0000011c = 0x00e0
  (pe_characteristics)characteristics : 0x0000011e = characteristics (bitfield) = 0x00000122 : IMAGE_FILE_EXECUTABLE_IMAGE |
FILE_LARGE_ADDRESS_AWARE | IMAGE_FILE_32BIT_MACHINE
(pe_image_optional_header32)optionalHeader : <struct>
  (pe_magic)magic : 0x00000120 = magic (enum) = 0x10b ; IMAGE_NT_OPTIONAL_HDR32_MAGIC
  majorLinkerVersion : 0x00000122 = 10 ; 0x0a ; ''
  minorLinkerVersion : 0x00000123 = 0 ; 0x00 ; ''
  sizeOfCode : 0x00000124 = 0x001bce00
```

# Visual mode

- Ncurses-like
- Static
- Dynamic
- Analysis
- Try it, really.

```
Eval variables: (asm.arch)
```

```
asm.parser = x86.pseudo
asm.pseudo = false
asm.segoff = false
asm.size = false
asm.stackptr = false
asm.syntax = intel
asm.tabs = 0
asm.trace = false
asm.ucase = true
> asm.varsub = true
asm.xrefs = true
```

```
Selected: asm.varsub (Substitute variables in disassembly)
```

```
f (fcn) _start 42
0x004013e2 31ed ebp,ebp,^=
0x004013e4 4989d1 rdx,r9,=
0x004013e7 5e rsp,[8],rsi,=,8,rsp,+=
0x004013e8 4889e2 rsp,rdx,=
0x004013eb 4883e4f0 0xffffffff0,rsp,&=
```



# ROP

- Efficient
- Multi-arch
- Filterable output
- No ROP-chain builder

```
[0x004013e2]> /R mov eax, pop, pop, pop, ret
0x0040364a ret
0x00403644      89d8  mov  eax, ebx
0x00403646      5b   pop  rbx
0x00403647      5d   pop  rbp
0x00403648      415c pop  r12
0x0040364a      c3   ret

0x0040366a ret
0x00403664      89d8  mov  eax, ebx
0x00403666      5b   pop  rbx
0x00403667      5d   pop  rbp
0x00403668      415c pop  r12
0x0040366a      c3   ret

0x00403685 ret
0x0040367f      89d8  mov  eax, ebx
0x00403681      5b   pop  rbx
0x00403682      5d   pop  rbp
0x00403683      415c pop  r12
0x00403685      c3   ret

0x00403b0d ret
0x00403b04      b8ffffff  mov  eax, 0xffffffff
0x00403b09      5b   pop  rbx
0x00403b0a      5d   pop  rbp
0x00403b0b      415c pop  r12
0x00403b0d      c3   ret

[0x004013e2]> █
```

# Shellcode compiling

```
jvoisin@kaa 1:11 ~ cat /tmp/meh.c
int main() {
    write (1, "Hello!\n", 7);
    exit(0);
}
jvoisin@kaa 1:12 ~ ragg2-cc -x /tmp/meh.c 2>/dev/null | rasm2 -d - 2>/dev/null
jmp 0x2
dec eax
lea esi, [0x1d]
mov edi, 0x1
mov edx, 0x7
mov eax, 0x1
syscall
xor edi, edi
mov eax, 0x3c
syscall
xor eax, eax
ret
dec eax
gs insb
insb
outsd
and [edx], ecx
jvoisin@kaa 1:12 ~ ragg2-cc /tmp/meh.c 2>/dev/null
/tmp/meh.c.bin
jvoisin@kaa 1:12 ~ ./tmp/meh.c.bin
Hello!
jvoisin@kaa 1:12 ~ █
```

# Binary patching

- Assemblers
- Cracker's toolbag
- Cacheable
- Patchfiles

```
[0x00000000]> wo?
Usage: wo[asmdxoAr124] [hexpairs] @ addr[:bsize]
| wow [val] == write looped value (alias for 'wb')
| woa [val] += addition (f.ex: woa 0102)
| wos [val] -= subtraction
| wom [val] *= multiply
| wod [val] /= divide
| woe [from-to] [step] .. create sequence
| wox [val] ^= xor (f.ex: wox 0x90)
| woo [val] |= or
| woA [val] &= and
| wor [val] random bytes (alias for 'wr $b')
| wor [val] >>= shift right
| wo1 [val] <<= shift left
| wo2 [val] 2= 2 byte endian swap
| wo4 [val] 4= 4 byte endian swap
| woD [len] De Bruijn Pattern (syntax woD length @ addr)
| woO [len] De Bruijn Pattern Offset (syntax: woO value)
[0x00000000]> □
```

# Debugger

- Works in visual mode
- Several backends
- Classic features
- Tracing

```
[0x7fb084700210 185 /bin/true]> f tmp;sr s.. @ sym.stderr+-2079350864 # 0x7fb084700210
- offset -      0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0x7fff965a2d30  0100 0000 0000 0000 cd47 5a96 ff7f 0000 .....GZ.....
0x7fff965a2d40  0000 0000 0000 0000 d747 5a96 ff7f 0000 .....GZ.....
0x7fff965a2d50  2348 5a96 ff7f 0000 3548 5a96 ff7f 0000 #HZ.....5HZ.....
0x7fff965a2d60  5348 5a96 ff7f 0000 6248 5a96 ff7f 0000 SHZ.....bHZ.....
r15 0x00000000      r14 0x00000000      r13 0x00000000
r12 0x00000000      rbp 0x00000000      rbx 0x00000000
r11 0x00000000      r10 0x00000000     r9 0x00000000
r8 0x00000000       rax 0x00000000     rcx 0x00000000
rdx 0x00000000     rsi 0x00000000     rdi 0x7fff965a2d30
orax 0xffffffffffff rip 0x7fb084700213      rflags = 1I
rsp 0x7fff965a2d30
0x7fb084700210  4889e7      mov rdi, rsp
;-- rip:
0x7fb084700213  e818380000 call 0x7fb084703a30 ;[1]
0x7fb084703a30(unk) ; rip
0x7fb084700218  4989c4      mov r12, rax
0x7fb08470021b  8b05d71b280 mov eax, [rip+0x221bd7] ; 0x7fb084701df8
0x7fb084700221  5a        pop rdx
0x7fb084700222  488d24c4   lea rsp, [rsp+rax*8]
0x7fb084700226  29c2      sub edx, eax
0x7fb084700228  52        push rdx
0x7fb084700229  4889d6     mov rsi, rdx
0x7fb08470022c  4989e5     mov r13, rsp
0x7fb08470022f  4883e4f0   and rsp, 0xffffffffffff0
0x7fb084700233  48b3d261e2. mov rdi, [rip+0x221e26] ; 0x7fb084702060
0x7fb08470023a  498d4cd510 lea rcx, [r13+rdx*8+0x10] ; 0x00000010
0x7fb08470023f  498d5508   lea rdx, [r13+0x8]
0x7fb084700243  31ed      xor ebp, ebp
0x7fb084700245  e866ef0000 call 0x7fb08470f1b0 ;[2]
0x7fb08470f1b0(unk) ; rip
0x7fb08470024a  488d150ff30. lea rdx, [rip+0xf30f] ; 0x7fb08470f500
0x7fb084700251  4c89ec     mov rsp, r13
0x7fb084700254  41ffe4     jmp r12
0x7fb084700257  660f1f8400. o16 nop [rax+rax]
0x7fb084700260  488d0599d2. lea rax, [rip+0x222d99] ; 0x7fb084703000
```

# Conclusion

---

- Open
- Attracting contributors
- Full of features
- Steep learning curve

## Conclusion

---

Radare2 is nice.  
You should use it.

## Conclusion

---

We're doing a workshop a 14h o'clock in Fischbach!