Building an alternative to IDA

What about radare2?
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
radare in 2006
radare2 in 2009
written in pure C
350k LoC
several contributors
RSoC
# Platforms

<table>
<thead>
<tr>
<th>Runs on</th>
<th>Handles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>PE/COFF</td>
</tr>
<tr>
<td>Linux</td>
<td>ELF</td>
</tr>
<tr>
<td>BSD</td>
<td>Mach0</td>
</tr>
<tr>
<td>OSX</td>
<td>DEX/JAVA</td>
</tr>
<tr>
<td>Android</td>
<td>BIOS/TE</td>
</tr>
<tr>
<td>iOS</td>
<td>GB/GBA/DS</td>
</tr>
<tr>
<td>Smartwatch</td>
<td>XBOX</td>
</tr>
<tr>
<td>Browser</td>
<td>Plan9</td>
</tr>
</tbody>
</table>
What about a GUI?
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?

```c
int main(int argc, char* argv[]){
    if (argc > 1 && !strcmp(argv[1], "1337"))
        puts ("Lose");
    puts ("Win");
    return 0;
}
```
But what about graphs?
Analysis

- Functions detection
- Local var detection
- FLIRT integration
- (X)REF
- DWARF and PDB
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\x73\x68\x68\x2f\x62\x69\xe89\xe3\x50\x53\xe89\xe1\x99\xb0\x0b\xcd\x00"
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) = 0x1c ; IMAGE_FILE_MACHINE_I386
     numberOfSections : 0x0000010e = 0x0005
     timeZoneStamp : 0x00000010 = 0x52d829f8
     pointerToSymbolTable : 0x00000114 = 0x00000000
     numberOfSymbols : 0x00000118 = 0x00000000
     sizeOfOptionalHeader : 0x0000011c = 0x00e0
  (pe_characteristics)characteristics : 0x0000001e = characteristics (bitfield) = 0x00000122 : IMAGE_FILE_EXECUTABLE_IMAGE | FILE_LARGE_ADDRESS_AWARE | IMAGE_FILE_32BIT_MACHINE
  (pe_image_optional_header32)optionalHeader : <struct>
    (pe_magic)magic : 0x000000120 = magic (enum) = 0x10b ; IMAGE_NT_OPTIONAL_HDR32_MAGIC
    majorLinkerVersion : 0x000000122 = 10 ; 0x0a ; ''
    minorLinkerVersion : 0x000000123 = 0 ; 0x00 ; ''
    sizeOfCode : 0x000000124 = 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)

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

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

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

- Open
- Attracting contributors
- Full of features
- Steep learning curve
Radare2 is **nice**.
You should use it.
We’re doing a workshop at 14h o’clock in Fischbach!