Retargetable
Decompiler

Who Are We?

- Peter Matula
  - Senior software developer @Avast
  - Main developer of the RetDec decompiler
  - Love Rock climbing, Ski mountaineering & Beer
  - peter.matula@avast.com
- Marek Milkovič
  - Senior software developer @Avast
  - Works on RetDec preprocessing stage and YARA-related tools
  - Interested in C++, reverse engineering and compilers
  - @dev_metthal, marek.milkovic@avast.com
- Peter Kubov
  - Student, intern @Avast
  - Actually did most of the work
  - peter.kubov@avast.com

Goals

- RetDec↔Radare2 integration:
  - Become a viable decompilation option for r2 users
  - Use r2/Cutter as user interface
  - Make RetDec better while doing so
- This talk:
  - Discussion: developers ↔ users
  - Discussion: developers ↔ developers

Agenda

- RetDec introduction
- RetDec↔IDA
  - Plugin for the other reversing tool
- RetDec↔Radare2
  - The starting point
- RetDec improvements
  - How to make it better?
- Demo
  - Yay, it is doing something!
- Future
  - It could be (much) better

RetDec

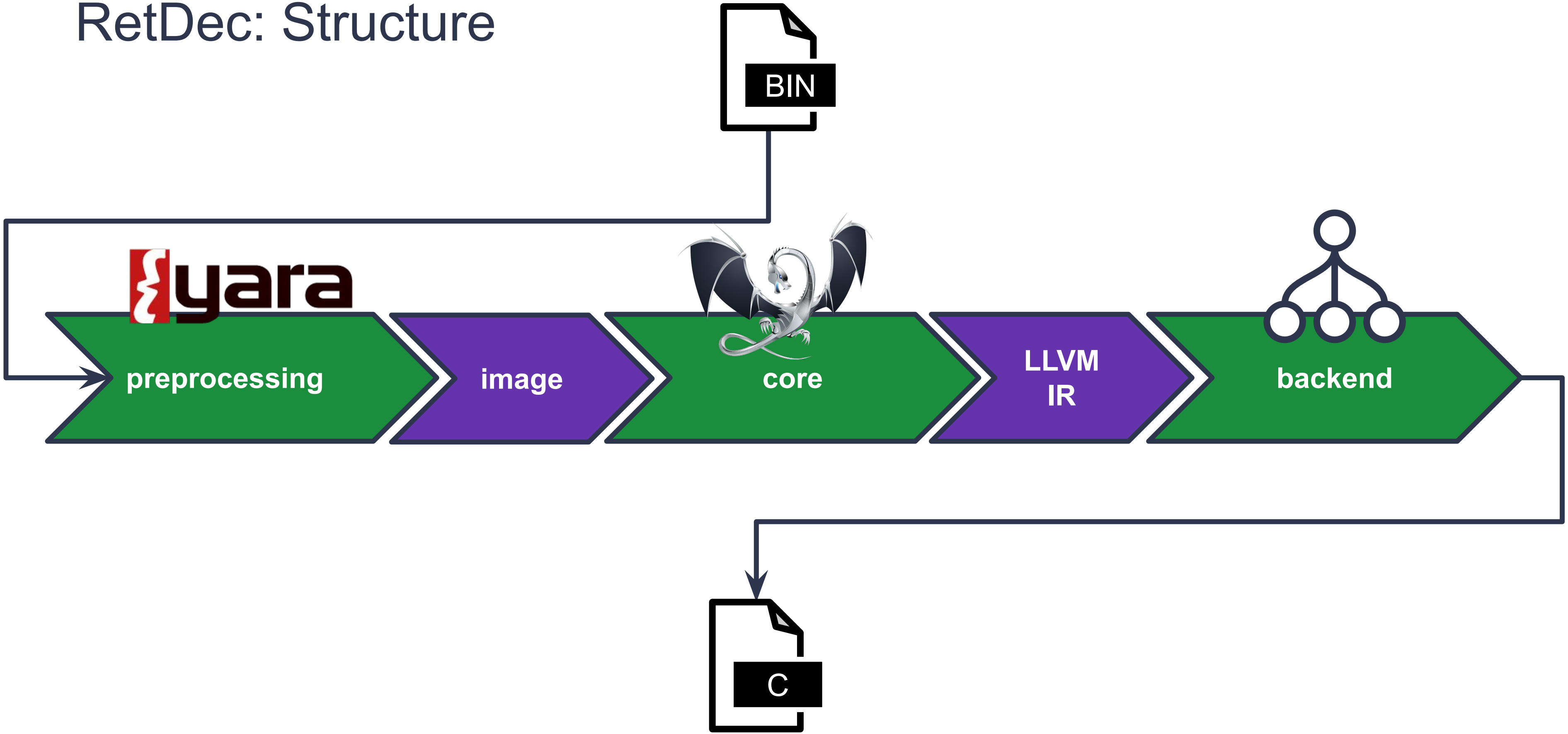
- Set of reversing tools
- Chained together → generic binary code decompiler
- Separate → research, other (internal) projects, …
- Core based on LLVM
- Supports: x86, x64, ARM, ARM64, MIPS, PowerPC

https://retdec.com

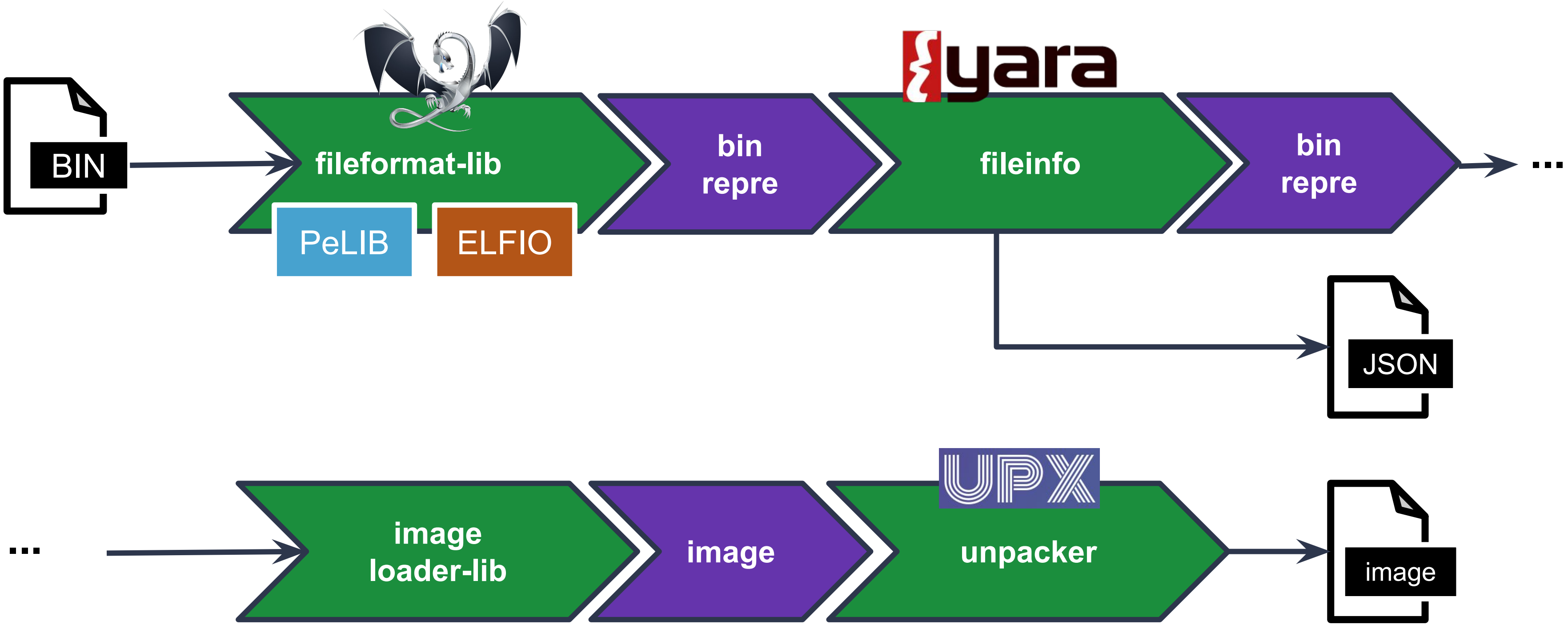
https://github.com/avast/retdec

https://twitter.com/retdec

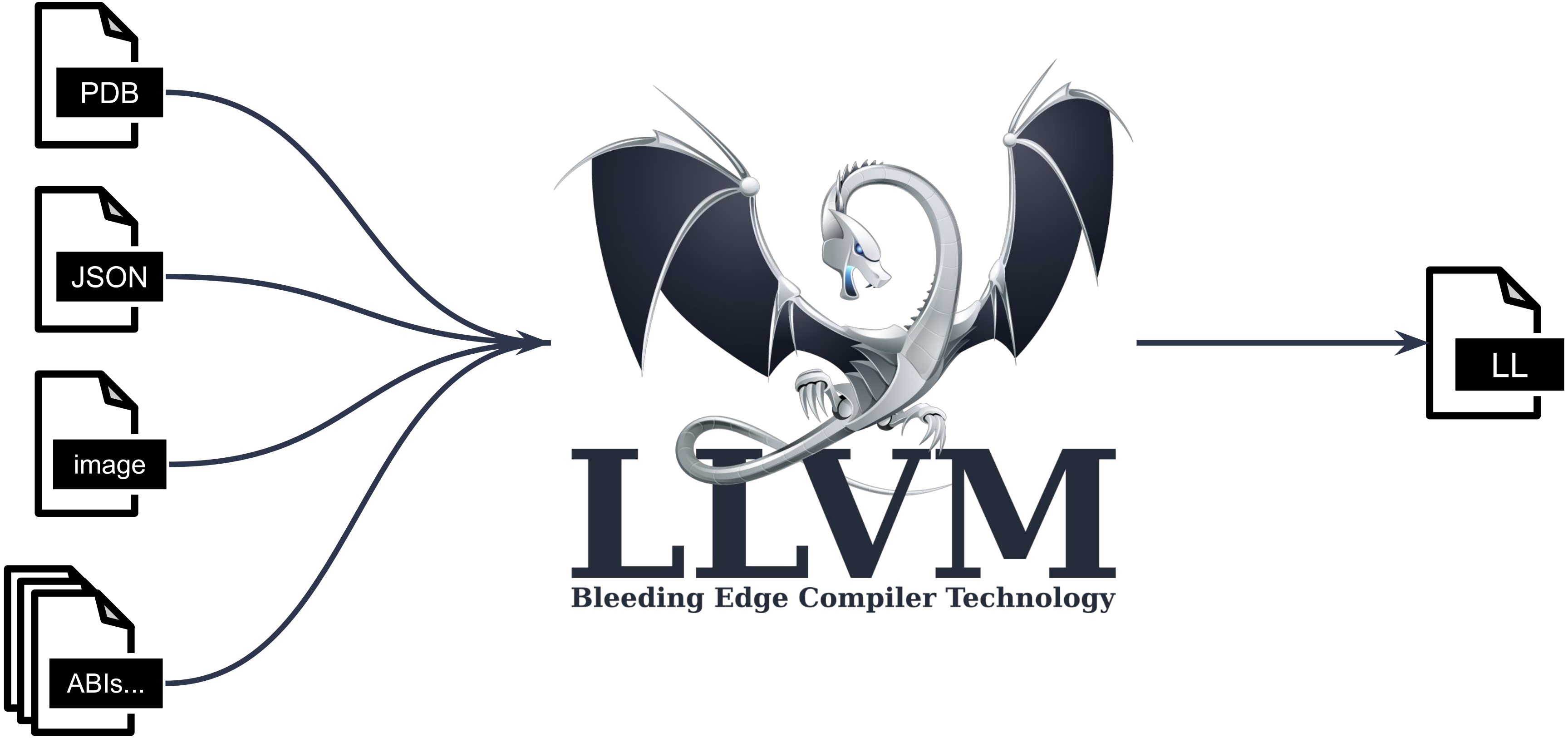
RetDec: Structure



RetDec: Structure

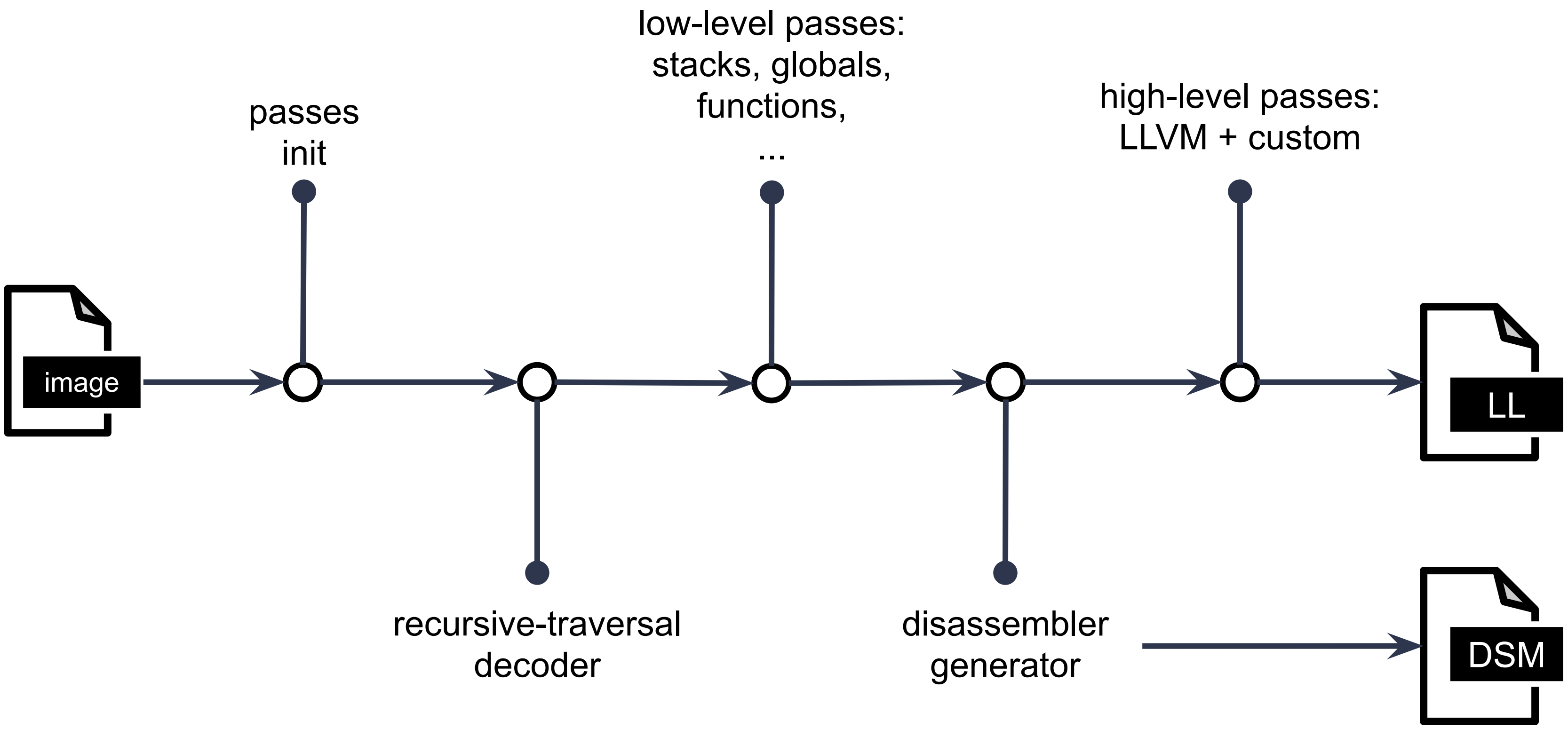


RetDec: Core



avast

RetDec: Core

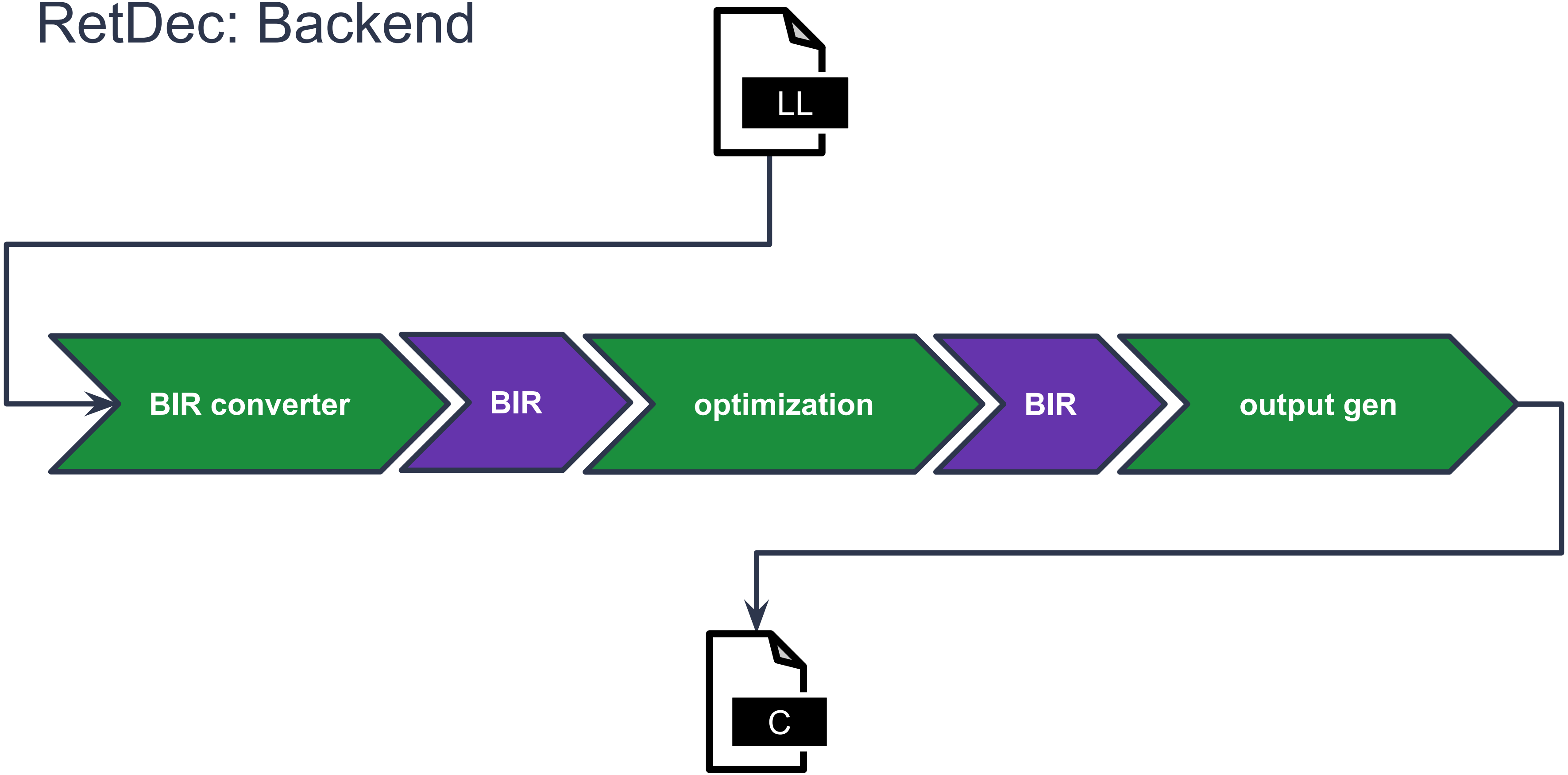


avast

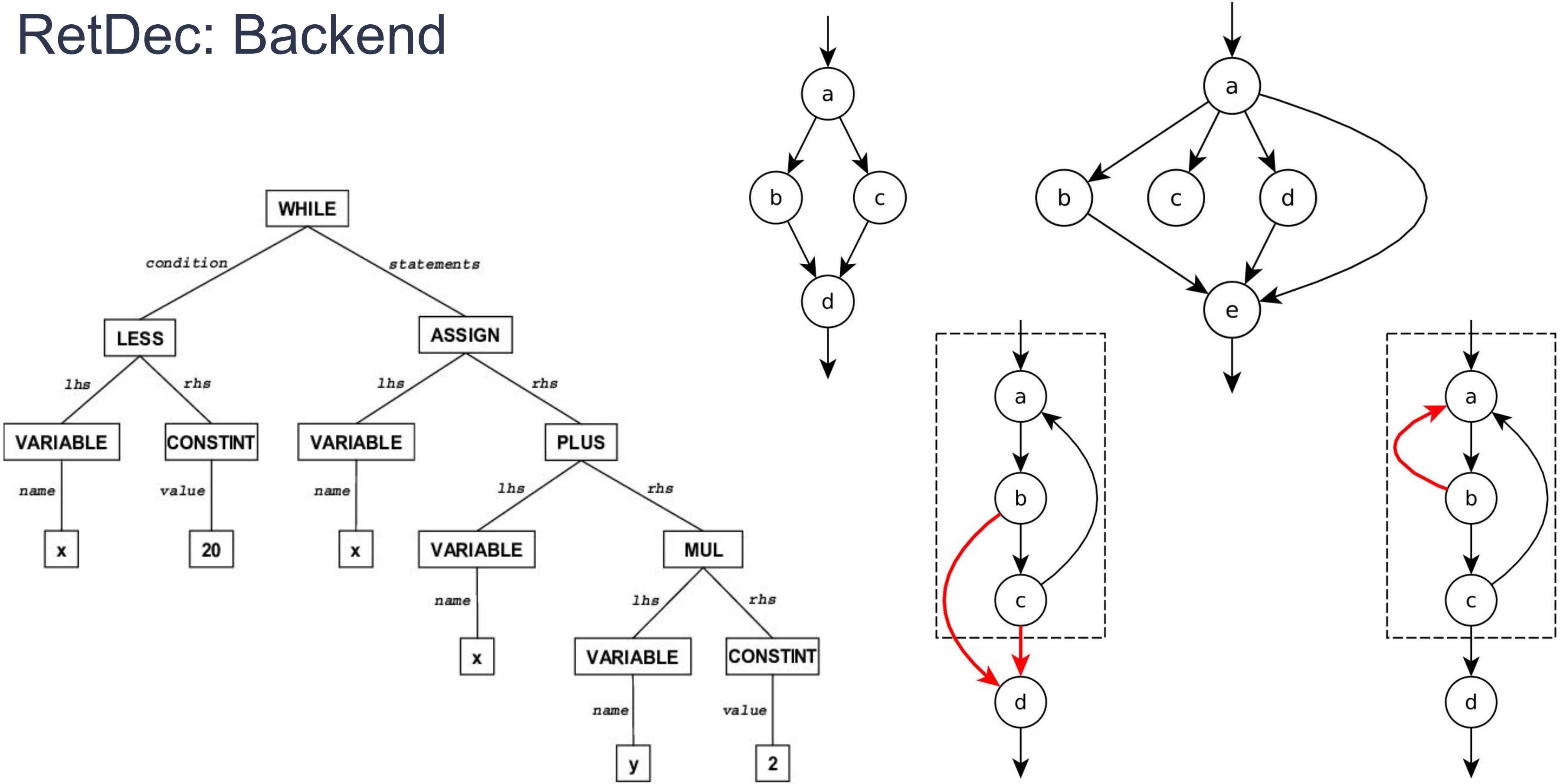
RetDec: Core LLVM Intermediate Representation

```
; Universal IR for efficient compiler transformations and analyses
@global = global float                ; load/store for allocated objects
define i32 @function(i1 %arg) {
    br i1 %arg, label %load_lab, label %return_lab
load_lab:                             ; (un)conditional branches & switches
    %x = load float, float* @global   ; SSA for temporaries
    %conv = fptoui float %x to i32    ; strongly typed
    return i32 %conv
return_lab:                           ; no ifs, fors, whiles, etc.
    return i32 1234
}
```

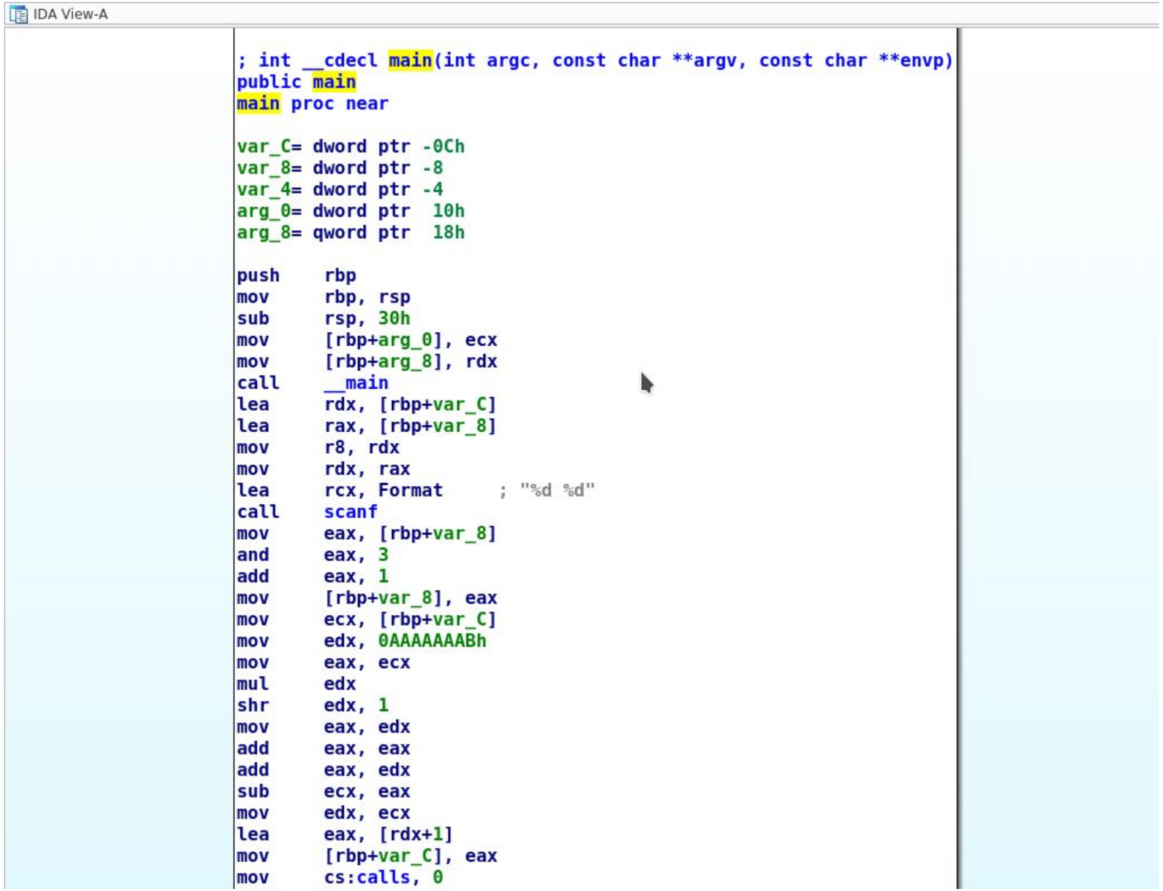
RetDec: Backend



RetDec: Backend

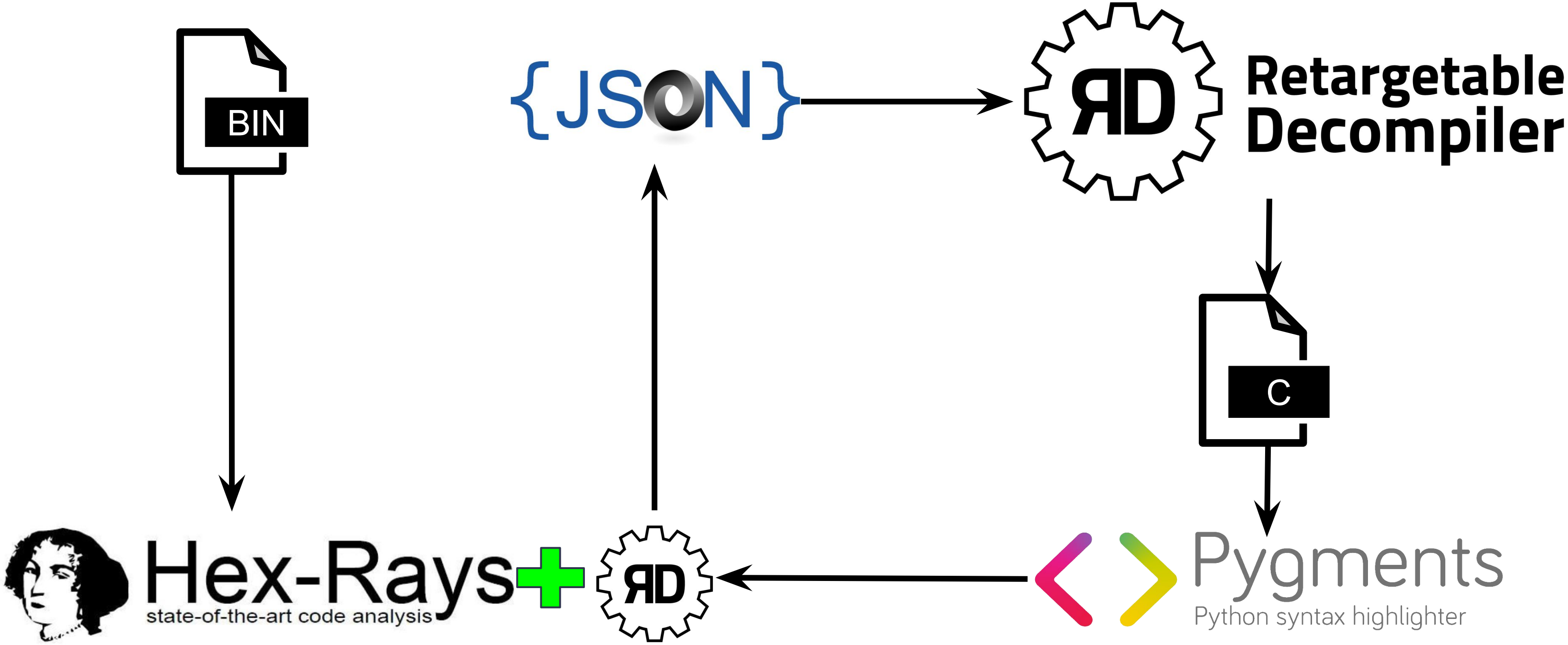


RetDec: IDA Plugin



```
; int __cdecl main(int argc, const char **argv, const char **envp)
public main
main proc near

var_C= dword ptr -0Ch
var_8= dword ptr -8
var_4= dword ptr -4
arg_0= dword ptr  10h
arg_8= qword ptr  18h

push    rbp
mov     rbp, rsp
sub     rsp, 30h
mov     [rbp+arg_0], ecx
mov     [rbp+arg_8], rdx
call    __main
lea     rdx, [rbp+var_C]
lea     rax, [rbp+var_8]
mov     r8, rdx
mov     rdx, rax
lea     rcx, Format     ; "%d %d"
call    scanf
mov     eax, [rbp+var_8]
and     eax, 3
add     eax, 1
mov     [rbp+var_8], eax
mov     ecx, [rbp+var_C]
mov     edx, 0AAAAAAABh
mov     eax, ecx
mul     edx
shr     edx, 1
mov     eax, edx
add     eax, eax
add     eax, edx
sub     ecx, eax
mov     edx, ecx
lea     eax, [rdx+1]
mov     [rbp+var_C], eax
mov     cs:calls, 0
```

RetDec: IDA Plugin

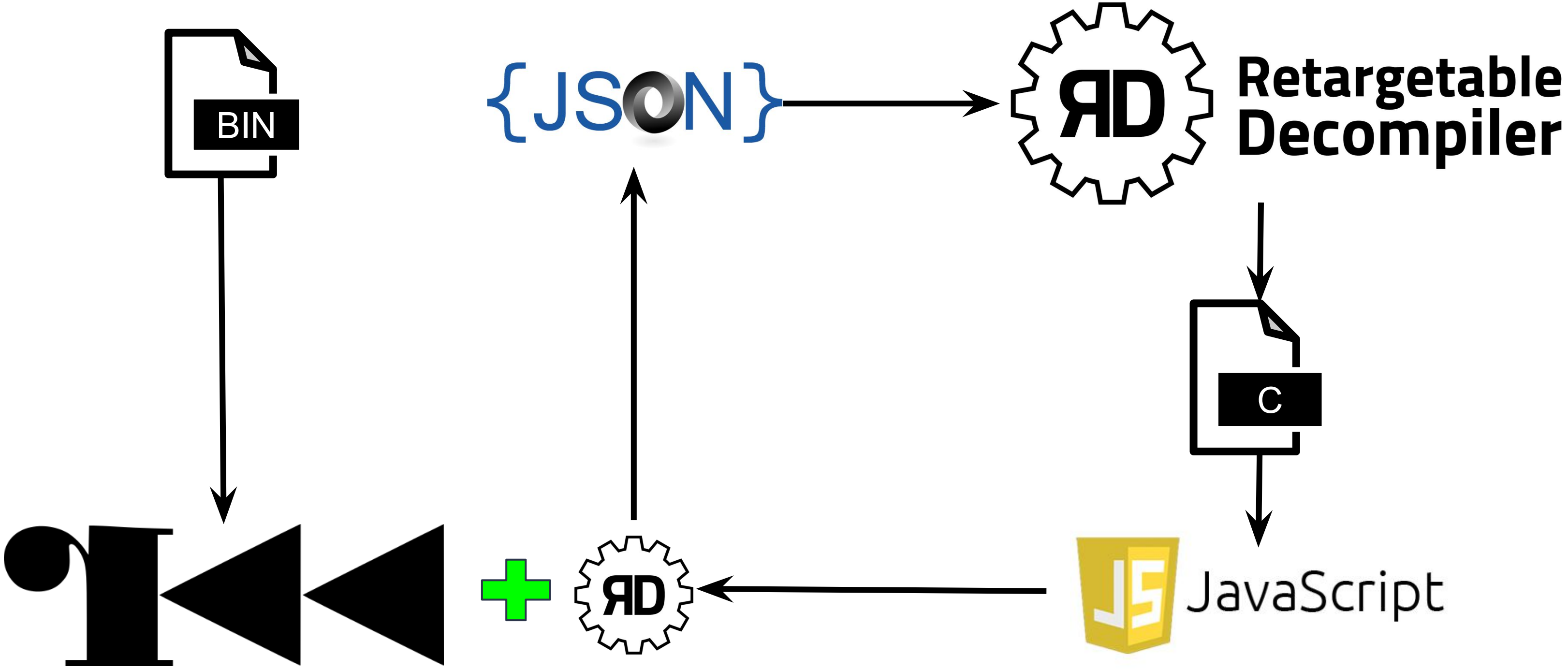


RetDec: Radare2 Plugin

https://github.com/securisec/r2retdec

```
ubuntu@ubuntu-xenial:~$ 
```

RetDec: Radare2 Plugin



Improvements: Setup

- Plugin:

  https://github.com/xkubov/r2retdec

  - Will be moved …
  - Issues, wiki, etc. at the new location

- RetDec:

  https://github.com/avast/retdec/tree/r2con

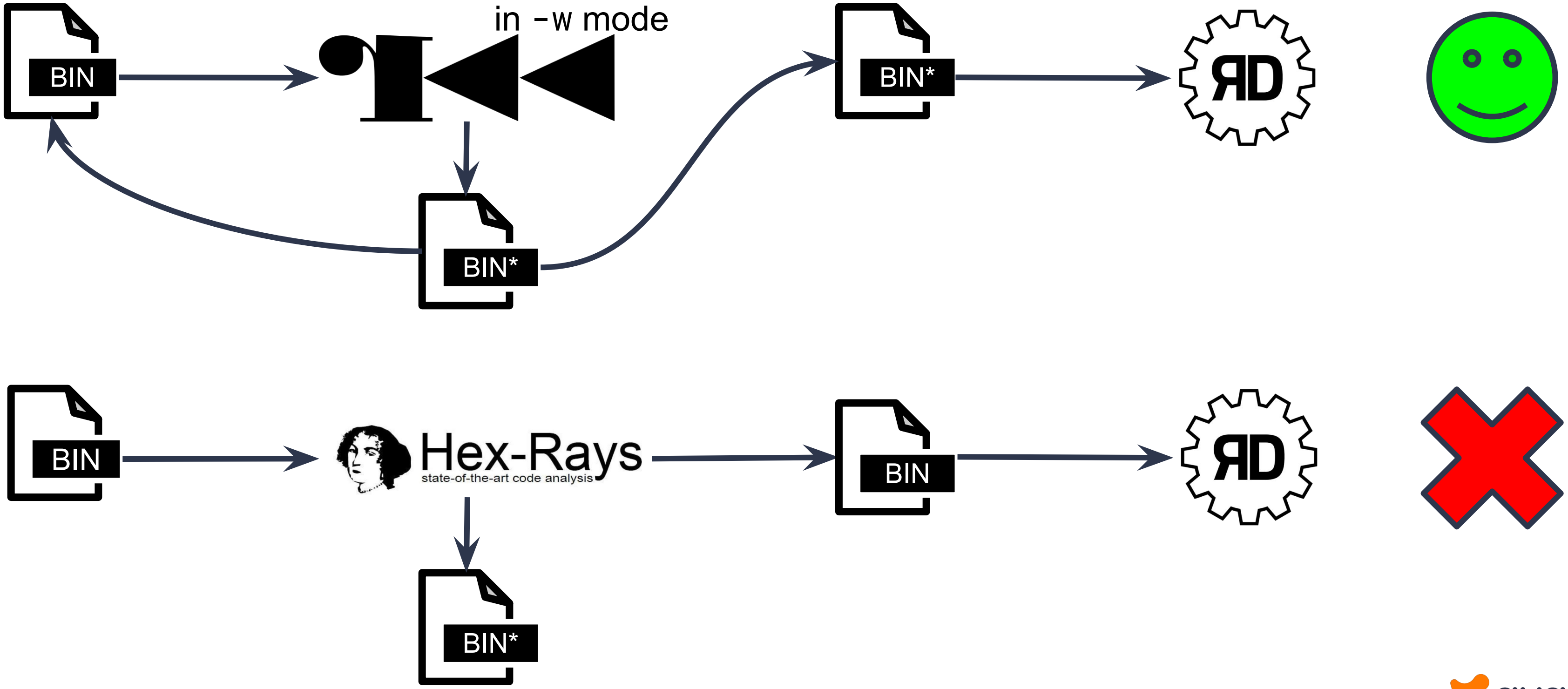
  https://github.com/avast/retdec-regression-tests/tree/r2con

  https://github.com/avast/retdec-regression-tests-framework/tree/r2con

  - `r2con` branches
  - Will be merged to `master`

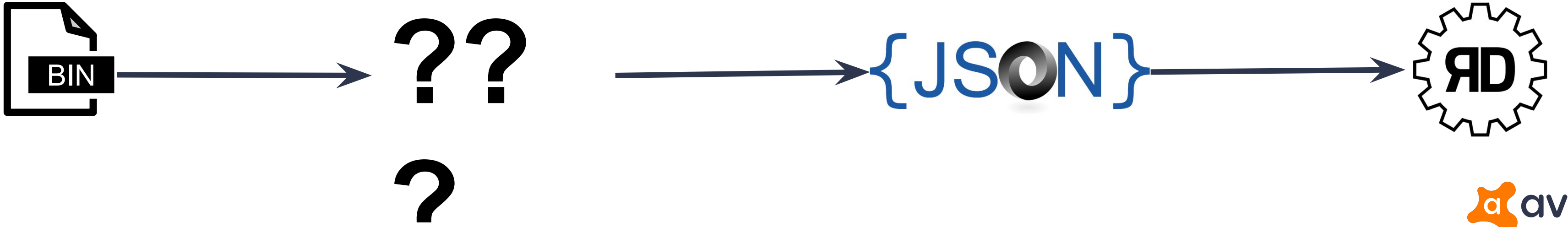
Improvements: Input Files

- Binary modification → problem?



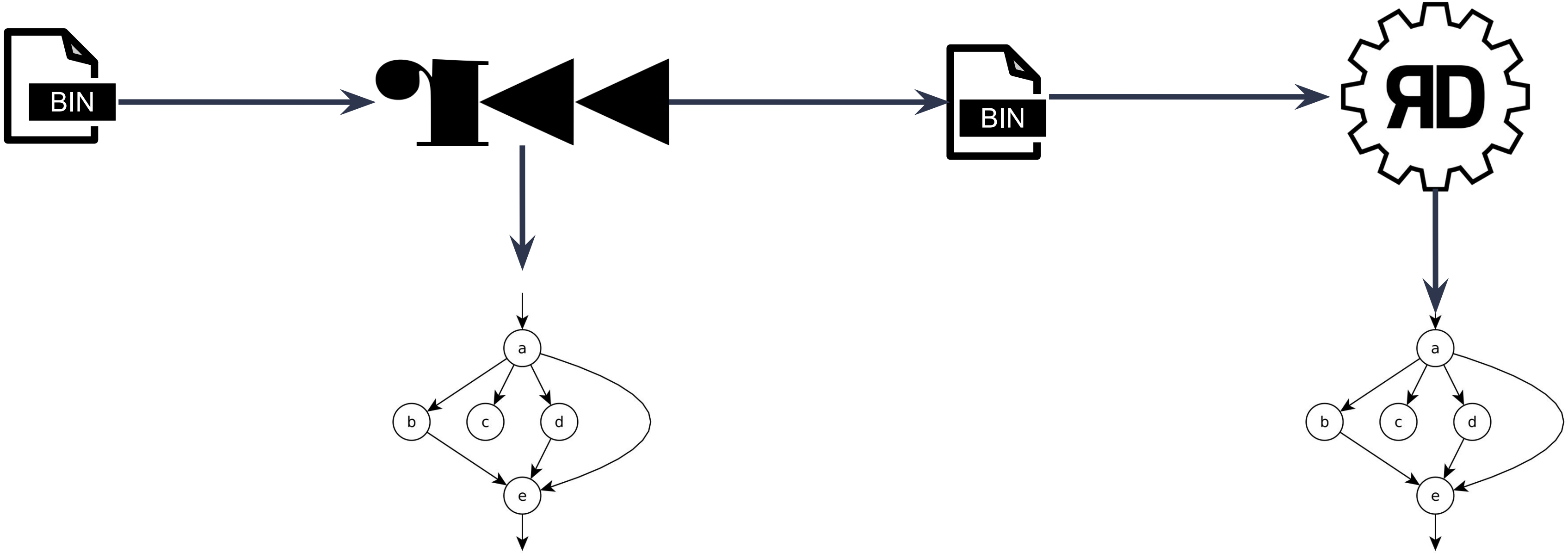
Improvements: Input Files

- Discussion:
  - Extract the current binary data and pass them to RetDec in configuration JSON (e.g. base64 encoded)?
  - Pros?
    - Single, self-contained, OFF-agnostic input
  - Cons?
    - Another way how pass input… conceptual?
    - Entire binary?… big
    - Only needed parts?... what is needed?



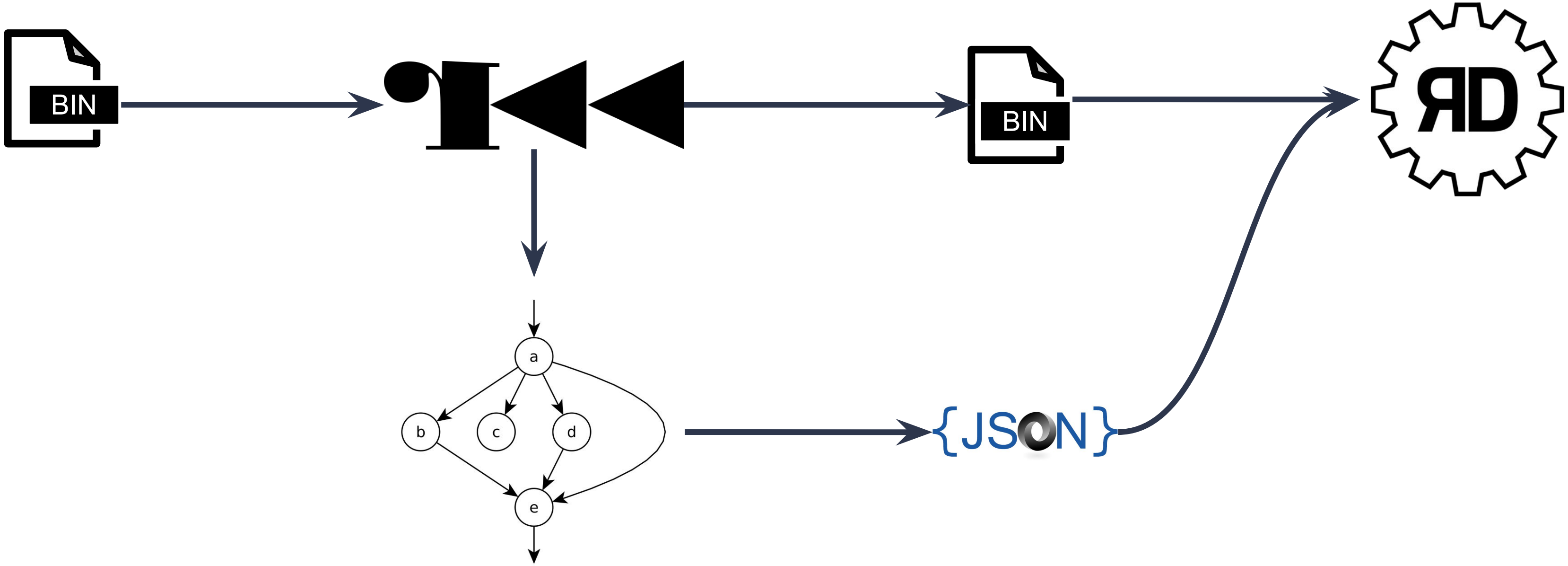
Improvements: Disassembly & Control Flow

- Problem: duplicate recursive-traversal disassembling in r2 and RetDec
  - non-trivial, expensive, potentially different



Improvements: Disassembly & Control Flow

- Solution: extract CF in r2, pass it in JSON, use it in RetDec disassembler



Improvements: Useful Info from Radare2

- Problem: we want matching info in r2 and RetDec
- Solution: extract all possible info from r2 and use it in RetDec
  - Function names
  - Demangled names
  - Function calling conventions
  - Function arguments: names, types
  - Local variables: names, offsets, types
  - Global variables: flag names
    - types, etc.: https://github.com/radare/radare2/issues/8565
  - Types = Including complex structure types

Improvements: Complex Types

- Problem: RetDec suck(ed) at complex data types usage

```
// Original C
struct XYZ { int x;   int y;   int z; }
struct XYZ xyz;

int fnc() {
    xyz.x = 1;
    xyz.y = 2;
    xyz.z = 3;
}
```

Improvements: Complex Types

- Problem: RetDec suck(ed) at complex data types usage

```
// Decompiled C - without any additional information
int glob_0x1000;
int glob_0x1004;
int glob_0x1008;

int fnc() {
    glob_0x1000 = 1;
    glob_0x1004 = 2;
    glob_0x1008 = 3;
}
```

Improvements: Complex Types

- Problem: RetDec **suck(ed)** at complex data types usage

```
// Decompiled C - with additional information about data type @ 0x1000
struct XYZ { int e1;  int e2;  int e3; }
struct XYZ xyz;  // 0x1000
int glob_0x1004;
int glob_0x1008;
int fnc() {
    xyz.e1 = 1;
    glob_0x1004 = 2;
    glob_0x1008 = 3;
}
```

Improvements: Complex Types

- Solution: new aggregation algorithm for global objects (addresses)...

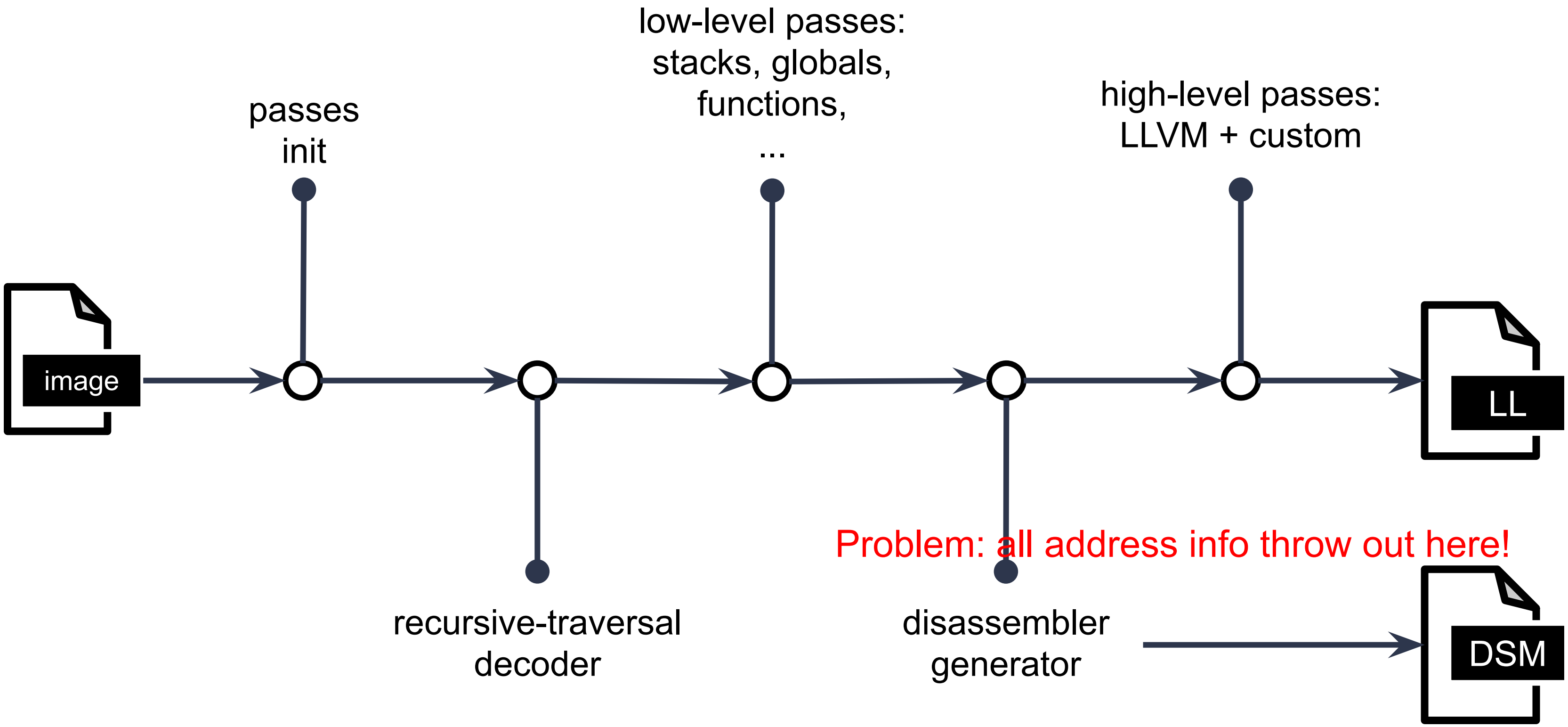
```
// Decompiled C - with additional information about data type @ 0x1000
struct XYZ { int e1;  int e2;  int e3; }
struct XYZ xyz;  // 0x1000

int fnc() {
    xyz.e1 = 1;
    xyz.e2 = 2;
    xyz.e3 = 3;
}
```

Improvements: Complex Types

- … and also for local objects (offsets)

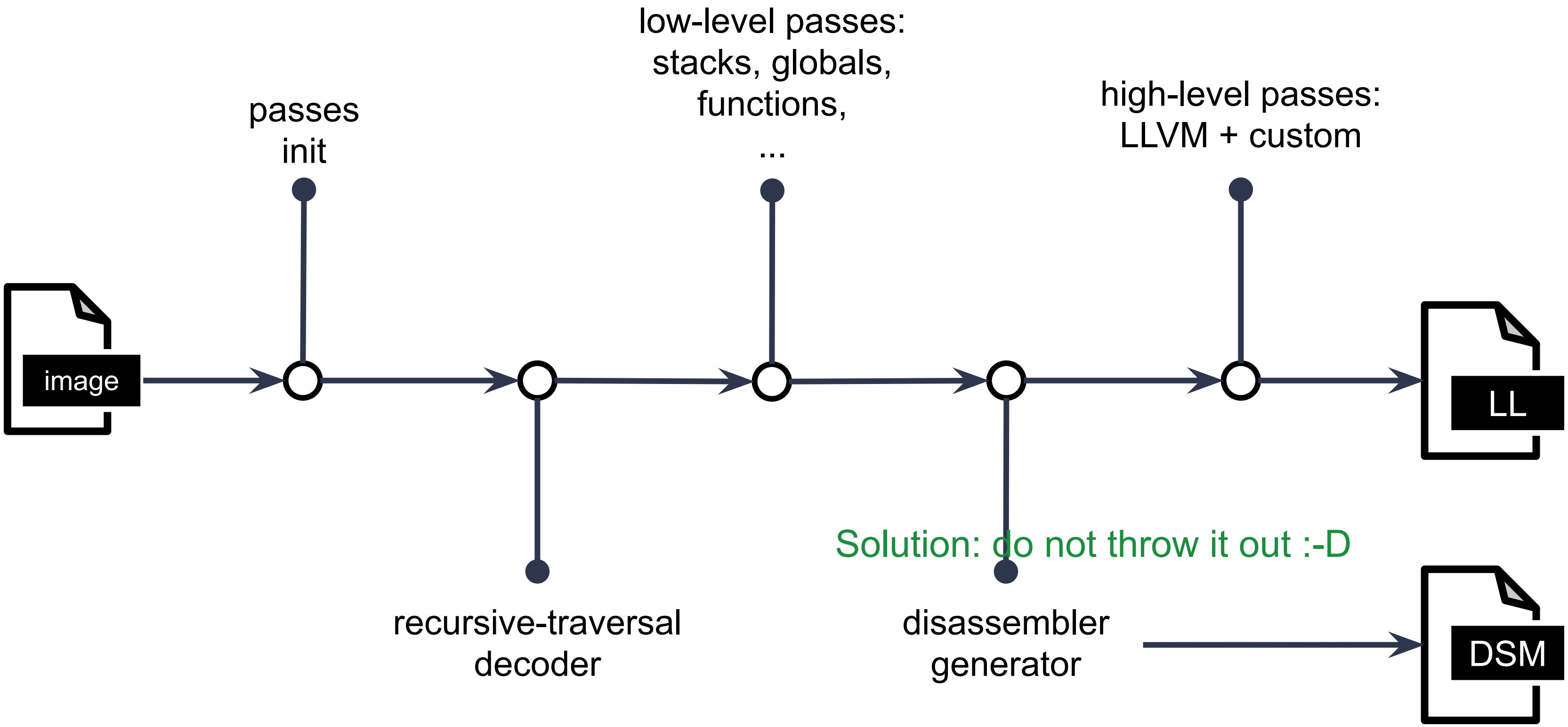
```
// Decompiled C - with additional information about data type for xyz
struct XYZ { int e1;  int e2;  int e3; }

int fnc(struct XYZ xyz) {
     xyz.e1 = 1;
     xyz.e2 = 2;
     xyz.e3 = 3;
}
```

Improvements: ASM Addresses for Statements



Problem: all address info throw out here!

Improvements: ASM Addresses for Statements



passes init
low-level passes: stacks, globals, functions, ...
high-level passes: LLVM + custom
image
LL
recursive-traversal decoder
disassembler generator
DSM

Solution: do not throw it out :-D

Improvements: ASM Addresses for Statements

- Solution: not that easy...
  - Implemented as LLVM IR instruction metadata

```
%x = load float, float* @global, !_asm_addr !0
!0 = !{i64 0x1000}
```

  - Hacked LLVM passes to propagate it through
  - Added the concept of addresses to an entire backend
- Shortcomings:
  - Added the concept only to Statements, not Expressions

```
tmp = ackermann(5)
res = tmp
```

optimization

```
res = ackermann(5)
```

  - More work on this needed...

Improvements: 3rd-Party-Friendly Output

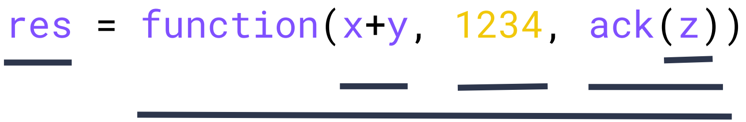
- Original state:
  - C and Python-like text output
- Problems:
  - What if RetDec users want to automatically process the output?
    - E.g. syntax highlighting in IDA plugin
  - What if we want to attach additional info to the output?
    - Assembly addresses for output C lines
- Solution:
  - Serialize plain text output to JSON
  - Dropped Python-like output

Improvements: 3rd-Party-Friendly Output

- JSON output:
  - Stream of tokens that make up the source code
  - Similar to what would lexer would produce
  - Custom additional information entries
- Address entries:
  - Original design:

```
[(line , address)]
```

  - Not a single address (Hex-Rays knows it):



```
res = function(x+y, 1234, ack(z))
```

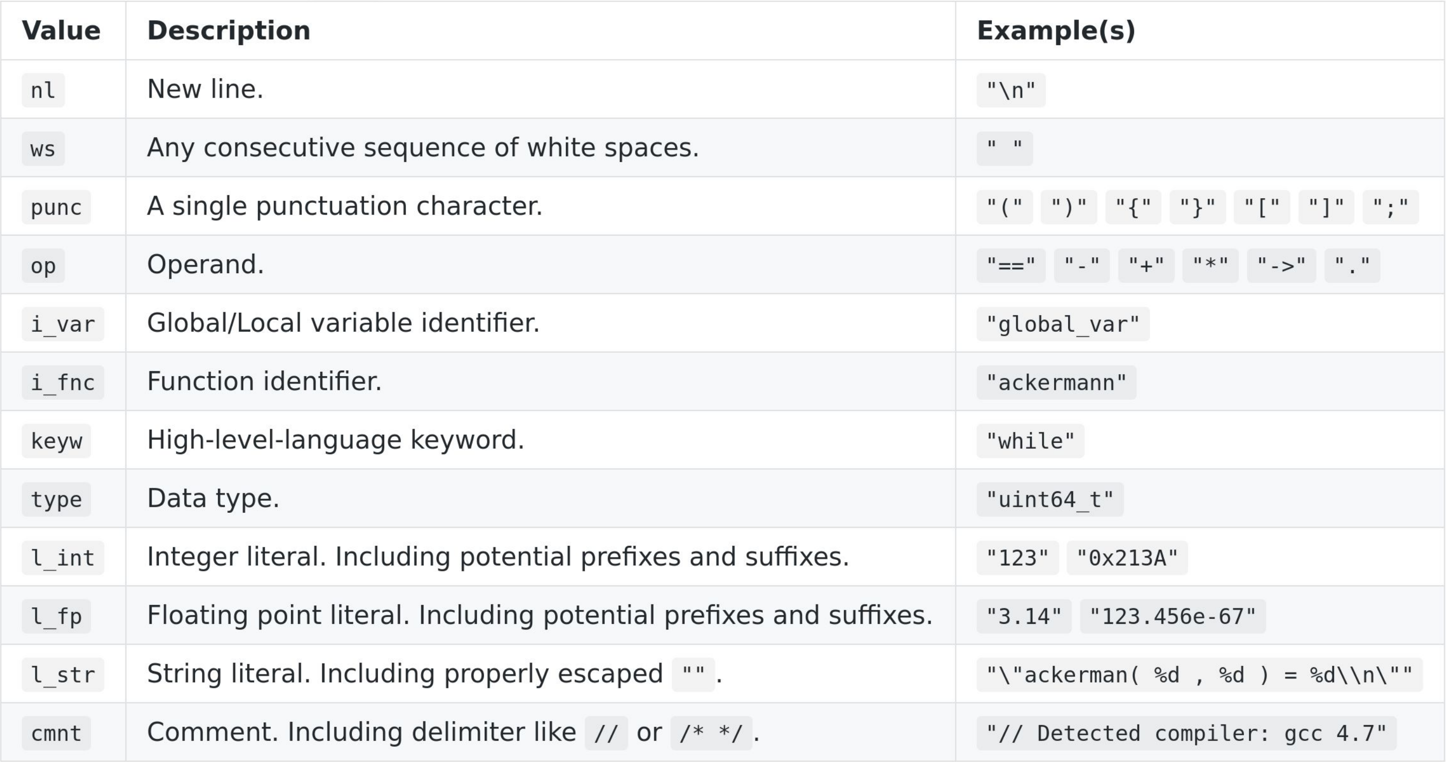
  - Address is a modifier of subsequent token stream

Improvements: 3rd-Party-Friendly Output

```
{
     "language" : "C",
     "tokens" :
     [
          {
               "addr" : "0x1000"
          },
          {
               "kind" : "type",
               "val" : "int32_t"
          },
          {
               "kind" : "i_fnc",
               "val" : "iterative_ackermann"
          },
          {
               "addr" : "0x1005"
          },
          // ...
     ]
}
```

Improvements: 3rd-Party-Friendly Output

https://github.com/avast/retdec/wiki/Decompiler-outputs



Value	Description	Example(s)
`nl`	New line.	`"\n"`
`ws`	Any consecutive sequence of white spaces.	`" "`
`punc`	A single punctuation character.	`"("` `")"` `"{"` `"}"` `"["` `"]"` `";"`
`op`	Operand.	`"=="` `"-"` `"+"` `"*"` `"->"` `"."`
`i_var`	Global/Local variable identifier.	`"global_var"`
`i_fnc`	Function identifier.	`"ackermann"`
`keyw`	High-level-language keyword.	`"while"`
`type`	Data type.	`"uint64_t"`
`l_int`	Integer literal. Including potential prefixes and suffixes.	`"123"` `"0x213A"`
`l_fp`	Floating point literal. Including potential prefixes and suffixes.	`"3.14"` `"123.456e-67"`
`l_str`	String literal. Including properly escaped `""`.	`"\"ackerman( %d , %d ) = %d\\n\""`
`cmnt`	Comment. Including delimiter like `//` or `/* */`.	`"// Detected compiler: gcc 4.7"`

Demo

Near Future

- Radare2 info extraction:
  - Extract and use more/all available info
  - All kinds of comments, argument locations (registers, etc.)
- Bugs and improvements:
  - PoC/Alpha version → expect some problems → solve them as they come
  - Selective decompilation: wasn’t the prefered mode so far
- GUI = Cutter:
  - We know nothing about it, except we want it
  - We don’t like GUI programming

Possible Future

- For discussion:
  - How much should RetDec trust info from r2?
  - How much should RetDec use its own analyses and potentially deviate from info in r2?
  - Should we propagate info from RetDec back to r2 if we think it is better? Should it be on user demand (manual) or automatic?
  - We could make all this configurable, but it would also make all of it more complicated
  - There is plenty of other tools in the RetDec toolset, would there be interest in bringing them to r2 via RetDec plugin or separate plugins?
    - Fileinfo, statically linked code detection, YARA scanners, type information, ...

Ideal Future

avast

Questions?

https://retdec.com

https://github.com/avast/retdec

https://twitter.com/retdec