Ghidra To The Next Level

Ding Zhanzhao (anciety)

Portal Lab, StarCross Tech.

TRACK 2

About Me

- Ding Zhanzhao | Anciety @StarCross PortalLab
- r3kapig CTF player (pwn/re) 2017-2021
  - Team Leader since 2019
  - DEFCON Final Qualified 2017-2021
  - 0CTF 1st
  - WCTF 3rd
  - ...
- Open Source Dev
- Rev/Bin analysis tool enthusiast

Content

- How current ghidra works
- BinCraft
  - Dedication
  - Things Done
  - Things About to Be Done
  - Lessons Learned

What are needed in reversing?

F5

What are needed in reversing?

- Static Analysis
  - Read assembly: disassembly
  - Read C-like Code: decompilation
    - And those that improve decompilation
- Dynamic Analysis
  - Emulation
  - Debugging

What we have now?

- IDA: the ruler
- Ghidra: open sourced by NSA since 2019
- Binary Ninja: still improving
- Rev.ng?
- Radare2?

About Ghidra

- Open Sourced in 2019
  - written in Java + Native Lang (C++)
- (almost) Feature Complete
  - Decompiler √
  - Graphs √
  - Debugger √
  - extensible: Scripting/Plugin √
- Great Architecture: sleigh DSL, etc.
  - Extensible lifting (translate binary => IR + assembly)
- Only One in Open Source World that competes with IDA

About Ghidra

- Problems?
- Historical (Code, Experience)

About Ghidra

Use C99 types in decompiler #2880

Closed **XVilka** wants to merge 1 commit into `NationalSecurityAgency:master` from `XVilka:fix-types`

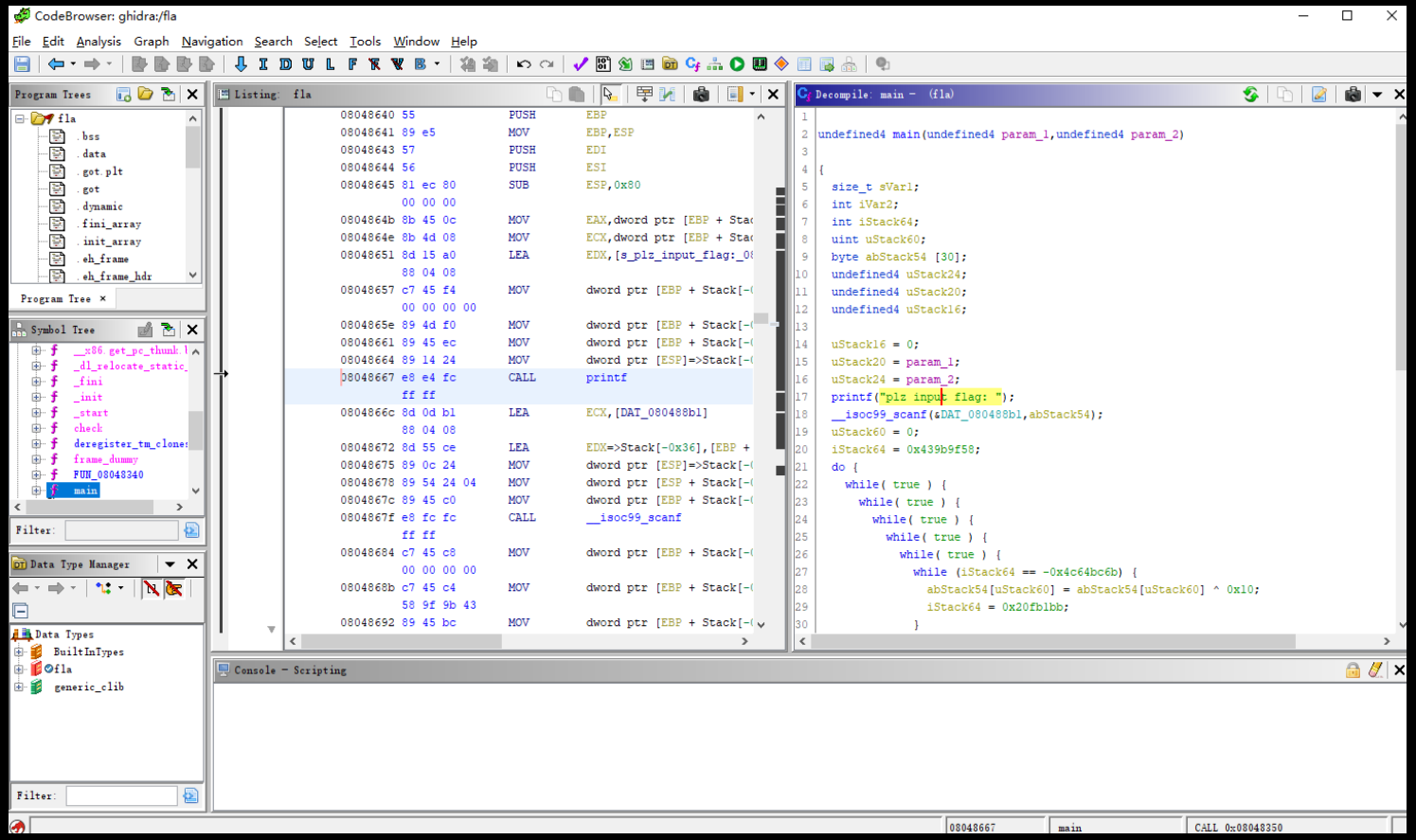
Conversation 2 · Commits 1 · Checks 0 · Files changed 1

XVilka commented on 30 Mar

I suppose this part was written when C/C++ compilers were not as good as today. These days, even MSVC suppports C99 standard, thus lets reduce the code churn and easier future porting to new platforms.

1

About Ghidra



BinCraft - Dedication

- Ghidra is nice already
- But we want more!
  - Modern experience
  - Modern features (apart from officially supported)
  - Modern Code (gradually)

BinCraft - Dedication (cont.)

- vs. Official Ghidra: improve while we use
  - better understanding of what WE need
- Why not just contribute to official?
  - Flavor Differs: decompiled code style, UI...
  - Tune the tool OUR way
  - Provide one more choice!

BinCraft - what

- A collection of ghidra related projects
- **ghidracraft -- forked ghidra**
- sleighcraft -- Rust API to sleigh processor
- pcodecraft -- API to abstract ghidra decompiler

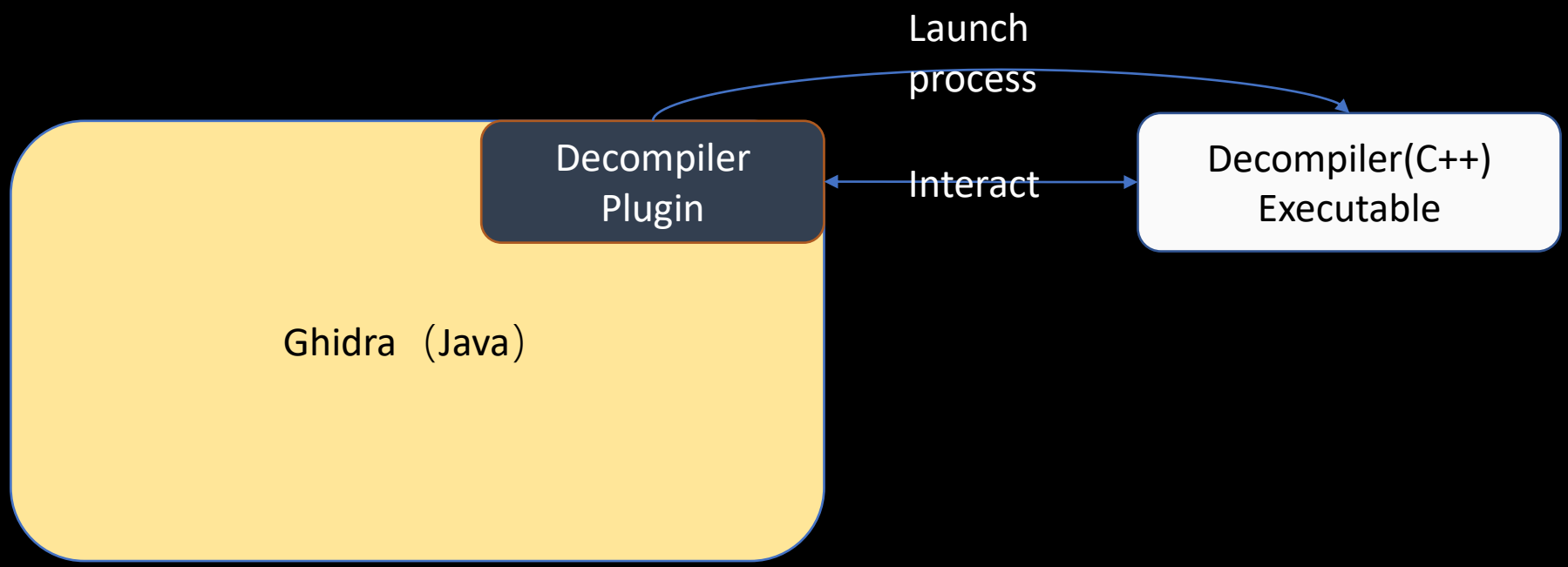
ghidracraft - what done

- Decompilation minor improvements (style tuning)
- Pcode Patch feature
- (Partially) Modernize UI style

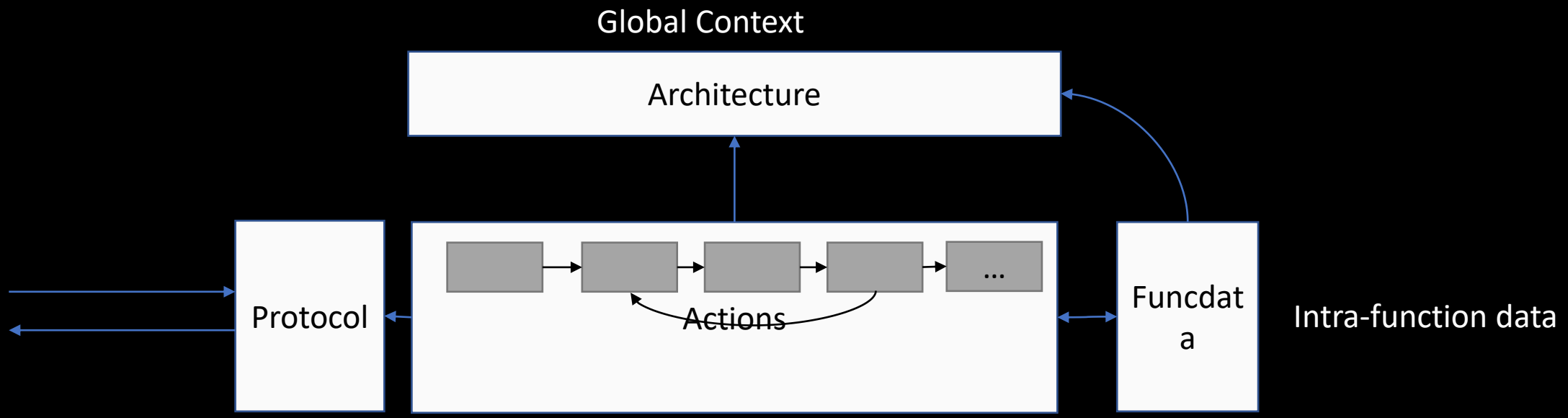
About Decompilation

- Complex Code
  - Complex Algorithm
  - Complex Architecture
  - C++
- DIFFICULT TO TUNE

Ghidra Decompiler



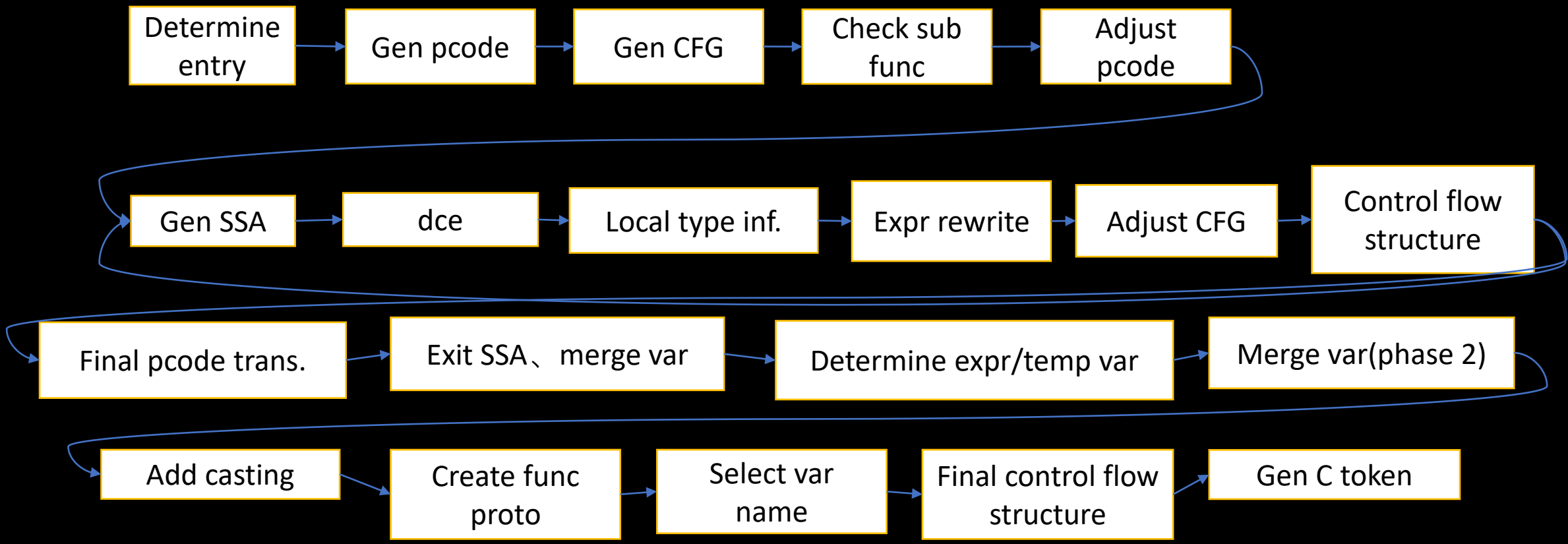
Ghidra Decompiler C++



Ghidra Decompiler C++

- IR——Pcode
- Variables：Varnode
  Address Space：register、ram（memory）、unique（temp）、const
  offset
  size
  E.g：(register, 0x0, 0x8) -> rax; (register, 0x0, 0x4) -> eax;
- Operations
  Arith: INT_ADD、INT_SUB……
  Control flow：BRANCH、CBRANCH……
  Special：MULTIEQUAL（phi-node）、INDIRECT……

Ghidra Decompiler C++



Problems in Decompilation

- Magic functions: CONCAT, ZEXT...
- Lengthy default var names
- Analysis failure
- Other minor issues
  - display format convert...

Magic Functions

- Reason: Pcode not translatable
- Format: CONCAT44 -> concat 4 bytes and 4 bytes into 8 bytes
- Better way? type casting
- Add rewrite rule

```
__ptr = (void *)CONCAT44(uStack156,local_a0);
```

```
__ptr = (void *)((uVar1 << 0x20) + local_a0);
```

Default Var names

```
undefined4 main(undefined4 param_1,undefined4 param_2)

{
  size_t Var1;
  int Var2;
  int iStack64;
  uint uStack60;
  byte abStack54 [30];
  undefined4 uStack24;
  undefined4 uStack20;
  undefined4 uStack16;

  uStack16 = 0;
  uStack20 = param_1;
  uStack24 = param_2;
  printf("plz input flag: ");
  __isoc99_scanf(&DAT_080488b1,abStack54);
```

```
undefined4 main(undefined4 p1,undefined4 p2)

{
  size_t v1;
  int v2;
  int s4;
  uint s5;
  byte s3 [30];
  undefined4 s6;
  undefined4 s7;
  undefined4 s8;

  s8 = 0;
  s7 = p1;
  s6 = p2;
  printf("plz input flag: ");
  __isoc99_scanf(&DAT_080488b1,s3);
```

Analysis Failure

- DEFCON quals 2021 rad
- Stack Var failure
- Function Params Failure

```
122
123   auStack4144[0] = 0x178a8e;
124   lVar28 = __rust_probestack();
125   lVar28 = -lVar28;
126   *(undefined8 *)((long)auStack4144 + lVar28) = 0x178a97;
127   env_logger::init();
128   *(undefined8 *)((long)auStack4144 + lVar28) = 0x178aa9;
129   std::path::Path::is_file();
130   if (extraout_AL == '\0') {
131     *(undefined8 *)((long)auStack4144 + lVar28) = 0x178b43;
132     State::new(&stack0xffffffffffffffd8 + lVar28);
133     lVar33 = *(long *)(&stack0xffffffffffffffd8 + lVar28);
134     uVar35 = *(ulong *)(&stack0xfffffffffffffee0 + lVar28);
135     pvVar42 = *(void **)(&stack0xfffffffffffffee8 + lVar28);
136     pvVar29 = *(void **)(&stack0xfffffffffffffef0 + lVar28);
137     *(undefined8 *)((long)aplStack400 + lVar28 + 8) =
138          *(undefined8 *)(&stack0xfffffffffffffff8 + lVar28);
139     *(undefined8 *)((long)auStack4144 + lVar28) = 0x178b8e;
140     memcpy(&stack0x00008ff8 + lVar28,&stack0x00000000 + lVar28,0x8fd8);
141     if (lVar33 != 1) {
142       *(undefined8 *)((long)auStack4144 + lVar28) = 0x178be9;
143       memcpy(&stack0x00011fd8 + lVar28,&stack0x00009fd0 + lVar28,0x8000);
144       *(undefined8 *)((long)auStack4144 + lVar28) = 0x178bf9;
145       std::alloc::__default_lib_allocator::__rust_alloc();
146       if (__dest == (void *)0x0) goto LAB_0017b6ca;
147       *(undefined8 *)((long)auStack4144 + lVar28) = 0x178c1b;
148       memcpy(__dest,&stack0x00011fd8 + lVar28,0x8000);
149       *(void **)((long)&pvStack488 + lVar28 + 8) = __dest;
150       pvVar29 = (void *)0x0;
151       *(ulong *)((long)&uStack496 + lVar28) = 0;
152       __ptr = __dest;
153       goto LAB_00179589;
154     }
```

Analysis Failure

- And rsp, X prevents stack var analysis
- Rsp+X => X constant propagated.
- If not, fails!
- Imperfect Solution: ignore

```
2  void entry(void)
3
4  {
5    ulong uVar1;
6
7    uVar1 = (ulong)&stack0xfffffffffffffd0 & 0xfffffffffffff000;
8    *(undefined8 *)(uVar1 - 0x11fe0) = 1;
9    *(undefined8 *)(uVar1 - 0x11fd8) = 1;
10   *(undefined8 *)(uVar1 - 0x1c008) = 0x401047;
11   test_func(uVar1 - 0x11fe0,1);
12   return;
13 }
14
```

Decompile: entry - (aand.out)

```
1
2  void entry(void)
3
4  {
5    undefined8 uStack77832;
6    undefined8 uStack77824;
7
8    uStack77832 = 1;
9    uStack77824 = 1;
10   test_func(&uStack77832,1);
11   return;
12 }
13
```

Analysis Failure

- Empty Params
- Rust use empty params when optimized
- Solution: ignore param info if Rust and optimized
- (merged)

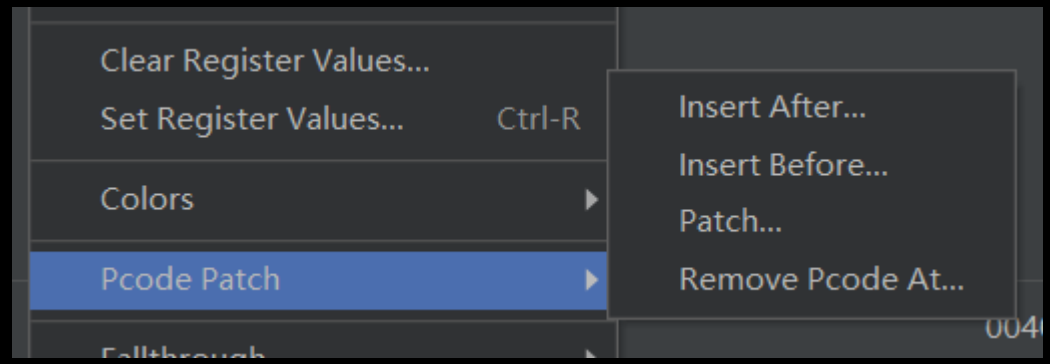
```
env_logger::init();
std::path::Path::is_file();
if (extraout_AL == '\0') {
  State::new(&bStack114728);
  uVar239 = uStack114700;
  uVar225 = uStack114704;
```

Analysis Failure

- String display issue

Pcode Patch

- pcode patch: support scripting and **manual**!



Pcode Patch

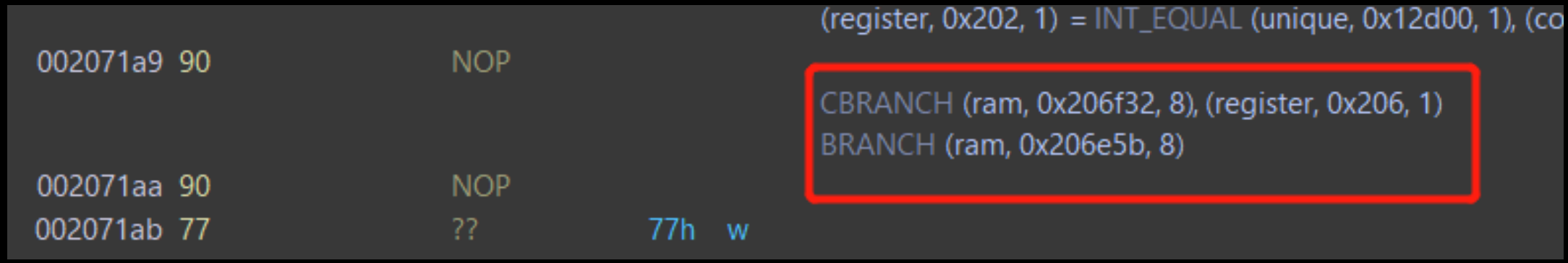
- Vs IDA: microcode API
  - Not a “pass” (works on phase)
  - Support manual

Pcode Patch

- Use case
  - quickly modify program semantic (don’t care about padding)
  - Deobfuscation (without carry about byte length)
  - Total custom decompilation
    - Provides Pcode the IR
    - Gives you the decompiled code

Pcode Patch

- Manipulating Program Semantic Manually



Pcode Patch

- Use pcode patch against OLLVM: x86, arm, aarch, mips, PPC

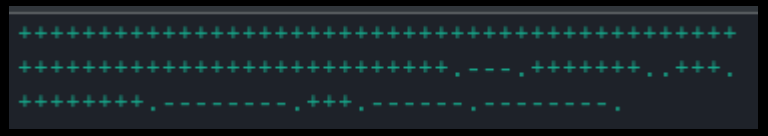
```
/* check3(char*) */

void check3(char *param_1)

{
  int iVar1;
  int local_20;

  iVar1 = check2(param_1);
  local_20 = 0xf607ae;
  while( true ) {
    while (local_20 == 0xf607ae) {
      local_20 = 0x5819697a;
      if (iVar1 == 0) {
        local_20 = 0x4bf1b86e;
      }
    }
    if (local_20 == 0x31271051) break;
    if (local_20 == 0x4bf1b86e) {
      printf("error!\n");
      local_20 = 0x31271051;
    }
    else {
      if (local_20 == 0x5819697a) {
        printf("you get it!\n");
        local_20 = 0x31271051;
      }
    }
  }
  return;
}
```

```
/* check3(char*) */

void check3(char *param_1)

{
  uint uVar1;
  undefined8 in_RSI;

  uVar1 = check2(param_1);
  if (uVar1 == 0) {
    printf("error!\n",in_RSI,(ulong)uVar1,0x4bf1b86e);
  }
  else {
    printf("you get it!\n",in_RSI,(ulong)uVar1,0x4bf1b86e);
  }
  return;
}
```

Pcode Patch

- Custom Decompilation: patch the IR, get the decompilation
  - Create a new segment
  - Patch your IR there
  - Get your decompilation and profit!
- Circumstances
  - get a weird instruction set, want the decompilation
  - Want to skip the sleigh engine (no binary!), get only the decompilation

Pcode Patch



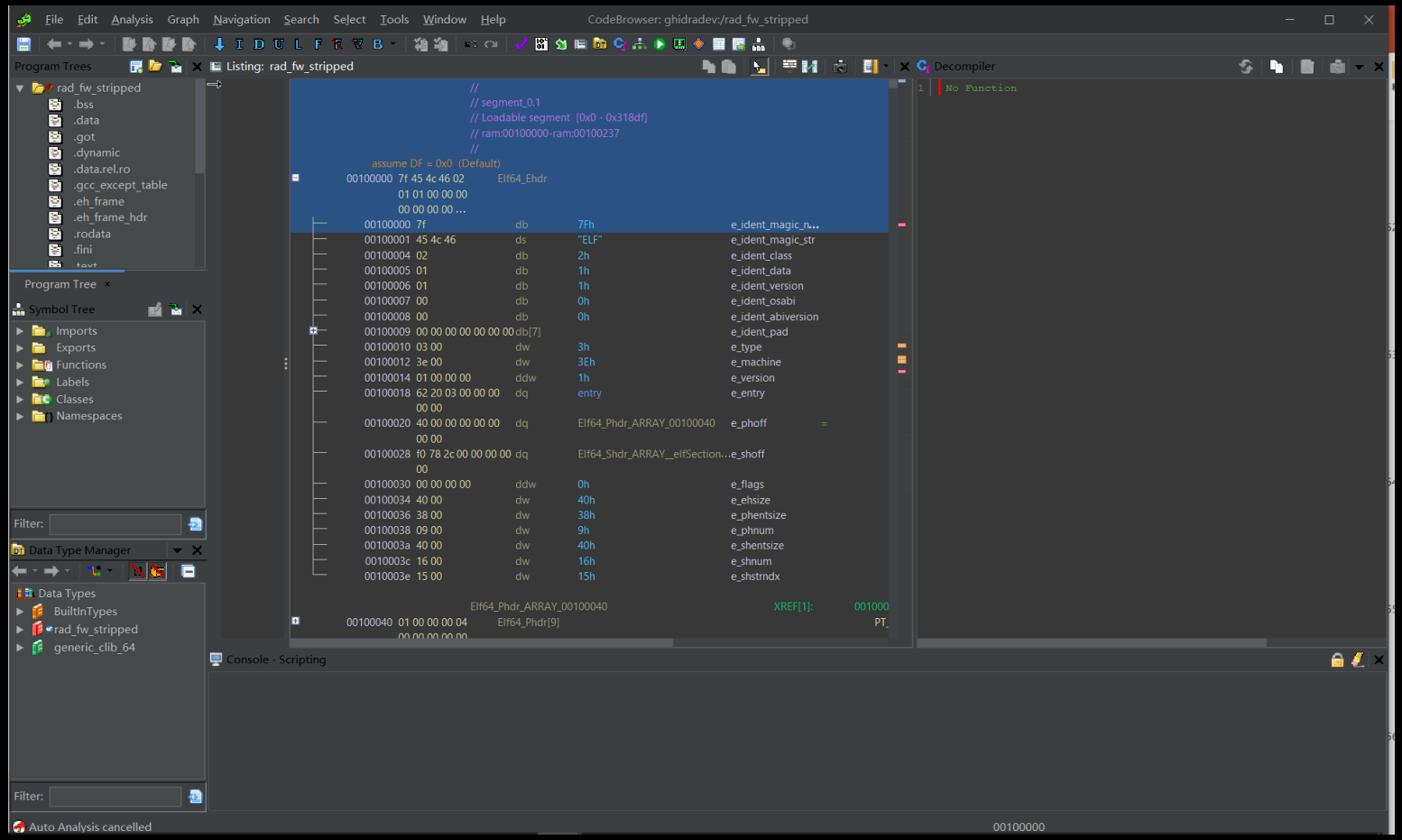
Decompile: main - (demo_empty_bf)

```
undefined  [16] main(void)

{
  ulong uVar1;

  _memory = (void *)((long)_memory + 0x48);
  uVar1 = 0;
  write(1,_memory,1);
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + -1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + -1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + -1;
  write(1,*(void **)(&memory + uVar1),1);
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + 1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + 1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + 1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + 1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + 1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + 1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + 1;
  write(1,*(void **)(&memory + uVar1),1);
  write(1,*(void **)(&memory + uVar1),1);
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + 1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + 1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + 1;
  write(1,*(void **)(&memory + uVar1),1);
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + 1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + 1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + 1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + 1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + 1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + 1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + 1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + 1;
  write(1,*(void **)(&memory + uVar1),1);
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + -1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + -1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + -1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + -1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + -1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + -1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + -1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + -1;
  write(1,*(void **)(&memory + uVar1),1);
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + 1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + 1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + 1;
  write(1,*(void **)(&memory + uVar1),1);
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + -1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + -1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + -1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + -1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + -1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + -1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + -1;
  write(1,*(void **)(&memory + uVar1),1);
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + -1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + -1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + -1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + -1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + -1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + -1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + -1;
  *(long *)(&memory + uVar1) = *(long *)(&memory + uVar1) + -1;
  write(1,*(void **)(&memory + uVar1),1);
  return ZEXT816(uVar1) << 0x40;
}
```

Modernized UI

- Fix some classical stylings
- Add configurable color-setting

Modernized UI



What to be done

- GraalVM integration

Graal VM

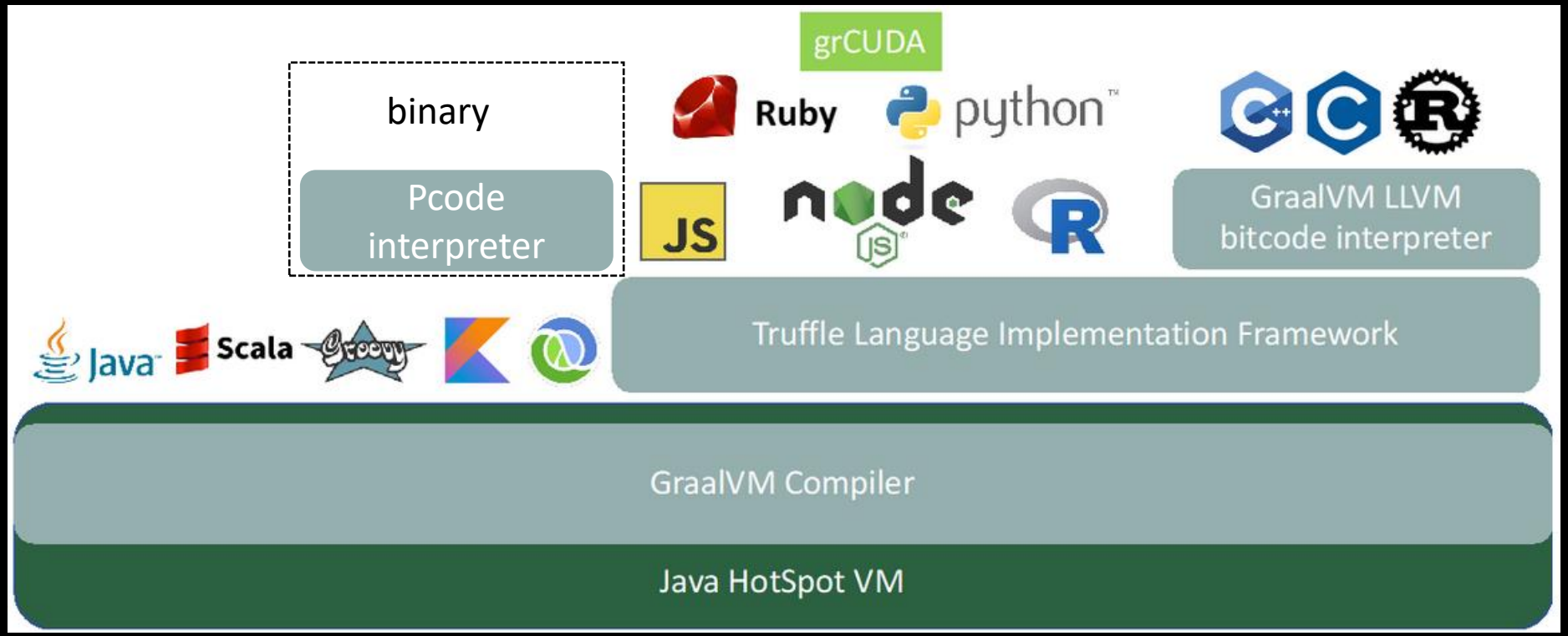
- Oracle Open source Compiler framework
  - jitted language implementation (even LLVM)
  - general instrumentation framework
- To us?
  - jitted pcode emulation (vs. current impl)
  - instrumentation (and more! tainted analysis? symbolic execution?)

Graal VM

- Interpreter => jitted code
  - By Partial Evaluation
- What do we get?
  - Write pcode interpreter => jitted!
  - Oh, wait, instrumentation works?

Graal VM

Binary Analysis with GraalVM!



Graal VM

- Vs LLVM impl(sulong)
- Both have unstructured control flow
- Both low-level
- Follow!
  - Manuel Rigger, Matthias Grimmer, Hanspeter Mössenböck Sulong – Execution of LLVM-Based Languages on the JVM
  - Manuel Rigger, Matthias Grimmer, Christian Wimmer, Thomas Würthinger, Hanspeter Mössenböck Bringing low-level languages to the JVM: efficient execution of LLVM IR on Truffle

Graal VM impl. stages.

- Stage 0: refactor current emulation to allow reuse √
- Stage 1: interpreter in Truffle √
- Stage 2: tune the interpreter to allow jitting
  - Follow sulong (LLVM) implementation
- Stage 3: instrumentation framework
  - Encapsulation
  - tainted analysis
  - symbolic execution

Lessons Learned

- Ghidra is a huge beast
  - Problem: huge, sometimes doc missing
  - Advantage: complete
  - Needed: time and community

Thanks For Listening

Star us!

https://github.com/StarCrossPortal/bincraft

https://github.com/StarCrossPortal/ghidracraft