# GSoC’20

## Type analysis improvements?

# Who am I?

- Karel Hajek
- Second year CS student at Brno University of Technology
- I like to know how things really work

# GSoC - Google Summer of Code

- Project was “Type analysis improvements”, but…
- Started with debug formats at kinda stuck with it
- So in the end the work done was on:
  - DWARF
  - PDB
  - Itanium vtables and RTTI
- Started working on type analysis on the very end, but that didn’t have result yet when making these slides.

# DWARF - Load type information

## Challenges:

- State of the DWARF code base was unknown
- Lot of the parsing didn’t really work
- DWARF is quite complex with many standard versions (DWARF 2 starting at 100 pages, growing up to DWARF 5 with ~500 pages)
- Where to start?

# DWARF

## Started by writing tests:

- Found the problems and started fixing them
- Once I got comfortable with the codebase I started parsing .debug_info section
- Support for parsing of all of the DWARF versions (you can see what all we can parse with `id` command)
- Started processing type information out of the parsed section

# DWARF

```
 Compilation Unit @ offset 0x0:
  Length:        0x494
  Version:       4
  Abbrev Offset: 0x0
  Pointer Size:  8

<0xb>: Abbrev Number: 1    (DW_TAG_compile_unit)
     DW_AT_producer            : (indirect string, offset: 0xc3): GNU C++14 9.3.0 -mtune=generic -march
synchronous-unwind-tables -fstack-protector-strong -fstack-clash-protection -fcf-protection
     DW_AT_language            : 4   (C++)
     DW_AT_name                : (indirect string, offset: 0x7c): main.cpp
     DW_AT_comp_dir            : (indirect string, offset: 0x1e9): /home/hound/Projects/r2test/dwarf/cp
     DW_AT_ranges              : <0x0>
     DW_AT_low_pc              : 0x0
     DW_AT_stmt_list           : <0x0>
<0x29>: Abbrev Number: 2    (DW_TAG_structure_type)
     DW_AT_name                : (indirect string, offset: 0x0): MyStruct
     DW_AT_byte_size           : 16
     DW_AT_decl_file           : 1
     DW_AT_decl_line           : 1
     DW_AT_decl_column         : 8
     DW_AT_siblings            : <0x68>
<0x36>: Abbrev Number: 3    (DW_TAG_member)
     DW_AT_name                : a
     DW_AT_decl_file           : 1
     DW_AT_decl_line           : 2
     DW_AT_decl_column         : 33
     DW_AT_type                : <0x6e>
     DW_AT_data_member_location : 0
<0x41>: Abbrev Number: 3    (DW_TAG_member)
     DW_AT_name                : b
     DW_AT_decl_file           : 1
     DW_AT_decl_line           : 3
     DW_AT_decl_column         : 33
     DW_AT_type                : <0x8a>
     DW_AT_data_member_location : 8
```

```
<0x4c>: Abbrev Number: 4    (DW_TAG_subprogram)
     DW_AT_external            : 1
     DW_AT_name                : (indirect string, offset: 0x0): MyStruct
     DW_AT_decl_file           : 1
     DW_AT_decl_line           : 4
     DW_AT_decl_column         : 2
     DW_AT_linkage_name        : (indirect string, offset: 0x18b): _ZN8MyStructC4EPVh
     DW_AT_declaration         : 1
     DW_AT_object_pointer      : <0x5c>
<0x5c>: Abbrev Number: 5    (DW_TAG_formal_parameter)
     DW_AT_type                : <0x8f>
     DW_AT_artificial          : 1
<0x61>: Abbrev Number: 6    (DW_TAG_formal_parameter)
     DW_AT_type                : <0x84>
<0x66>: Abbrev Number: 0    (DW_TAG_null_entry)
<0x67>: Abbrev Number: 0    (DW_TAG_null_entry)
<0x68>: Abbrev Number: 7    (DW_TAG_pointer_type)
     DW_AT_byte_size           : 8
     DW_AT_type                : <0x7a>
<0x6e>: Abbrev Number: 8    (DW_TAG_volatile_type)
     DW_AT_type                : <0x68>
<0x73>: Abbrev Number: 9    (DW_TAG_base_type)
     DW_AT_byte_size           : 1
     DW_AT_encoding            : 8
     DW_AT_name                : (indirect string, offset: 0x32): unsigned char
<0x7a>: Abbrev Number: 10   (DW_TAG_const_type)
     DW_AT_type                : <0x73>
<0x7f>: Abbrev Number: 8    (DW_TAG_volatile_type)
     DW_AT_type                : <0x73>
<0x84>: Abbrev Number: 7    (DW_TAG_pointer_type)
     DW_AT_byte_size           : 8
     DW_AT_type                : <0x7f>
<0x8a>: Abbrev Number: 10   (DW_TAG_const_type)
     DW_AT_type                : <0x84>
```

# DWARF

## Types:

- Processed automatically with DWARF parsing (just like line information)
- Tries to emulate C syntax during type parsing (doesn’t look that well languages like Go)

# DWARF

```
[0x00001060]> tc
union MyUnion {
        int x;
        long int y;
        short int kk;
        char ch;
        char *str;
        char[50] buf;
};
struct ForwardDeclaration {
};
struct MyClass {
        int myNum;
        char const *myString;
        char const & charRef;
        char[100][50] buffer;
        char const ***x;
        void (*)() funPtr1;
        void (*)() funPtr2;
        int && (*)() funPtr3;
        void * (*)()funPtr4;
        char & (*)() funPtr5;
        char & (****)() funPtr6;
};
struct MyStruct {
        unsigned char const * volatilea;
        unsigned char volatile * constb;
};
typedef MyStruct OtherStruct;
enum MyEnum {
        first = 1,
        third = 3,
        last = 99,
        neg = -1,
        large_neg = -130
};
enum ScopedEnum {
        a = 97,
        b = 98,
        z = 122
};
[0x00001060]> tk~MyStruct
MyStruct=struct
struct.MyStruct=a,b
struct.MyStruct.!size=128
struct.MyStruct.a=unsigned char const * volatile,0,0
struct.MyStruct.b=unsigned char volatile * const,8,0
typedef.OtherStruct=MyStruct
[0x00001060]>
```

```
1  struct MyStruct {
2      unsigned char const * volatile a;
3      volatile unsigned char * const b;
4      MyStruct(volatile unsigned char * const arg) : b(arg){}
5  };
6
7  enum MyEnum {
8      first = 1,
9      third = 3,
10     last = 99,
11     neg = -1,
12     large_neg = -130,
13 };
14
15 union MyUnion
16 {
17     int x;
18     long y;
19     short kk;
20     char ch;
21     char *str;
22     char buf[50];
23 };
24
25 enum class ScopedEnum : unsigned char {
26     a = 'a',
27     b = 'b',
28     z = 'z',
29 };
30
31 typedef MyStruct OtherStruct;
32
33 struct ForwardDeclaration;
34
35 class MyClass {
36   public:
37     int myNum;
38     const char *myString;
39     const char &charRef;
40     char buffer[100][50];
41     const char ***x;
42     void (*funPtr1)();
43     void (*funPtr2)(int);
44     int &&(*funPtr3)(int);
45     void *(*funPtr4)(int, volatile int,const char);
46     char &(*funPtr5)(int*, const char*);
47     char &(****funPtr6)(void (*)(void *, void *), const char*);
48     MyClass(const char &ref) : charRef(ref) {}
49 };
```

# DWARF

## Function information:

- Loading function address
- Function signature
- Arguments and variables
- Processed and saved into Sdb when DWARF is loaded, applied now with `aaa`
- Support for BP, SP and register based locations + globals at fixed address

# DWARF

```
<0x5eaa>: Abbrev Number: 156  (DW_TAG_subprogram)
     DW_AT_external            : 1
     DW_AT_name                : (indirect string, offset: 0x11522): main
     DW_AT_decl_file           : 3
     DW_AT_decl_line           : 20
     DW_AT_decl_column         : 5
     DW_AT_type                : <0x2f53>
     DW_AT_low_pc              : 0x401339
     DW_AT_high_pc             : 411
     DW_AT_frame_base          : 1 byte block: 0x9c
     DW_AT_GNU_all_tail_call_sites : 1
     DW_AT_siblings            : <0x5f4b>
<0x5ecd>: Abbrev Number: 49   (DW_TAG_variable)
     DW_AT_name                : zoo
     DW_AT_decl_file           : 3
     DW_AT_decl_line           : 21
     DW_AT_decl_column         : 6
     DW_AT_type                : <0x47c2>
     DW_AT_location            : 3 byte block: 0x91 0x80 0x7f
<0x5edd>: Abbrev Number: 49   (DW_TAG_variable)
     DW_AT_name                : bat
     DW_AT_decl_file           : 3
     DW_AT_decl_line           : 22
     DW_AT_decl_column         : 7
     DW_AT_type                : <0x4b39>
     DW_AT_location            : 2 byte block: 0x91 0x50
<0x5eec>: Abbrev Number: 49   (DW_TAG_variable)
     DW_AT_name                : cat
     DW_AT_decl_file           : 3
     DW_AT_decl_line           : 23
     DW_AT_decl_column         : 7
     DW_AT_type                : <0x490a>
     DW_AT_location            : 2 byte block: 0x91 0x48
```

```
<0x5efb>: Abbrev Number: 49   (DW_TAG_variable)
     DW_AT_name                : dog
     DW_AT_decl_file           : 3
     DW_AT_decl_line           : 24
     DW_AT_decl_column         : 7
     DW_AT_type                : <0x49bb>
     DW_AT_location            : 2 byte block: 0x91 0x40
<0x5f0a>: Abbrev Number: 79   (DW_TAG_variable)
     DW_AT_name                : (indirect string, offset: 0xf99a): bird
     DW_AT_decl_file           : 3
     DW_AT_decl_line           : 25
     DW_AT_decl_column         : 8
     DW_AT_type                : <0x4b50>
     DW_AT_location            : 3 byte block: 0x91 0xb8 0x7f
<0x5f1a>: Abbrev Number: 79   (DW_TAG_variable)
     DW_AT_name                : (indirect string, offset: 0x115ba): animal
     DW_AT_decl_file           : 3
     DW_AT_decl_line           : 30
     DW_AT_decl_column         : 10
     DW_AT_type                : <0x4609>
     DW_AT_location            : 3 byte block: 0x91 0xb0 0x7f
<0x5f2a>: Abbrev Number: 157  (DW_TAG_lexical_block)
     DW_AT_low_pc              : 0x401418
     DW_AT_high_pc             : 50
<0x5f3c>: Abbrev Number: 49   (DW_TAG_variable)
     DW_AT_name                : i
     DW_AT_decl_file           : 3
     DW_AT_decl_line           : 31
     DW_AT_decl_column         : 14
     DW_AT_type                : <0x2e80>
     DW_AT_location            : 2 byte block: 0x91 0x58
<0x5f49>: Abbrev Number: 0    (DW_TAG_null_entry)
<0x5f4a>: Abbrev Number: 0    (DW_TAG_null_entry)
```

# DWARF

- Uses anal/dwarf namespace in Sdb
- Parsed into sdb:

```
[0x00401140]> k anal/dwarf/*~main
fcn.main.addr=0x401339
fcn.main.name=main
fcn.main.sig=int main();
fcn.main.var.animal=b,-64,Mammal *
fcn.main.var.bat=b,-32,Bat *
fcn.main.var.bird=b,-56,Bird *
fcn.main.var.cat=b,-40,Cat *
fcn.main.var.dog=b,-48,Dog *
fcn.main.var.i=b,-24,size_t
fcn.main.var.zoo=b,-112,Zoo
fcn.main.vars=zoo,bat,cat,dog,bird,animal,i
main=fcn
```

- And also with the parsed type example, that radare2 can use

```
[0x00401140]> tc struct.Zoo
struct Zoo {
        vector<Mammal* animals;
};
```

# DWARF

- Applied information in disassembly
- Signature as a comment (because it works for variety of languages it can’t be used by C parser for function signatures

# DWARF

```
[0x00401140]> pd 20 @ main
            ; DATA XREF from entry0 @ 0x401161
            ;-- main:
┌ 301: int dbg.main (int argc, char **argv, char **envp);
            ; var Zoo zoo @ rbp-0x70
            ; var int64_t var_58h @ rbp-0x58
            ; var int64_t var_50h @ rbp-0x50
            ; var int64_t var_48h @ rbp-0x48
            ; var Mammal *animal @ rbp-0x40
            ; var Bird *bird @ rbp-0x38
            ; var Dog *dog @ rbp-0x30
            ; var Cat *cat @ rbp-0x28
            ; var Bat *bat @ rbp-0x20
            ; var size_t i @ rbp-0x18
            0x00401339      55             push rbp                    ; int main();
            0x0040133a      4889e5         mov rbp, rsp
            0x0040133d      4154           push r12
            0x0040133f      53             push rbx
            0x00401340      4883ec60       sub rsp, 0x60
            0x00401344      488d4590       lea rax, [zoo.animals]
            0x00401348      4889c7         mov rdi, rax                ; int64_t arg1
            0x0040134b      e8ee030000     call method Zoo::Zoo()      ; dbg.Zoo::Zoo()
            0x00401350      bf30000000     mov edi, 0x30               ; '0' ; 48
            0x00401355      e856fdffff     call sym operator new(unsigned long) ; sym.imp.operator_new_unsigned_long
            0x0040135a      4889c3         mov rbx, rax
            0x0040135d      4889df         mov rdi, rbx                ; int64_t arg1
            0x00401360      e865020000     call method Bat::Bat()      ; dbg.Bat::Bat()
            0x00401365      48895de0       mov qword [bat], rbx
            0x00401369      bf10000000     mov edi, 0x10               ; 16
            0x0040136e      e83dfdffff     call sym operator new(unsigned long) ; sym.imp.operator_new_unsigned_long
            0x00401373      4889c3         mov rbx, rax
            0x00401376      4889df         mov rdi, rbx                ; int64_t arg1
            0x00401379      e818030000     call method Cat::Cat()      ; dbg.Cat::Cat()
            0x0040137e      48895dd8       mov qword [cat], rbx
```

# DWARF

- One more example:
- Creating flags for globals

```
<0x1c1>: Abbrev Number: 14   (DW_TAG_subprogram)
     DW_AT_external            : 1
     DW_AT_name                : (indirect string, offset: 0xe83): create_donkey
     DW_AT_decl_file           : 1
     DW_AT_decl_line           : 33
     DW_AT_decl_column         : 16
     DW_AT_prototyped          : 1
     DW_AT_type                : <0x1bb>
     DW_AT_low_pc              : 0x1171
     DW_AT_high_pc             : 86
     DW_AT_frame_base          : 1 byte block: 0x9c
     DW_AT_GNU_all_tail_call_sites : 1
     DW_AT_siblings            : <0x236>
<0x1e3>: Abbrev Number: 15   (DW_TAG_formal_parameter)
     DW_AT_name                : (indirect string, offset: 0x3ad): faceLength
     DW_AT_decl_file           : 1
     DW_AT_decl_line           : 33
     DW_AT_decl_column         : 35
     DW_AT_type                : <0x38>
     DW_AT_location            : 2 byte block: 0x91 0x5c
<0x1f2>: Abbrev Number: 15   (DW_TAG_formal_parameter)
     DW_AT_name                : (indirect string, offset: 0x1c4): owner
     DW_AT_decl_file           : 1
     DW_AT_decl_line           : 33
     DW_AT_decl_column         : 61
     DW_AT_type                : <0x111>
     DW_AT_location            : 2 byte block: 0x91 0x50
<0x201>: Abbrev Number: 15   (DW_TAG_formal_parameter)
     DW_AT_name                : (indirect string, offset: 0x484f): name
     DW_AT_decl_file           : 1
     DW_AT_decl_line           : 33
     DW_AT_decl_column         : 74
     DW_AT_type                : <0x70>
     DW_AT_location            : 2 byte block: 0x91 0x48
<0x210>: Abbrev Number: 12   (DW_TAG_variable)
     DW_AT_name                : (indirect string, offset: 0x2648): default_leg_count
     DW_AT_decl_file           : 1
     DW_AT_decl_line           : 34
     DW_AT_decl_column         : 13
     DW_AT_type                : <0x38>
     DW_AT_location            : 9 byte block: 0x03 0x10 0x40 0x00 0x00 0x00 0x00 0x00 0x00
<0x226>: Abbrev Number: 12   (DW_TAG_variable)
     DW_AT_name                : (indirect string, offset: 0xe8a): donkey
     DW_AT_decl_file           : 1
     DW_AT_decl_line           : 35
     DW_AT_decl_column         : 17
     DW_AT_type                : <0x1bb>
     DW_AT_location            : 2 byte block: 0x91 0x68
<0x235>: Abbrev Number: 0    (DW_TAG_null_entry)
```

# DWARF

```
[0x00001171]> pdf @ dbg.create_donkey
            ; CALL XREF from dbg.main @ 0x11e4
            ;-- create_donkey:
┌ 86: dbg.create_donkey (void *arg1, void *arg2, void *arg3);
│           ; var char *name @ rbp-0x28
│           ; var Human *owner @ rbp-0x20
│           ; var int faceLength @ rbp-0x14
│           ; var Donkey *donkey @ rbp-0x8
│           ; arg void *arg1 @ rdi
│           ; arg void *arg2 @ rsi
│           ; arg void *arg3 @ rdx
│           0x00001171      f30f1efa       endbr64                    ; Donkey * create_donkey(int faceLength,Human * owner,char * name);
│           0x00001175      55             push rbp
│           0x00001176      4889e5         mov rbp, rsp
│           0x00001179      4883ec30       sub rsp, 0x30
│           0x0000117d      897dec         mov dword [faceLength], edi ; arg1
│           0x00001180      488975e0       mov qword [owner], rsi      ; arg2
│           0x00001184      488955d8       mov qword [name], rdx       ; arg3
│           0x00001188      bf18000000     mov edi, 0x18               ; size_t size
│           0x0000118d      e8befeffff     call sym.imp.malloc         ;  void *malloc(size_t size)
│           0x00001192      488945f8       mov qword [donkey], rax
│           0x00001196      488b45f8       mov rax, qword [donkey]
│           0x0000119a      8b55ec         mov edx, dword [faceLength]
│           0x0000119d      895004         mov dword [rax + 4], edx
│           0x000011a0      488b45f8       mov rax, qword [donkey]
│           0x000011a4      488b55e0       mov rdx, qword [owner]
│           0x000011a8      48895010       mov qword [rax + 0x10], rdx
│           0x000011ac      488b45f8       mov rax, qword [donkey]
│           0x000011b0      488b55d8       mov rdx, qword [name]
│           0x000011b4      48895008       mov qword [rax + 8], rdx
│           0x000011b8      8b15522e0000   mov edx, dword [global_default_leg_count] ; [0x4010:4]=4
│           0x000011be      488b45f8       mov rax, qword [donkey]
│           0x000011c2      8910           mov dword [rax], edx
│           0x000011c4      90             nop
│           0x000011c5      c9             leave
└           0x000011c6      c3             ret
[0x00001171]>
```

# DWARF

TODOs:

- A lot more tests!
- Support for new things from DWARF 5
- Support for DWARF in separate file
- More DWARF register mappings for different arches
- Parsing of additional sections (.debug_frame for locations)
- Optimalizations
- Getting and applying more information, calling conventions, inheritance etc.?

If you find any problems, make a github issue or mention me on telegram!

# PDB

- Similar to DWARF, but only types
- Parsing fixes, lot of refactoring

```
[0x140001014]> idpi | head -n20
struct MyStruct { // size 0x10
  const uint8_t* a; // offset +0x0
  volatile uint8_t* b; // offset +0x8
  method MyStruct MyStruct; // offset +0x0
};
enum ScopedEnum { // type: uint8_t
  a = 97,
  b = 98,
  z = 122,
};
enum MyEnum { // type: int32_t
  first = 1,
  third = 3,
  last = 99,
};
struct MyClass { // size 0x13d8
  int32_t myNum; // offset +0x0
  const char* myString; // offset +0x8
  const char* charRef; // offset +0x10
  char[50][50] buffer; // offset +0x18
[0x140001014]>
```

# Vtables, RTTI (Itanium)

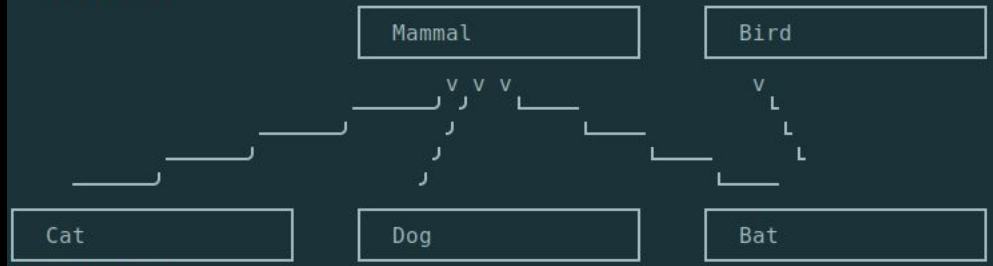
- Changed vtable detection heuristics
- Added fallback option to RTTI parsing (parsing as much sane values possible) so it doesn't depend on type_info name symbol
- Integrated RTTI inheritance information into `ac`

```
[hound@odin:~/.../r2test/talk_demos] r2 a.out
 -- For a full documentation see `r2 -qc iz /lib/libr_core.so`
[0x00401140]> avrr
[0x00401140]> acl
Bat: Mammal, Bird
Bird
Cat: Mammal
Dog: Mammal
Mammal
[0x00401140]>
```

# Vtables, RTTI (Itanium)

- Also added inheritance graph to visualize the class structure

```
[0x00401140]> acll
Bat: Mammal, Bird
  (vtable at 0x403108)
  (vtable at 0x403120)
  virtual_0 @ 0x40163c (vtable + 0x0)
Bird
  (vtable at 0x403138)
  virtual_0 @ 0x4015ac (vtable + 0x0)
Cat: Mammal
  (vtable at 0x4030d8)
  virtual_0 @ 0x4016cc (vtable + 0x0)
Dog: Mammal
  (vtable at 0x4030f0)
  virtual_0 @ 0x401678 (vtable + 0x0)
Mammal
  (vtable at 0x403150)
  virtual_0 @ 0x401564 (vtable + 0x0)
[0x00401140]> acg
```




```
[0x00401140]>
```

# Thanks!

- Thanks to GSoC for the opportunity
- Thanks to Radare2 team for all the help! (especially XVilka for mentoring)
- I hope I can stick around for a bit longer

- All my PRs https://github.com/radareorg/radare2/pulls?q=is%3Apr+author%3AHoundThe

- My https://github.com/HoundThe