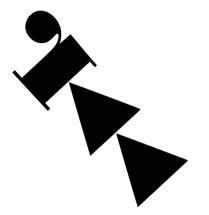
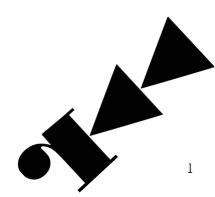


Function Argument Detection And Type Matching In Radare2

Oddcoder (@anoddcoder) Xvilka (@akochkov) R2Con 2016





GSoC

- Improving and fixing current types support
- Providing an uniform way to work with local variables
- Providing an uniform way to define argument types
- Very complex task involving a code refactoring
- http://radare.org/gsoc/2016/ideas.html#title_3

Task implications

- It required a change of the way to handle syscalls.
- Changing the 't" commands syntax and set.
- Changing the 'af* commands syntax and set.
- It was required to simplify fetching types information to radeco.

Future directions

- It's planned to add integration with signatures mechanisms.
- Supporting object-oriented programming data structures.
- Integration with demangling.
- Integration with DWARF/PDB/etc debug information.

\$ whoami

- Computer and communication engineering student.
- Hobbyist system software developer, exploitation, etc...
- Active developer at Radare2 project.
- Participant in Google Summer Of Code 2016.



A word of gratitude

Problems I was trying to solve

Auto detect formal arguments and local variables.

Gather type information about them.

Rules of the game No constraint solvers No debugging. Architecture independent



Challenges

A bit of everything is implemented.

That was my first time to work on large code bases.

I had to learn a lot on the fly.

Function Arguments/Locals

- Worked on x86 only.
- No 2 variables with same name in the whole binary.
- Only [ebp + stuff] could be manipulated.
- No stack based, or register based args/locals.

- It is arch independent.
- variables are local to their function.
- Adding support for stack based, or register based args/locals.



Old R2

```
; DATA XREF from 0x0040056d (entry0)
    0x00400646
                     55
                                    push rbp
                                    mov rbp, rsp
    0x00400647
                     4889e5
                                    sub rsp, 0x20
    0x0040064a
                     4883ec20
                                    mov dword [rbp - 0x14], edi
                     897dec
    0x0040064e
                     488975e0
                                    mov gword [rbp - 0x20], rsi
    0x00400651
    0x00400655
                     837dec01
                                     cmp dword [rbp - 0x14], 1 ; [0x1:4]=0x2464c45
,=< 0x00400659
                     7f11
                                    jg 0x40066c
                                    mov edi, str.Usage_echo_string_; "Usage echo <string>" @ 0x400770
    0x0040065b
                     bf70074000
    0x00400660
                     e89bfeffff
                                    call sym.imp.puts
                                    mov eax, 0
    0x00400665
                     b8000000000
                     eb64
                                    jmp 0x4006d0
.==< 0x0040066a
    ; JMP XREF from 0x00400659 (sym.main)
                                    mov rax, qword [rbp - 0x20]
'-> 0x0040066c
                     488b45e0
    0x00400670
                     4883c008
                                     add rax, 8
                                    mov rax, gword [rax]
    0x00400674
                     488b00
                                    mov rdi, rax
    0x00400677
                     4889c7
                     e891feffff
                                    call sym.imp.strlen
    0x0040067a
                                    mov dword [rbp - 4], eax
    0x0040067f
                     8945fc
    0x00400682
                     8b45fc
                                    mov eax, dword [rbp - 4]
    0x00400685
                     4898
                                    cdge
                     4889c7
                                    mov rdi, rax
    0x00400687
                                     call sym.imp.malloc
    0x0040068a
                     e8a1feffff
                     488945f0
                                    mov gword [rbp - 0x10], rax
    0x0040068f
                     8b45fc
                                    mov eax, dword [rbp - 4]
    0x00400693
                     4863d0
                                    movsxd rdx, eax
    0x00400696
    0x00400699
                     488b45e0
                                    mov rax, gword [rbp - 0x20]
                                    add rax, 8
    0x0040069d
                     4883c008
    0x004006a1
                     488b08
                                    mov rcx, gword [rax]
    0x004006a4
                     488b45f0
                                    mov rax, gword [rbp - 0x10]
    0x004006a8
                     4889ce
                                    mov rsi, rcx
                     4889c7
    0x004006ab
                                    mov rdi, rax
                     e83dfeffff
                                    call sym.imp.strncpy
    0x004006ae
    0x004006b3
                     488b45f0
                                    mov rax, gword [rbp - 0x10]
                     4889c7
    0x004006b7
                                    mov rdi, rax
                     e841feffff
                                    call sym.imp.puts
    0x004006ba
                                    mov rax, qword [rbp - 0x10]
    0x004006bf
                     488b45f0
                                    mov rdi, rax
    0x004006c3
                     4889c7
                     e815feffff
                                    call sym.imp.free
    0x004006c6
    0x004006cb
                     b800000000
                                    mov eax, 0
    ; JMP XREF from 0x0040066a (sym.main)
```

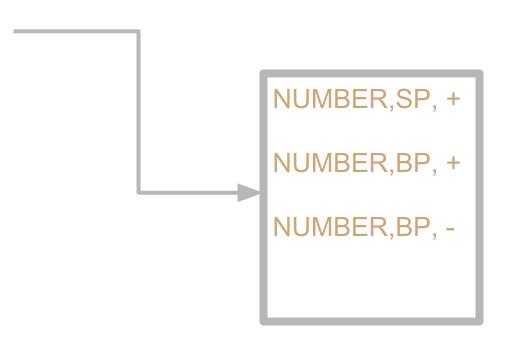
New R2

```
; var int local_10h @ rbp-0x10
    ; var int local_4h @ rbp-0x4
     DATA XREF from 0x0040056d (entry0)
    0x00400646
                    55
                                   push rbp
    0x00400647
                    4889e5
                                   mov rbp, rsp
                                   sub rsp. 0x20
    0x0040064a
                    4883ec20
                                   mov dword [rbp - local_14h], edi
    0x0040064e
                    897dec
                                   mov gword [rbp - local_20h], rsi
                    488975e0
    0x00400651
                                   cmp dword [rbp - local_14h], 1; [0x1:4]=0x2464c45
                    837dec01
    0x00400655
=< 0x00400659
                    7f11
                                   jg 0x40066c
                                   mov edi, str.Usage_echo_string_; "Usage echo <string>" @ 0x400770
    0x0040065b
                    bf70074000
                    e89bfeffff
                                   call sym.imp.puts
    0x00400660
                                   mov eax. 0
    0x00400665
                    b8000000000
                                   jmp 0x4006d0
==< 0x0040066a
                    eb64
    ; JMP XREF from 0x00400659 (sym.main)
                                   mov rax, qword [rbp - local_20h]
'-> 0x0040066c
                    488b45e0
                                   add rax. 8
    0x00400670
                    4883c008
                                   mov rax, gword [rax]
    0x00400674
                    488b00
                    4889c7
                                   mov rdi, rax
    0x00400677
                                   call sym.imp.strlen
    0x0040067a
                    e891feffff
                                   mov dword [rbp - local_4h], eax
                    8945fc
    0x0040067f
                                   mov eax, dword [rbp - local_4h]
                    8b45fc
    0x00400682
                    4898
    0x00400685
                                   cdge
                                   mov rdi, rax
    0x00400687
                    4889c7
                    e8a1feffff
                                   call sym.imp.malloc
    0x0040068a
    0x0040068f
                    488945f0
                                   mov gword [rbp - local_10h], rax
    0x00400693
                    8b45fc
                                   mov eax, dword [rbp - local_4h]
                                   movsxd rdx, eax
    0x00400696
                    4863d0
                                   mov rax, gword [rbp - local_20h]
                    488b45e0
                    4883c008
                                   add rax, 8
    0x0040069d
                                   mov rcx, gword [rax]
    0x004006a1
                    488b08
                    488b45f0
                                   mov rax, gword [rbp - local_10h]
    0x004006a4
                                   mov rsi, rcx
    0x004006a8
                    4889ce
                                   mov rdi, rax
    0x004006ab
                    4889c7
                                   call sym.imp.strncpy
                    e83dfeffff
    0x004006ae
                                   mov rax, qword [rbp - local_10h]
    0x004006b3
                    488b45f0
                    4889c7
                                   mov rdi, rax
    0x004006b7
                                   call sym.imp.puts
    0x004006ba
                    e841feffff
    0x004006bf
                    488b45f0
                                   mov rax, gword [rbp - local_10h]
    0x004006c3
                    4889c7
                                    mov rdi, rax
                                   call sym.imp.free
    0x004006c6
                    e815feffff
```

How does Automatic detection work?
It is all strings matching on ESIL.

1. Locate candidate var.

1. Locate candidate var.



- 1. Locate candidate var.
- 2. Identify type of candidate var.

- 1. Locate candidate var.
- 2. Identify type of candidate var.

```
NUMBER,BP, + (arg)
```

NUMBER, BP, - (*local*)

NUMBER, SP, + (?)

- 1. Locate candidate var.
- 2. Identify type of candidate var.
- 3. Naming and registering var.

- 1. Locate candidate var.
- 2. Identify type of candidate var.
- 3. Naming and registering var.

local_(Number)h [_counter]

var_(Number)h [_counter]

Type Matching

- Implementing type hints for standard function.
- Depends on calling convention of that function.
- Needs emulation to avoid confusion with stack & registers tracing.

What is Calling Convention?

- It is the way functions call another another function.
- How arguments are arranged in the stack?
- Who cleans the callee stack?
- Where is the return value placed?

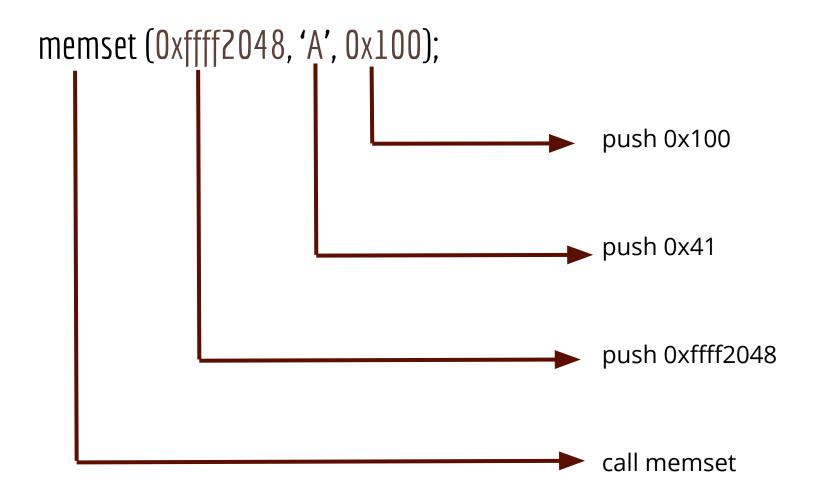
Cdecl

- Most common calling convention in linux i386.
- Arguments are pushed right to left on the stack.
- Return value is put in eax.
- The caller is the one who cleans the stack.

Cdecl

memset (0xffff2048, 'A', 0x100);

Cdecl



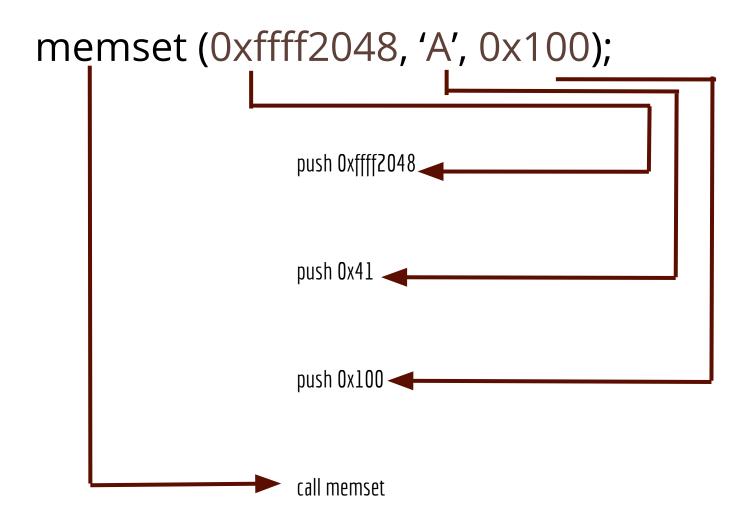
Stdcall

- Popular on i386 windows machines.
- Arguments pushed left to right on the stack.
- Return value is also put in eax.
- Callee cleans its own stack frame.

Stdcall

memset (0xffff2048, 'A', 0x100);

Stdcall



X64 Windows ABI

- Arguments passed into RCX, RDX, R8, R9, then the remaining arguments are put on stack right to left.
- Return value is put into rax

And much more

- And there are lots of other calling conventions.
- Calling conventions for each architecture, and for each Operating system.
- Even some programming languages and some compilers got their own calling conventions (ABI)

Why would we care?

We already had this.

```
; DATA XREF from 0x0040056d (entry0)
    0x00400646
                     55
                                    push rbp
                     4889e5
                                    mov rbp, rsp
    0x00400647
                                    sub rsp, 0x20
                     4883ec20
    0x0040064a
                                    mov dword [rbp - 0x14], edi
                     897dec
    0x0040064e
                     488975e0
                                    mov gword [rbp - 0x20], rsi
    0x00400651
    0x00400655
                     837dec01
                                    cmp dword [rbp - 0x14], 1 ; [0x1:4]=0x2464c45
 ,=< 0x00400659
                     7f11
                                    jg 0x40066c
                                    mov edi, str.Usage_echo_string_; "Usage echo <string>" @ 0x400770
    0x0040065b
                     bf70074000
                     e89bfeffff
                                    call sym.imp.puts
    0x00400660
                                    mov eax, 0
    0x00400665
                     b800000000
.==< 0x0040066a
                     eb64
                                    jmp 0x4006d0
    ; JMP XREF from 0x00400659 (sym.main)
                                    mov rax, gword [rbp - 0x20]
| '-> 0x0040066c
                     488b45e0
    0x00400670
                     4883c008
                                     add rax, 8
    0x00400674
                     488b00
                                    mov rax, gword [rax]
    0x00400677
                     4889c7
                                    mov rdi, rax
    0x0040067a
                     e891feffff
                                    call sym.imp.strlen
    0x0040067f
                     8945fc
                                    mov dword [rbp - 4], eax
                                    mov eax, dword [rbp - 4]
    0x00400682
                     8b45fc
                     4898
                                    cdge
    0x00400685
                     4889c7
                                    mov rdi, rax
    0x00400687
                     e8a1feffff
                                    call sym.imp.malloc
    0x0040068a
                                    mov gword [rbp - 0x10], rax
                     488945f0
    0x0040068f
                                    mov eax, dword [rbp - 4]
                     8b45fc
    0x00400693
                     4863d0
                                    movsxd rdx, eax
    0x00400696
                                    mov rax, qword [rbp - 0x20]
    0x00400699
                     488b45e0
                                    add rax, 8
    0x0040069d
                     4883c008
    0x004006a1
                     488b08
                                    mov rcx, gword [rax]
                                    mov rax, qword [rbp - 0x10]
    0x004006a4
                     488b45f0
    0x004006a8
                     4889ce
                                    mov rsi, rcx
                     4889c7
                                    mov rdi, rax
    0x004006ab
    0x004006ae
                     e83dfeffff
                                    call sym.imp.strncpy
                                    mov rax, gword [rbp - 0x10]
    0x004006b3
                     488b45f0
                     4889c7
                                    mov rdi, rax
    0x004006b7
                                    call sym.imp.puts
    0x004006ba
                     e841feffff
                                    mov rax, qword [rbp - 0x10]
    0x004006bf
                     488b45f0
    0x004006c3
                                    mov rdi, rax
                     4889c7
                     e815feffff
                                    call sym.imp.free
    0x004006c6
    0x004006cb
                     b800000000
                                    mov eax, 0
    ; JMP XREF from 0x0040066a (sym.main)
```

And we turned it into that

```
var int local_10h @ rbp-0x10
     var int local_4h @ rbp-0x4
     DATA XREF from 0x0040056d (entry0)
    0x00400646
                    55
                                   push rbp
                                   mov rbp, rsp
    0x00400647
                    4889e5
                                   sub rsp. 0x20
    0x0040064a
                    4883ec20
                                   mov dword [rbp - local_14h], edi
    0x0040064e
                    897dec
                                   mov gword [rbp - local_20h], rsi
                    488975e0
    0x00400651
                                   cmp dword [rbp - local 14h], 1 : [0x1:4]=0x2464c45
    0x00400655
                    837dec01
                    7f11
                                   jg 0x40066c
 =< 0x00400659
                    bf70074000
                                   mov edi, str.Usage_echo_string_; "Usage echo <string>" @ 0x400770
    0x0040065b
    0x00400660
                    e89bfeffff
                                   call sym.imp.puts
    0x00400665
                    b8000000000
                                   mov eax. 0
==< 0x0040066a
                    eb64
                                   imp 0x4006d0
    ; JMP XREF from 0x00400659 (sym.main)
                                   mov rax, gword [rbp - local_20h]
'-> 0x0040066c
                    488b45e0
    0x00400670
                    4883c008
                                   add rax. 8
                                   mov rax, gword [rax]
    0x00400674
                    488b00
                    4889c7
                                   mov rdi, rax
    0x00400677
                                   call sym.imp.strlen
    0x0040067a
                    e891feffff
                    8945fc
                                   mov dword [rbp - local_4h], eax
    0x0040067f
                                   mov eax, dword [rbp - local_4h]
    0x00400682
                    8b45fc
    0x00400685
                    4898
                                   cdae
                                   mov rdi, rax
    0x00400687
                    4889c7
                                   call sym.imp.malloc
    0x0040068a
                    e8a1feffff
                                   mov gword [rbp - local_10h], rax
                    488945f0
    0x0040068f
                                   mov eax, dword [rbp - local_4h]
                    8b45fc
    0x00400693
                    4863d0
                                   movsxd rdx, eax
    0x00400696
                                   mov rax, gword [rbp - local_20h]
    0x00400699
                    488b45e0
    0x0040069d
                    4883c008
                                   add rax. 8
                                   mov rcx, gword [rax]
    0x004006a1
                    488b08
                    488b45f@
                                   mov rax, gword [rbp - local_10h]
    0x004006a4
    0x004006a8
                    4889ce
                                   mov rsi, rcx
    0x004006ab
                    4889c7
                                   mov rdi, rax
    0x004006ae
                    e83dfeffff
                                   call sym.imp.strncpy
                                   mov rax, gword [rbp - local_10h]
    0x004006b3
                    488b45f0
    0x004006b7
                    4889c7
                                   mov rdi, rax
    0x004006ba
                    e841feffff
                                   call sym.imp.puts
    0x004006bf
                    488b45f@
                                   mov rax, gword [rbp - local_10h]
                    4889c7
                                   mov rdi, rax
                                   call sym.imp.free
    0x004006c6
                    e815feffff
```

And this is our final goal

```
; DATA XREE Trom 0X00400560 (entry0)
                    55
                                   push rbp
    0x00400646
    0x00400647
                    4889e5
                                   mov rbp, rsp
    0x0040064a
                    4883ec20
                                   sub rsp. 0x20
                                   mov dword [rbp - local_14h], edi
    0x0040064e
                    897dec
                    488975e0
                                   mov gword [rbp - local_20h], rsi
    0x00400651
    0x00400655
                    837dec@1
                                   cmp dword [rbp - local_14h], 1; [0x1:4]=0x2464c45
=< 0x00400659
                    7f11
                                   ig 0x40066c
                                   mov edi, str. Usage echo string; "Usage echo <string>" @ 0x400770; con
    0x0040065b
                    bf70074000
                    e89bfeffff
                                   call sym.imp.puts
                                                                ;[3]
    0x00400660
                    b8000000000
                                   mov eax, 0
    0x00400665
==< 0x0040066a
                    eb64
                                   imp 0x4006d0
                                   mov rax, gword [rbp - local 20h]
'-> 0x0040066c
                    488b45e0
                                   add rax. 8
    0x00400670
                    4883c008
                                   mov rax, qword [rax]
    0x00400674
                    488b00
                                                                ; const char * s
    0x00400677
                    4889c7
                                   mov rdi, rax
                    e891feffff
                                   call sym.imp.strlen
                                                                :[5]
    0x0040067a
                                   mov dword [rbp - local_4h], eax
    0x0040067f
                    8945fc
                                   mov eax, dword [rbp - local_4h]
                    8b45fc
                    4898
                                   cdae
                    4889c7
                                   mov rdi, rax
                                                                ; size_t size
                                   call sym.imp.malloc
                    e8alfeffff
    0x0040068a
                                                                ; [6]
                                   mov gword [rbp - local_10h], rax
    0x0040068f
                    488945f0
                    8b45fc
                                   mov eax, dword [rbp - local_4h]
    0x00400693
                    4863d0
                                   movsxd rdx, eax
                                                                ; size_t n
                                   mov rax, gword [rbp - local_20h]
    0x00400699
                    488b45e0
                    4883c008
                                   add rax, 8
                                   mov rcx, gword [rax]
    0x004006a1
                    488b08
                    488b45f0
                                   mov rax, gword [rbp - local_10h]
    0x004006a4
    0x004006a8
                    4889ce
                                   mov rsi, rcx
                                                                ; const char * src
                                                                ; char * dest
    0x004006ab
                    4889c7
                                   mov rdi, rax
                                   call sym.imp.strncpy
    0x004006ae
                    e83dfeffff
                                   mov rax, gword [rbp - local 10h]
                    488b45f0
    0x004006b3
                                   mov rdi, rax
                    4889c7
                                                                : const char * s
    0x004006b7
                                   call sym.imp.puts
    0x004006ba
                    e841feffff
                                   mov rax, gword [rbp - local 10h]
                    488b45f0
    0x004006bf
                    4889c7
                                   mov rdi, rax
                                                                  cum imp libe start main-0v40 . [0]
```

1. Read next instruction.

- 1. Read next instruction.
- 2. Record that instruction for traceback.

- 1. Read next instruction.
- 2. Record that instruction for traceback.
- 3. Record stack and register accesses.

- 1. Read next instruction.
- 2. Record that instruction for traceback.
- 3. Record stack and register accesses.
- 4. Go to step 1 **4**.



- 1. Read next instruction.
- 2. Record that instruction for traceback.
- 3. Record stack and register accesses.
- 4. Go to step 1 **4**.



→ This loop is interrupted when we find a call instruction

Interesting stuff starts when we catch a call instruction.

1- Grab function's definition from types database.

- 1- Grab function's definition from types database.
- 2- Do the math to calculate the location of all arguments.

- 1- Grab function's definition from types database.
- 2- Do the math to calculate the location of all arguments.
- 3 Search the trace log for writing to these locations.

Now it is up to us what to do with all that information.

SHOW TIME

Let's add types SDB(s)



See also

GitHub issue https://github.com/radare/radare2/issues/3655

RT (Radare Today) article http://radare.today/posts/GSOC-The-last-commit-213c6f/

Calling convention db docs:

https://github.com/radare/radare2/blob/master/doc/calling-conventions.md

Types db docs: https://github.com/radare/radare2/blob/master/doc/types.md