An Open-Source Machine-Code Decompiler

Peter Matula

Marek Milkovič

Who Are We?

- Peter Matula
  - Senior software developer @Avast (previously @AVG)
  - Main developer of the RetDec decompiler
  - Developing reversing tools for 6 years
  - Love rock climbing & beer
  - peter.matula@avast.com

- Marek Milkovič
  - Software developer @Avast (previously @AVG)
  - Works on preprocessing stage of the RetDec decompiler
  - Works on YARA related tools
  - Interested in C++, reverse engineering and compilers
  - @dev_metthal, marek.milkovic@avast.com

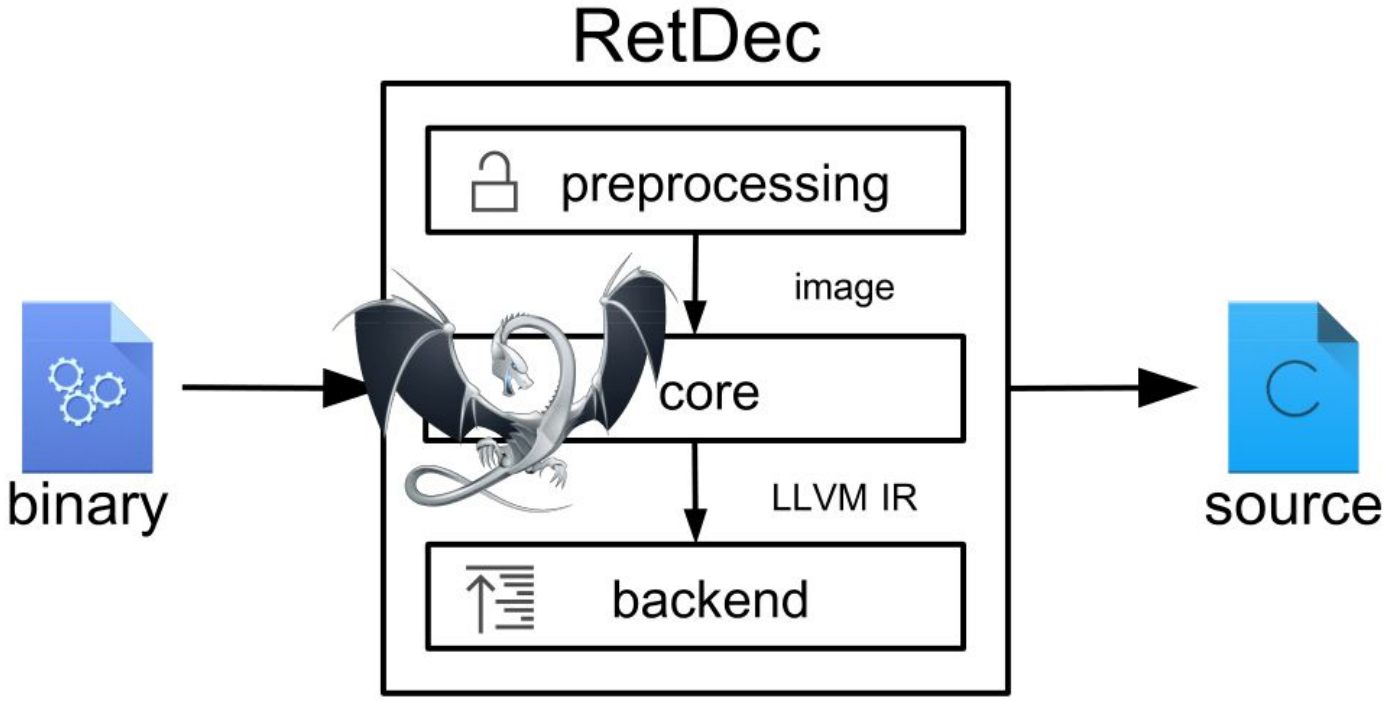
What Is RetDec?

- Set of reversing tools
- Chained together → generic binary code decompiler
- Separate → research, other (internal) projects, ...
- Core based on LLVM
- History
  - 2011-2013 AVG + BUT FIT via TAČR TA01010667 grant
  - 2013-2016 AVG + BUT FIT students via diploma theses
  - 2016-* Avast + BUT FIT students
  - December 2017 Opened-sourced under the MIT license @github
- https://retdec.com/
- https://github.com/avast-tl/retdec
- https://twitter.com/retdec

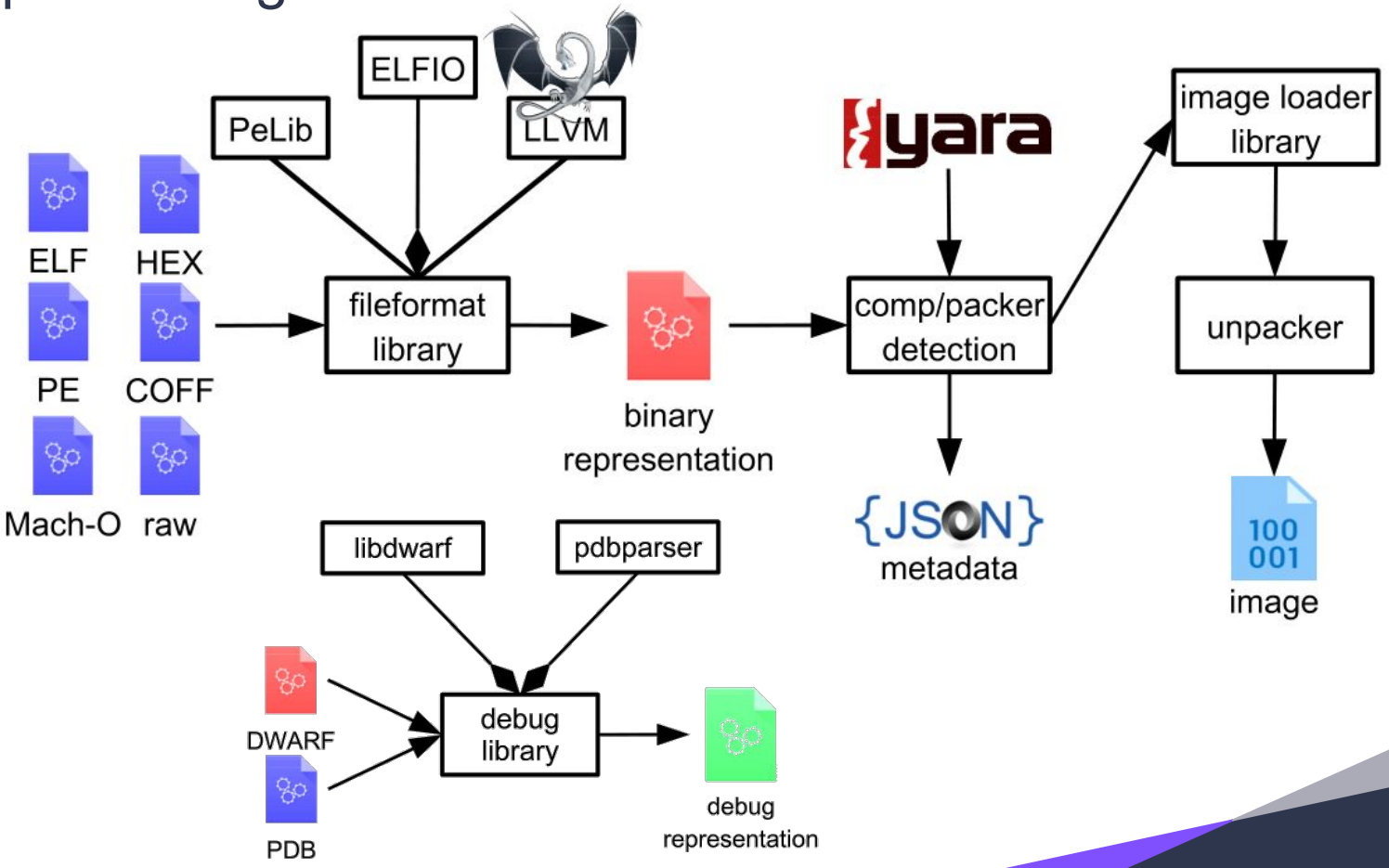
What Is RetDec?

- Supports
  - 32-bit archs: x86, ARM, PowerPC, MIPS
  - OFFs: ELF, PE, COFF, Mach-O, Intel HEX, AR, raw data
  - … working on 64-bit x86, and others
- Does
  - Compiler/packer detection
  - Statically linked code detection
  - OS loader simulation
  - Recursive traversal disassembling
  - High-level code structuring
- Runs on
  - Linux
  - Windows
  - macOS (kinda)

RetDec Structure



Preprocessing



Preprocessing: Unpacker

- Static unpacker
- Signatures + heuristics
- Supports: UPX, MPRESS
- Unpacking of modified variants
- Decompilation of unpacked file
  - Code/Data section separation
- UPX
  - Missing UPX header
  - ADD/XOR/… instruction inserted into unpacking stub (ad-hoc)

Preprocessing: Unpacker

```
  000725e0: 40 64 15 7f d4 01 ff fe be 60 17 11 7f 48 38 1b  @d.......`...H8.
  000725f0: 0f 28 01 00 92 24 61 d0 7f 00 40 25 49 ff 00 00  .(...$a...@%I...
- 00072600: 00 00 55 50 58 21 00 00 00 00 00 00 55 50 58 21  ..UPX!......UPX!
+ 00072600: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
  00072610: 0d 16 08 07 ca 54 49 13 0c 04 33 ad 90 b5 07 00  .....TI...3.....
  00072620: 1c 62 01 00 70 41 1b 00 49 4a 00 df f4 00 00 00  .b..pA..IJ......
```

Our unpacker

```
[UPX] Detected NRV2E unpacking stub based on signature.
[UPX] Started unpacking of file 'file.upx.modified'.
[UPX] Unfiltering filter 0x0 with parameter 0.
[UPX] Unpacking block at file offset 0x1e2.
[UPX] Unfiltering filter 0x49 with parameter 74.
[UPX] Unpacking block at file offset 0x5a6c6.
[UPX] Unfiltering filter 0x0 with parameter 0.
[UPX] Additional packed data detected at the end of the file.
[UPX] Additional data are at file offset 0x5c3bc and have size of 0x16275.
[UPX] Unpacking block from additional data behind segment 2.
[UPX] Unfiltering filter 0x0 with parameter 0.
[UPX] Unpacking last block from additional data at the end of the file.
[UPX] Unfiltering filter 0x0 with parameter 0.
[UPX] Successfully unpacked 'file.upx.modified'!
```

UPX

```
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2017
UPX 3.94        Markus Oberhumer, Laszlo Molnar & John Reiser   May 12th 2017

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
upx: file.upx.modified: NotPackedException: not packed by UPX

Unpacked 0 files.
```

avast

Preprocessing: Stacofin

- YARA based statically linked code detection (F.L.I.R.T.-like technology)
- Lib → full pattern extractor → pattern (YARA) → aggregator → final pattern (YARA)
- Matching using YARA + Capstone

```
function_xyz():
  55 89 E5 83 E4 F0 83 EC
  20 E8 00 00 00 00 C7 44
  24 1C 00 00 00 00 C7 44
  24 18 00 00 00 00 C7 44
  24 14 00 00 00 00 8D 44
  24 14 89 44 24 08 8D 44
  24 18 89 44 24 04 C7 04
  24 44 90 40 00 E8 00 00
  00 00 8B 54 24 14 8B 44
  24 18 89 54 24 04 89 04
  24 E8 00 00 00 00 89 44
  24 1C 8B 54 24 14 8B 44
  24 18 8B 4C 24 1C 89 4C
  24 0C 89 54 24 08 89 44
  24 04 C7 04 24 4A 90 40
  00 E8 00 00 00 00 8B 44
  24 1C C9 C3
```

```
rule rule_0 {
    meta:
        name = "function_xyz"
        size = 132
        refs = "10 ___main 62 _scanf 82 _ack 122 _printf"
        altNames = ""
    strings:
        $1 = { 55 89 E5 83 E4 F0 83 EC 20 E8 ?? ?? ?? ?? C7 44 24 1C 00
               00 00 00 C7 44 24 18 00 00 00 00 C7 44 24 14 00 00 00 00
               8D 44 24 14 89 44 24 08 8D 44 24 18 89 44 24 04 C7 04 24
               44 90 40 00 E8 ?? ?? ?? ?? 8B 54 24 14 8B 44 24 18 89 54
               24 04 89 04 24 E8 ?? ?? ?? ?? 89 44 24 1C 8B 54 24 14 8B
               44 24 18 8B 4C 24 1C 89 4C 24 0C 89 54 24 08 89 44 24 04
               C7 04 24 4A 90 40 00 E8 ?? ?? ?? ?? 8B 44 24 1C C9 C3 }
    condition:
        $1
}
```

avast

Preprocessing: Fileinfo

- Universal binary file parser
  - Headers, sections/segments, symbol tables, ...
- PE, ELF, Mach-O, COFF, Intel HEX
- Plain text or JSON output
- PE
  - Import + export table
  - Certificates
  - Resources
  - .NET data types
  - PDB path
  - …
- Constantly adding new features (RTTI, statically linked code, …)

Preprocessing: Fileinfo

- Compiler/packer detection

```
Bytes on entry point     : 558bec83c4f0b8382c4500e8c42dfbffa1604045008b00e84cd5ffff8b0d44414500a1604045008b008b1520194500e84cd5
Detected tool            : Borland Delphi (6.0 - 7.0) (compiler), 70 from 70 significant nibbles (100%)
Detected tool            : Borland Delphi (6.0) (compiler), 42 from 42 significant nibbles (100%)
Detected tool            : Borland .NET (compiler), 130 from 144 significant nibbles (90.2778%)
Detected tool            : Private exe Protector (2.5x - 2.7x) (packer), 193 from 256 significant nibbles (75.3906%)
Detected tool            : Borland Delphi (5.0) with MCK (compiler), 28 from 38 significant nibbles (73.6842%)
...
```

- Import table and hashes

```
Import table
------------
Number of imports: 7
CRC32            : f9129496
MD5              : f2a8e40d282aacabfb580dcab4ef01dd
SHA256           : c1d9fd376f88fbcebeeeab44163bed2cc80f1058327feb465d6caaad2a3adce7

i     name                                    libName            address    delayed
-------------------------------------------------------------------------------------
0     LoadLibraryA                            KERNEL32.DLL       0x1000f594 No
1     GetProcAddress                          KERNEL32.DLL       0x1000f598 No
2     VirtualProtect                          KERNEL32.DLL       0x1000f59c No
3     VirtualAlloc                            KERNEL32.DLL       0x1000f5a0 No
4     VirtualFree                             KERNEL32.DLL       0x1000f5a4 No
5     ??1CSampleRateConverter2@@QAE@XZ        acdbase.dll        0x1000f5ac No
6     free                                    MSVCR90.dll        0x1000f5b4 No
```

Preprocessing: Fileinfo

- PDB path

```
Related PDB file
----------------
Type                    : RSDS
Path to original PDB file: c:\builds\moz2_slave\tb-rel-c-esr38-w32_bld-0000000\build\objdir-tb\mail\app\thunderbird.pdb
GUID                    : 8c03ab9b-8704-4dfa-98bb-2eae6d2c671f
Version of file (age)   : 1
Timestamp               : 2016-02-11 22:56:05
```

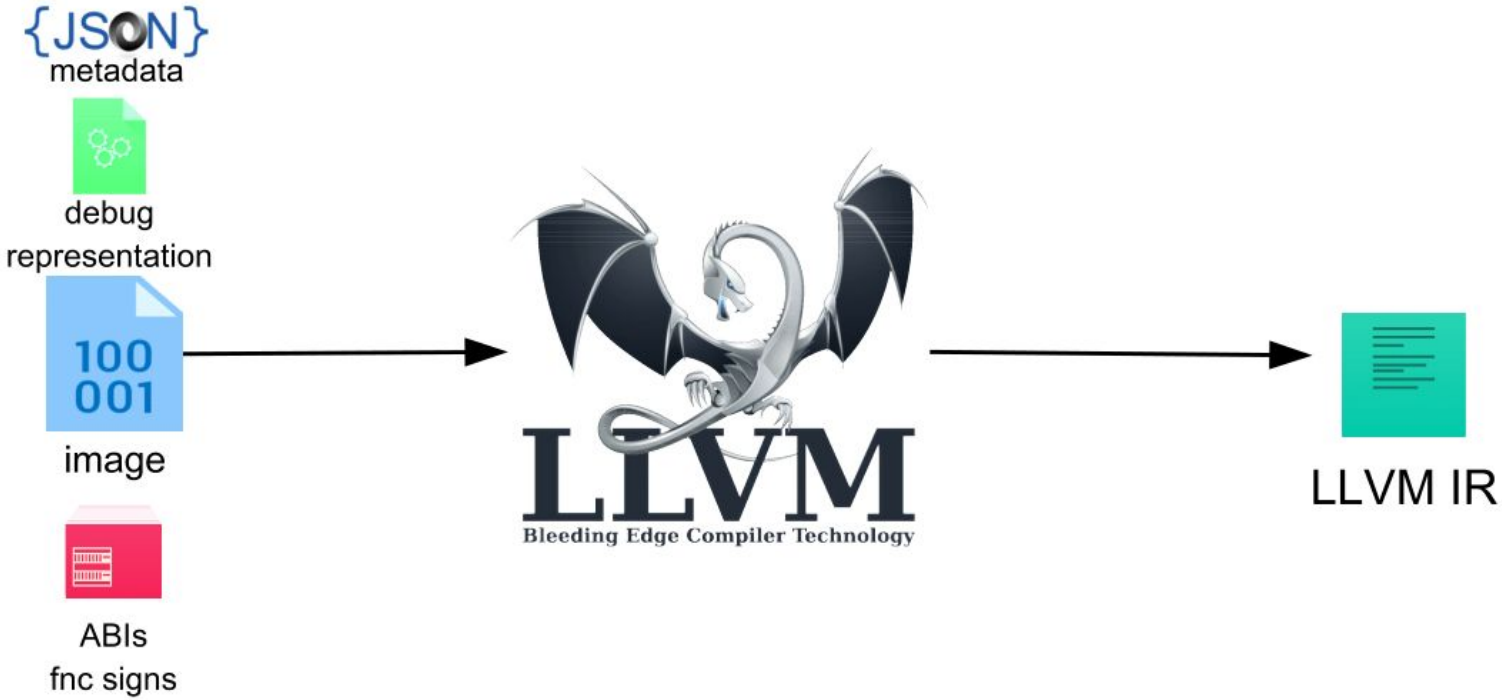
- Certificate (PE authenticode)

```
Certificate #4
Subject name        : Symantec Time Stamping Services CA - G2
Subject organization: Symantec Corporation
Subject             : /C=US/O=Symantec Corporation/CN=Symantec Time Stamping Services CA - G2
Issuer name         : Thawte Timestamping CA
Issuer organization : Thawte
Issuer              : /C=ZA/ST=Western Cape/L=Durbanville/O=Thawte/OU=Thawte Certification/CN=Thawte Timestamping CA
Public key algorithm: rsaEncryption
Signature algorithm : RSA-SHA1
Serial number       : 7E93EBFB7CC64E59EA4B9A77D406FC3B
Valid since         : Dec 21 00:00:00 2012 GMT
Valid until         : Dec 30 23:59:59 2020 GMT
SHA1                : 6C07453FFDDA08B83707C09B82FB3D15F35336B1
SHA256              : 0625FEE1A80D7B897A9712249C2F55FF391D6661DBD8B87F9BE6F252D88CED95
```

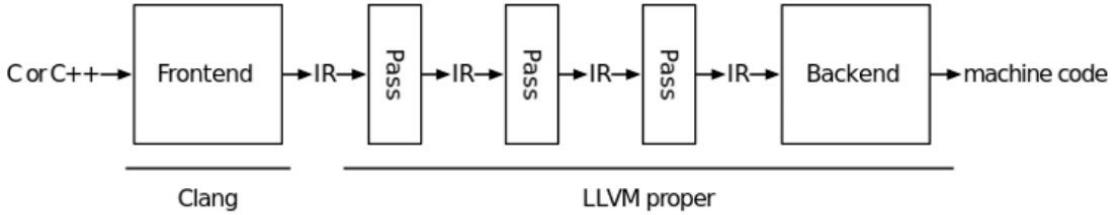
- .NET data types

```
public class regression_test_sample.BasicClass<T> : System.Object, regression_test_sample.BasicInterface<int>
    // Methods
    public .ctor()
    private static .cctor()
    public void PublicMethod()
    protected void ProtectedMethod()
    private void PrivateMethod()
```

Core


{JSON}
metadata
debug
representation
100
001
image
ABIs
fnc signs
LLVM
Bleeding Edge Compiler Technology
LLVM IR

Core: LLVM



- Clang: dozens of analyses & transformation & utility passes
- `clang -o hello hello.c -O3` → 217 passes
  - -targetlibinfo -tti -tbaa -scoped-noalias -assumption-cache-tracker -profile-summary-info -forceattrs -inferattrs -ipsccp -globalopt -domtree -mem2reg -deadargelim -domtree -basicaa -aa -instcombine …

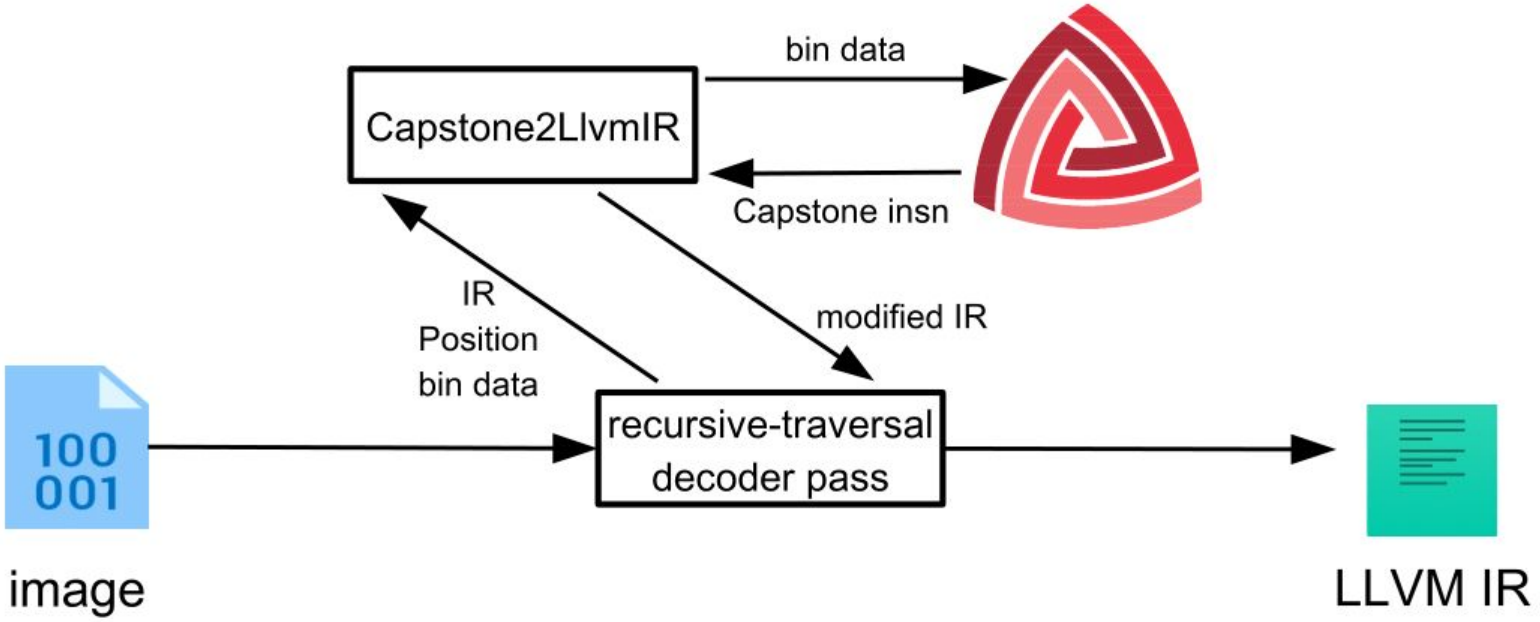
- RetDec: dozens of stock LLVM passes & our own passes
- `retdec-decompiler.sh input.exe`
  - -provider-init -decoder -main-detection -idioms-libgcc -inst-opt -register -cond-branch-opt -syscalls -stack -constants -param-return -local-vars -inst-opt -simple-types -generate-dsm -remove-asm-instrs -class-hierarchy -select-fncs -unreachable-funcs -inst-opt -value-protect <LLVM> -simple-types -stack-ptr-op-remove -inst-opt -idioms -global-to-local -dead-global-assign <LLVM> -phi2seq -value-protect

Core: LLVM IR

- LLVM Intermediate Representation
- Kind of assembly language
- ~62 instructions
- SSA = Static Single Assignment
- Load/Store architecture
- Functions, arguments, returns, data types
- (Un)conditional branches, switches
- Universal IR for efficient compiler transformations and analyses

```
1
2   @global = global i32
3   define i32 @function(i32 %arg)
4   {
5       %x = load i32, i32* @global
6       %y = add i32 %x, %arg
7       store i32 %y, @global
8       return i32 %y
9   }
10
```

Core: Binary to LLVM IR translation



Core: Capstone2LlvmIR

- Capstone insn → sequence of LLVM IR
- Hand-coded sequences for core instructions:
  - ARM + Thumb extension (32-bit)
  - MIPS (32/64-bit)
  - PowerPC (32/64-bit)
  - X86 (32/64-bit)
- Capstone: 64-bit ARM, SPARS, SYSZ, XCore, m68k, m680x, TMS320C64x
- Full semantics only for simple instructions
- More complex instructions translated as pseudo calls
  - `__asm_PMULHUW(mm1, mm2)`
- Implementation details, testing framework (Keystone + LLVM emulator), keeping LLVM IR ↔ ASM mapping, ...

Core: Capstone2LlvmIR

- `./retdec-capstone2llvmir -a mips -b 0x1000 -m 32 -t 'addi $at, $v0, 1000'`

```
1  @pc = internal global i32 0
2  @zero = internal global i32 0
3  @at = internal global i32 0
4  @v0 = internal global i32 0
5  @v1 = internal global i32 0
6  ; ...
7
8  define void @function()
9  {
10   ; 0x1000: addi $at, $v0, 1000
11   store volatile i64 4096, i64* @0
12   %0 = load i32, i32* @v0
13   %1 = add i32 %0, 1000
14   store i32 %1, i32* @at
15   ; ...
16   ret void
17 }
```

Core: Capstone2LlvmIR

- `./retdec-capstone2llvmir -a x86 -b 0x1000 -m 32 -t 'je 1234'`

```
1  @eax = internal global i32 0
2  ; ...
3  @zf = internal global i1 false
4  ; ...
5
6  define void @function()
7  {
8    ; 0x1000: je 0x1234
9    store volatile i64 4096, i64* @0
10   %0 = load i1, i1* @zf
11   call void @__pseudo_cond_branch(i1 %0, i32 4660)
12   ; ...
13   ret void
14 }
15
16 declare void @__pseudo_call(i32)
17 declare void @__pseudo_return(i32)
18 declare void @__pseudo_branch(i32)
19 declare void @__pseudo_cond_branch(i1, i32)
```

Core: Decoding

- Recursive-traversal decoding (disassembling) into LLVM IR
- Works on (analyses) LLVM IR, not assembly
- Priority queue: control flow targets, entry point, debug, symbols, ...

```
1   @eax = internal global i32 0
2   @zf = internal global i1 false
3   ; ...
4
5   define void @function()
6   {
7     ret void
8   }
9
10  declare void @__pseudo_call(i32)
11  declare void @__pseudo_return(i32)
12  declare void @__pseudo_branch(i32)
13  declare void @__pseudo_cond_branch(i1, i32)
```

avast

Core: Decoding

- Recursive-traversal decoding (disassembling) into LLVM IR
- Works on (analyses) LLVM IR, not assembly
- Priority queue: control flow targets, entry point, debug, symbols, ...

```
define void @function()
{
  ; 0x980 : add eax, ebx
  ; ...
  ; 0x1000: je 0x1234
  store volatile i64 4096, i64* @0
  %0 = load i1, i1* @zf
  call void @__pseudo_cond_branch(i1 %0, i32 4660)
  ; ...
  ret void
}
```

```
define void @function()
{
  ; 0x980 : add eax, ebx
  ; ...
  ; 0x1000: je 0x1234
  store volatile i64 4096, i64* @0
  %0 = load i1, i1* @zf
  br i2 %0, label %bb_1234, label %after_1000
after_1000:
  ; ...
bb_1234:
  ; ...
  ret void
}
```

Core: Pattern Matching

- LLVM IR is SSA → `<llvm/IR/PatternMatch.h>`
  - Simple and efficient mechanism for performing general tree-based pattern matches on the LLVM IR
- LLVM IR is load/store → Symbolic Tree Matching
  - Reaching definition analysis → symbolic tree → LLVM-like matcher

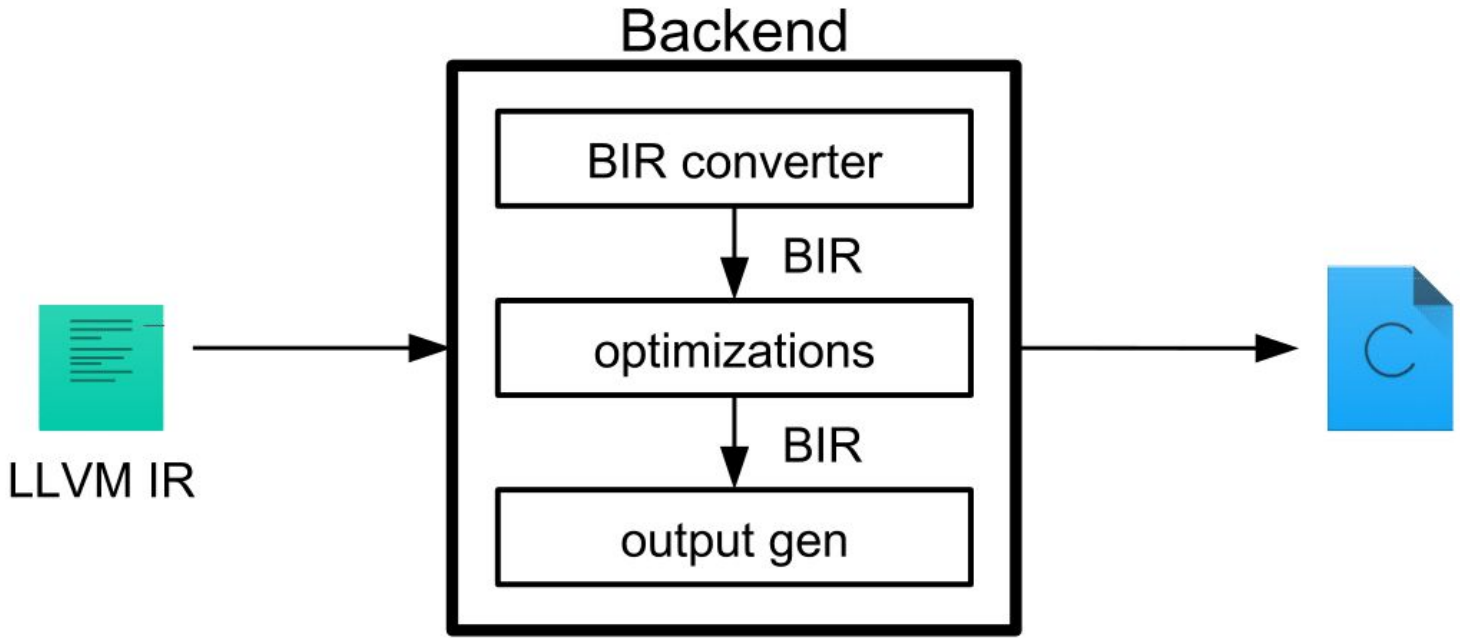
```
//                NE
//                /\
//              ---  ---
//             /        \
//           EQ          1
//           /\
//         ---  ---
//        /        \
//      sub         0
//      /\
//    ---  ---
//   /        \
// <val_1>   <val_2>
```

```
if (match(tree, m_c_ICmp(ICmpInst::ICMP_NE,
        m_c_ICmp(ICmpInst::ICMP_EQ,
                m_Sub(m_Value(val1), m_Value(val2)),
                m_Zero()),
        m_One())))
{
    // matched -> transform to:
    //        NE
    //        /\
    //      ---  ---
    //     /        \
    // <val_1>   <val_2>
    //
}
```

Core: Our Passes

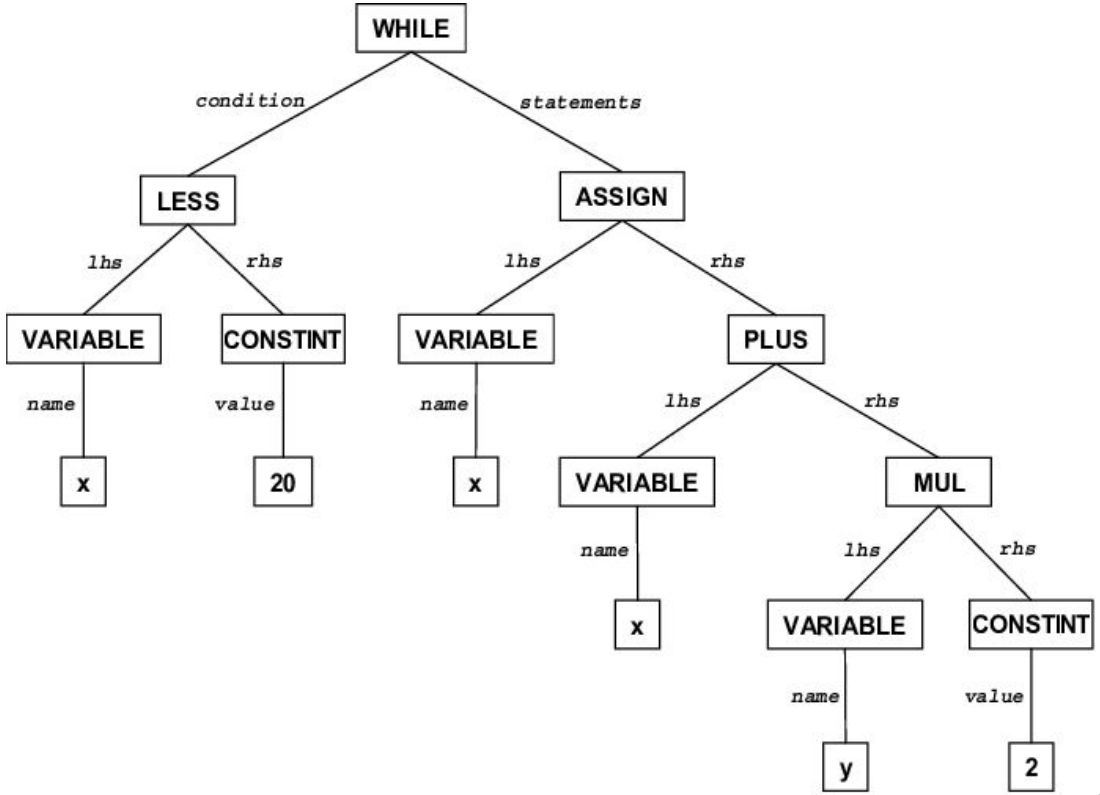
- Idiom detection
- Instruction optimization
- X86 FPU analysis
- Conditional branch transformation
- System calls detection
- Stack reconstruction
- Global variable reconstruction
- Data type propagation
- C++ class hierarchy reconstruction
- Localization (global to local variable transformation)
- ...

Backend

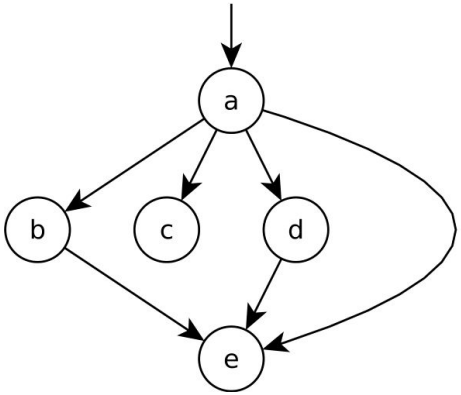


Backend: BIR

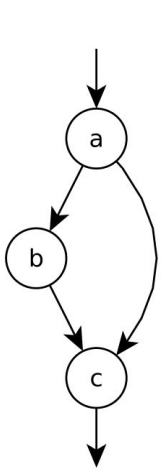
- BIR = Backend IR
- AST = Abstract syntax tree
-
```
while (x < 20)
{
    x = x + (y * 2);
}
```

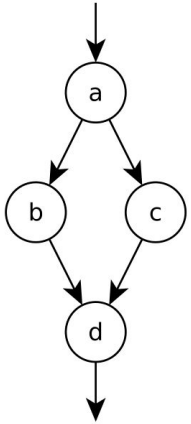


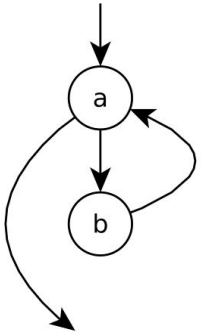
Backend: Code Structuring

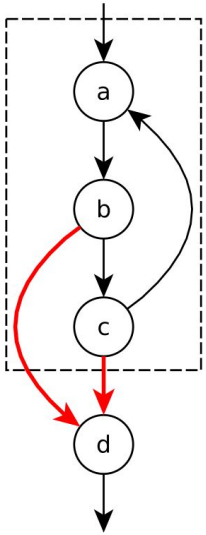


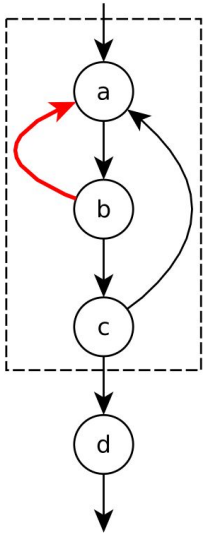
- LLVM IR: only (un)conditional branches & switches
- Identify high-level control-flow patterns
- Restructure BIR: if-else, for-loop, while-loop, switch, break, continue

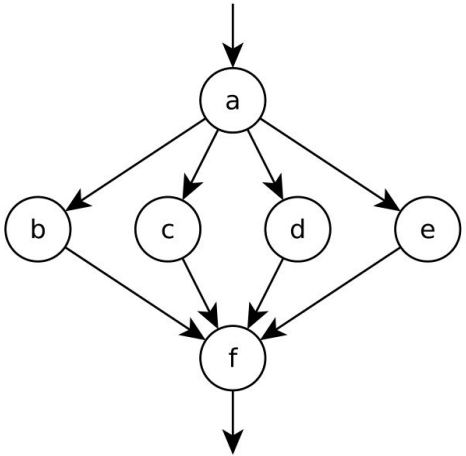












avast

Backend: Optimizations

- Copy propagation
  - Reducing the number of variables
- Arithmetic expression simplification
  - `a + -1 - -4` → `a + 3`
- Negation optimization
  - `if (!(a == b))` → `if (a != b)`
- Pointer arithmetic
  - `*(a + 4)` → `a[4]`
- Control flow conversions
  - `while (true) { … if (cond) break; … }`
  - `if/else chains` → `switch`
- ...

Backend: Code Generation

- Variable name assignment
  - Induction variables: `for (i = 0; i < 10; ++i)`
  - Function arguments: `a1, a2, a3, …`
  - General context names: `return result;`
  - Stdlib context names: `int len = strlen();`
- Stdlib context literals
  - `flock(sock_id, 7)` → `flock(sock_id, LOCK_SH | LOCK_EX | LOCK_NB)`
- Output generation
  - C
  - CFG = Control-Flow Graph
  - Call Graph

RetDec IDA Plugin

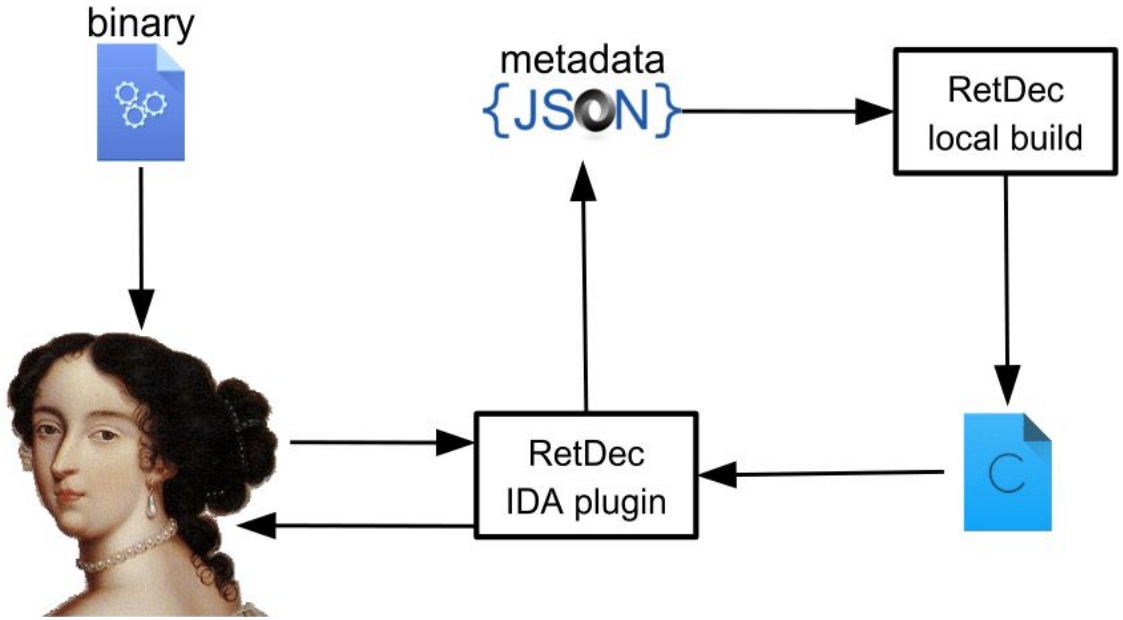
```
.text:004015BB    push    ebp
.text:004015BC    mov     ebp, esp
.text:004015BE    and     esp, 0FFFFFFF0h
.text:004015C1    sub     esp, 20h
.text:004015C4    call    ___main
.text:004015C9    mov     [esp+20h+var_4], 0
.text:004015D1    mov     [esp+20h+var_8], 0
.text:004015D9    mov     [esp+20h+var_C], 0
.text:004015E1    lea     eax, [esp+20h+var_C]
.text:004015E5    mov     [esp+20h+var_18], eax
.text:004015E9    lea     eax, [esp+20h+var_8]
.text:004015ED    mov     [esp+20h+var_1C], eax
.text:004015F1    mov     [esp+20h+Format], offset Format
.text:004015F8    call    _scanf
.text:004015FD    mov     edx, [esp+20h+var_C]
.text:00401601    mov     eax, [esp+20h+var_8]
.text:00401605    mov     [esp+20h+var_1C], edx
.text:00401609    mov     [esp+20h+Format], eax
.text:0040160C    call    _ack
.text:00401611    mov     [esp+20h+var_4], eax
.text:00401615    mov     edx, [esp+20h+var_C]
.text:00401619    mov     eax, [esp+20h+var_8]
.text:0040161D    mov     ecx, [esp+20h+var_4]
.text:00401621    mov     [esp+20h+var_14], ecx
.text:00401625    mov     [esp+20h+var_18], edx
.text:00401629    mov     [esp+20h+var_1C], eax
.text:0040162D    mov     [esp+20h+Format], offset aAckermanDDD
.text:00401634    call    _printf
.text:00401639    mov     eax, [esp+20h+var_4]
.text:0040163D    leave
.text:0040163E    retn
```

```
//
// This file was generated by the Retargetable Decompiler
// Website: https://retdec.com
// Copyright (c) 2017 Retargetable Decompiler <info@retdec.com>
//

#include <stdint.h>
#include <stdio.h>

// ------------------------ Functions ------------------------

int32_t _ack(int32_t a1, int32_t a2) {
    if (a1 == 0) {
        return a2 + 1;
    }
    int32_t result;
    if (a2 == 0) {
        result = _ack(a1 - 1, 1);
    } else {
        result = _ack(a1 - 1, _ack(a1, a2 - 1));
    }
    return result;
}
int main(int argc, char ** argv) {
    ___main();
    int32_t v1 = 0;
    int32_t v2 = 0;
    scanf("%d %d", &v1, &v2);
    int32_t result = _ack(v1, v2);
    printf("ackerman( %d , %d ) = %d\n", v1, v2, result);
    return result;
}
```

avast

RetDec IDA Plugin

- Look & feel native
- Same object names as IDA
- Interactive
  - We have to fake it
  - Local decompilation
- Built with IDA SDK 7.0
- Works in IDA 7.x
- Does not work in freeware IDA 7.0



avast

RetDec IDA Plugin

```
// From module:   /home/peter/decompiler/decompiler
// Address range: 0x804851c - 0x8048576
// Line range:    4 - 11
int32_t ack(int32_t m, int32_t n) {
    // 0x8
    if (m
        //
        //
        //
        re
    }
    // 0x8
    int32_t result; // 0x8048576_11
    if (n == 0) {
        // 0x8048536
        result = ack(m - 1, 1);
        // branch -> 0x8048575
    } else {
        // 0x804854e
        result = ack(m - 1, ack(m, n - 1));
        // branch -> 0x8048575
    }
    // 0x8048575
    return result;
```

Jump to ASM	A
Rename function	N
Change type declaration	Y
Open xrefs window	X
Open calls window	C
Edit func comment	/
Move backward	Esc
Move forward	Ctrl+Enter

```
// ---------------------- Global Variables ------------

int32_t __CTOR_LIST__ = -1; // 0x80497f4

// ------------------------ Functions -----------------

// Address range: 0x8048680 - 0x80486a9
int32_t __do_global_ctors_aux(void) {
    // 0x8048680
    if (__CTOR_LIST__ == -1) {
        // 0x80486a4
        return -1;
    }
    int32_t v1 = 0x8
    unknown_ffffffff
    // branch -> 0x8048698
    while (*(int32_t *)(v1 - 4) != -1) {
        // 0x8048698
        v1 -= 4;
        unknown_ffffffff();
        // continue -> 0x8048698
    }
    // 0x80486a4
    return -1;
```

Jump to ASM	A
Rename global variable	N
Edit func comment	/
Move backward	Esc
Move forward	Ctrl+Enter

RetDec IDA Plugin

DSM

```
                dd offset loc_4096B8; jump table
                ; indirect table for switch
byte_4096FC     db      0,      5,      1,      1
                db      5,      2,      5,      5
                db      3,      5,      5,      5
                db      5,      5,      5,      5
                db      5,      5,      5,      5
                db      5,      5,      5,      5
                db      5,      5,      5,      5
                db      5,      5,      5,      5
                db      5,      5,      5,      5
                db      5,      5,      5,      5
                db      5,      5,      5,      5
                db      5,      5,      5,      5
                db      5,      5,      5,      5
                db      5,      5,      5,      5
                db      5,      5,      5,      5
                db      5,      5,      5,      5
                db      5,      5,      5,      5
                db      5,      5,      5,      5
                db      5,      5,      5,      5
                db      5,      5,      5,      5
                db      5,      5,      5,      5
                db      5,      5,      5,      4
```

Hex-Rays

```
v4 = 1;
switch ( GetLastError() )
{
  case 0u:
    v4 = 0;
    break;
  case 2u:
  case 3u:
    v4 = 6;
    break;
  case 5u:
    v4 = 4;
    break;
  case 8u:
    v4 = 3;
    break;
  case 0x57u:
    v4 = 2;
    break;
  default:
    break;
}
```

RetDec

```
int32_t result = 1; // esi
switch (GetLastError()) {
    case 0: {
        result = 0;
        break;
    }
    case 2: {
        result = 6;
        break;
    }
    case 3: {
        result = 6;
        break:
    }
    case 5: {
        result = 4;
        break;
    }
    case 8: {
        result = 3;
        break;
    }
    case 87: {
        result = 2;
        break;
    }
}
```

What’s next?

- Output quality improvements
  - Major refactoring in RetDec v3.1
  - Still a lot of work is needed
- Better documentation
- New architectures (64-bit)
  - x64
  - ARM
  - …
- Better integration with IDA
- Better integration with other tools:
  - Binary Ninja
  - Radare2
  - x64dbg

Questions?

https://retdec.com
https://github.com/avast-tl
https://twitter.com/retdec