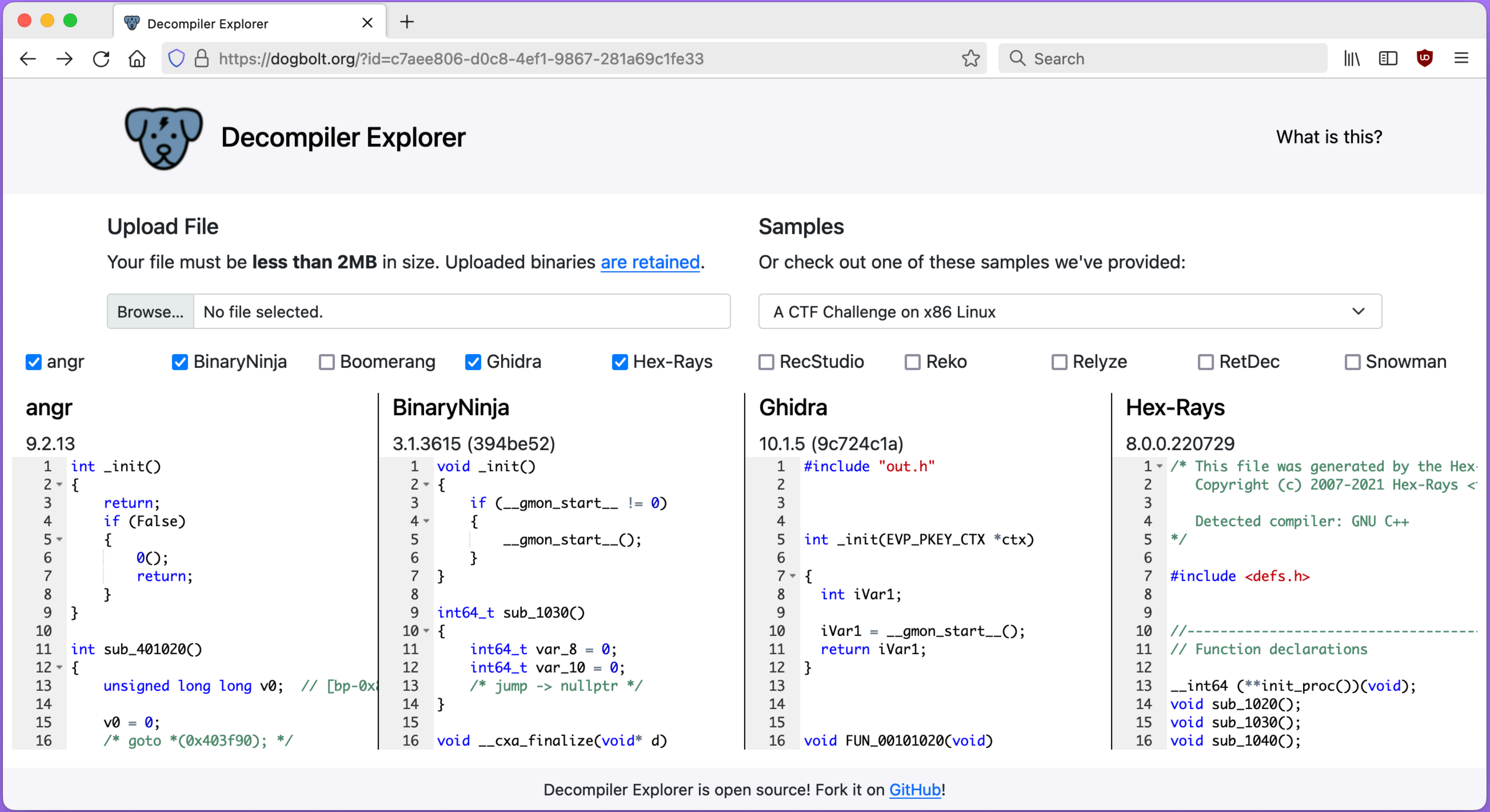
Decomperson

how humans decompile and what we can learn from it

Kevin Burk, Fabio Pagani, Christopher Kruegel, and Giovanni Vigna
University of California, Santa Barbara



Decompiler Explorer

https://dogbolt.org/?id=c7aee806-d0c8-4ef1-9867-281a69c1fe33

Decompiler Explorer

What is this?

Upload File

Your file must be **less than 2MB** in size. Uploaded binaries are retained.

Browse... No file selected.

Samples

Or check out one of these samples we've provided:

A CTF Challenge on x86 Linux

☑ angr ☑ BinaryNinja ☐ Boomerang ☑ Ghidra ☑ Hex-Rays ☐ RecStudio ☐ Reko ☐ Relyze ☐ RetDec ☐ Snowman

angr

9.2.13

```
int _init()
{
    return;
    if (False)
    {
        0();
        return;
    }
}

int sub_401020()
{
    unsigned long long v0;  // [bp-0x

    v0 = 0;
    /* goto *(0x403f90); */
```

BinaryNinja

3.1.3615 (394be52)

```
void _init()
{
    if (__gmon_start__ != 0)
    {
        __gmon_start__();
    }
}

int64_t sub_1030()
{
    int64_t var_8 = 0;
    int64_t var_10 = 0;
    /* jump -> nullptr */
}

void __cxa_finalize(void* d)
```

Ghidra

10.1.5 (9c724c1a)

```
#include "out.h"



int _init(EVP_PKEY_CTX *ctx)

{
  int iVar1;

  iVar1 = __gmon_start__();
  return iVar1;
}



void FUN_00101020(void)
```

Hex-Rays

8.0.0.220729

```
/* This file was generated by the Hex
   Copyright (c) 2007-2021 Hex-Rays <

   Detected compiler: GNU C++
*/

#include <defs.h>


//-------------------------------------
// Function declarations

__int64 (**init_proc())(void);
void sub_1020();
void sub_1030();
void sub_1040();
```

Decompiler Explorer is open source! Fork it on GitHub!

Reversing Studies

a traditional approach

- Reversers communicate in natural language
- Researchers use interviews to follow the reversing process
- Depth of understanding can be hard to quantify

How can we scale this up?

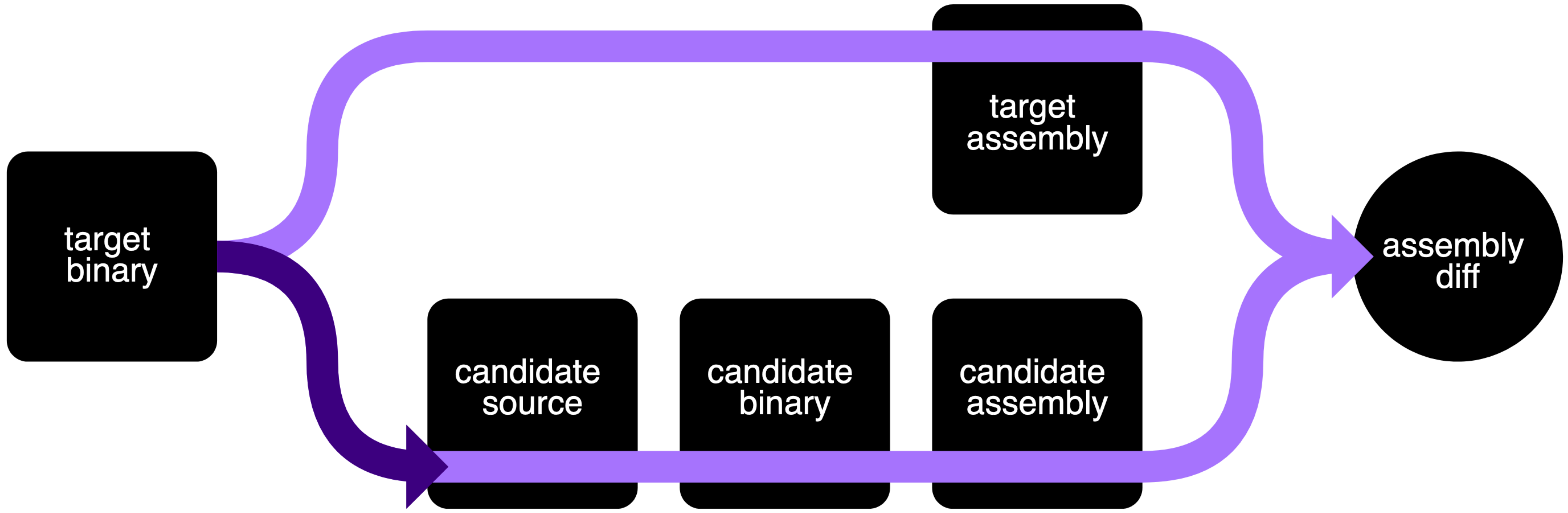
Reversing Studies

the decomperson approach

- Reversers communicate in source code
- Researchers use code diffs to follow the reversing process
- Quantify understanding as the quality of the assembly match
- Perfect understanding = perfect decompilation

Perfect Decompilation

the verification process



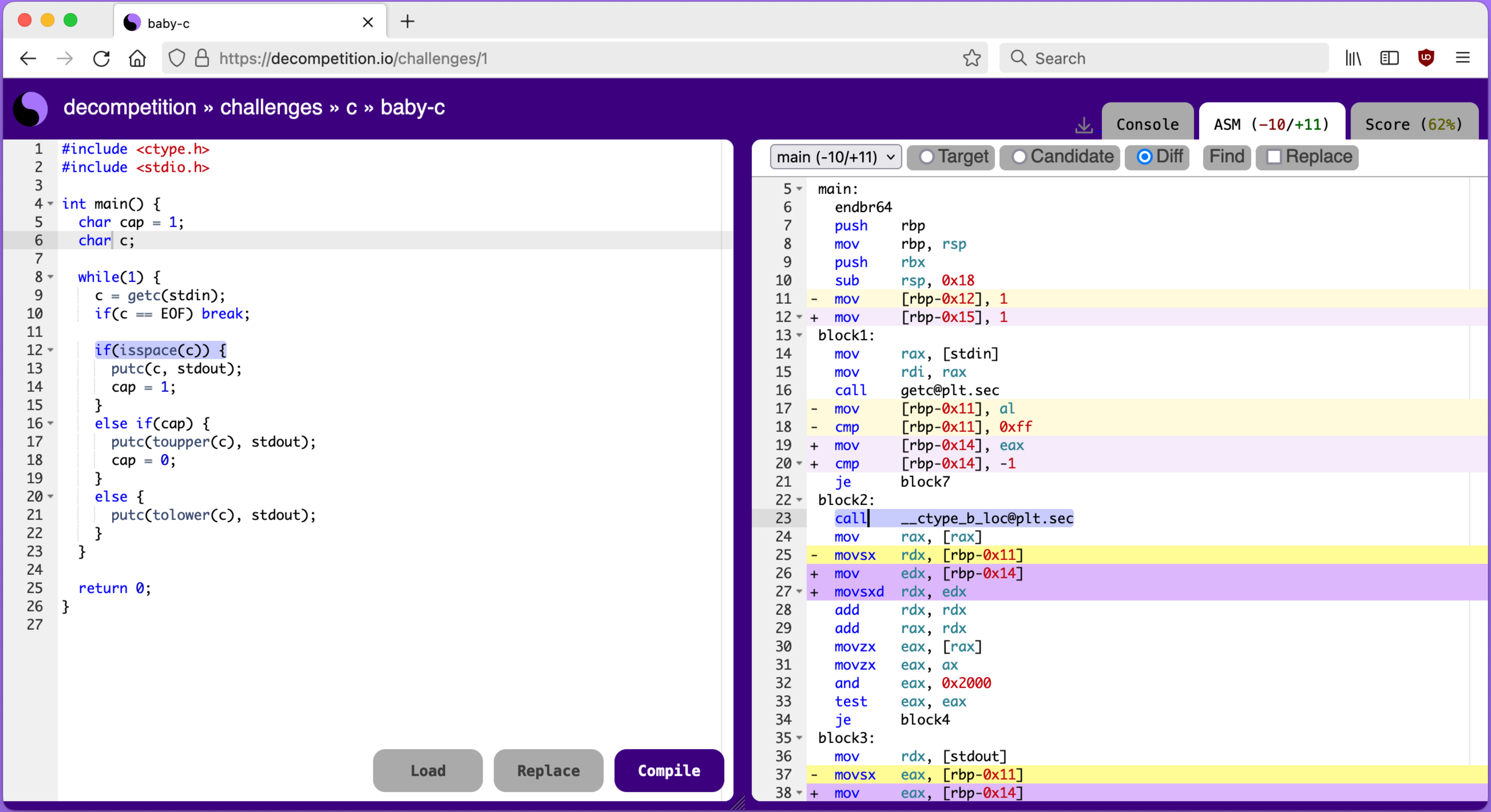
Perfect Decompilation

research questions

- Can humans do it?
- If so, how do they do it?
- Is this representative of the traditional reversing process?

Decompetition

a perfect decompilation competition



```c
#include <ctype.h>
#include <stdio.h>

int main() {
  char cap = 1;
  char c;

  while(1) {
    c = getc(stdin);
    if(c == EOF) break;

    if(isspace(c)) {
      putc(c, stdout);
      cap = 1;
    }
    else if(cap) {
      putc(toupper(c), stdout);
      cap = 0;
    }
    else {
      putc(tolower(c), stdout);
    }
  }

  return 0;
}
```

```
  main:
    endbr64
    push    rbp
    mov     rbp, rsp
    push    rbx
    sub     rsp, 0x18
-   mov     [rbp-0x12], 1
+   mov     [rbp-0x15], 1
  block1:
    mov     rax, [stdin]
    mov     rdi, rax
    call    getc@plt.sec
-   mov     [rbp-0x11], al
-   cmp     [rbp-0x11], 0xff
+   mov     [rbp-0x14], eax
+   cmp     [rbp-0x14], -1
    je      block7
  block2:
    call    __ctype_b_loc@plt.sec
    mov     rax, [rax]
-   movsx   rdx, [rbp-0x11]
+   mov     edx, [rbp-0x14]
+   movsxd  rdx, edx
    add     rdx, rdx
    add     rax, rdx
    movzx   eax, [rax]
    movzx   eax, ax
    and     eax, 0x2000
    test    eax, eax
    je      block4
  block3:
    mov     rdx, [stdout]
-   movsx   eax, [rbp-0x11]
+   mov     eax, [rbp-0x14]
```

Decompetition

challenges and languages

4	5	7	4	3
C Challenges	**C++** Challenges	**Go** Challenges	**Rust** Challenges	**Swift** Challenges

Decompetition

challenge scoring

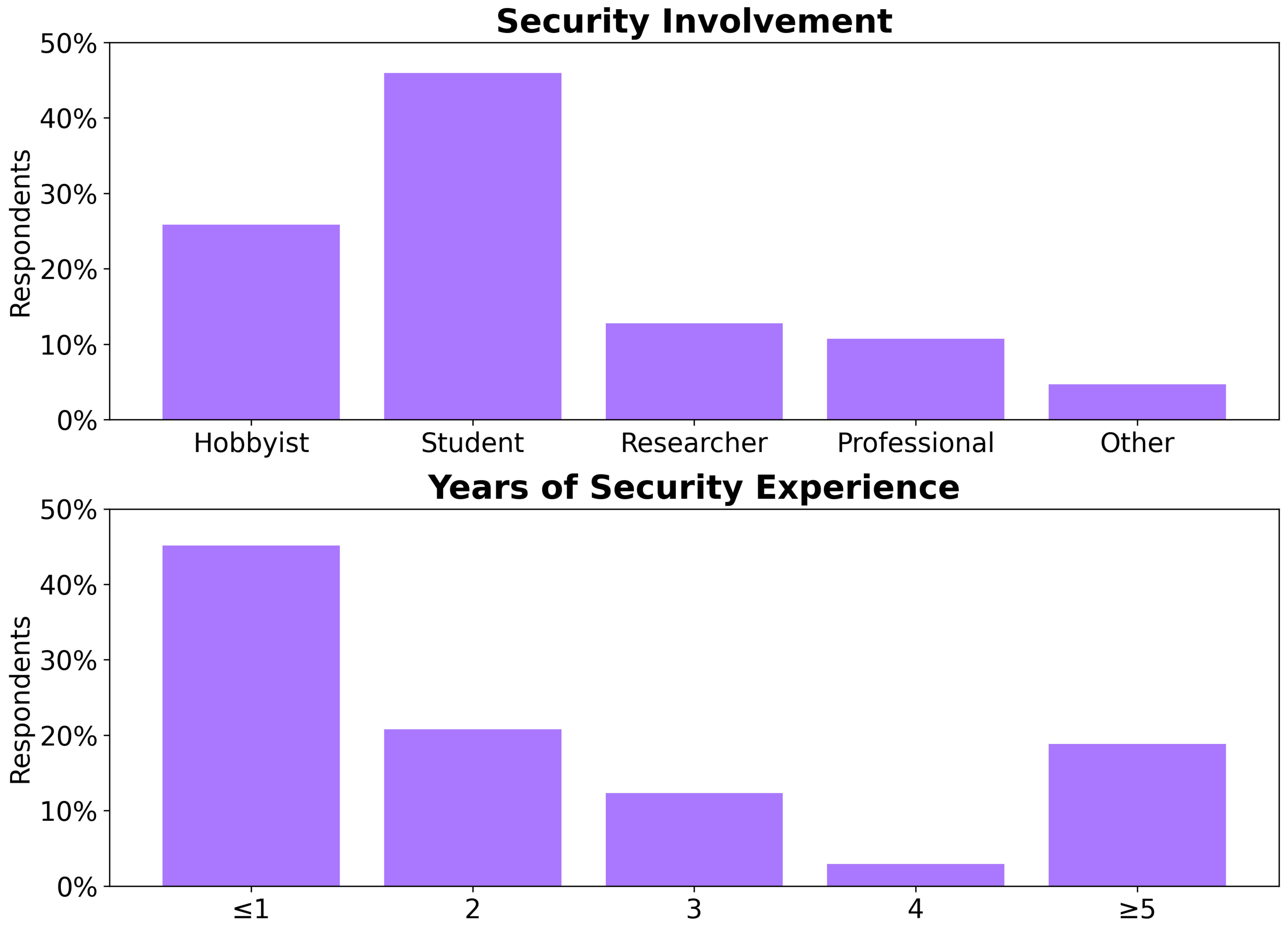
- **20%**
  secret test cases
- **60%**
  assembly diff (Jaccard Index)
- **20%**
  perfect match bonus

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Decompetition

players and statistics

- **188**
  active users
- **139**
  active teams
- **35,530**
  code submissions



Results

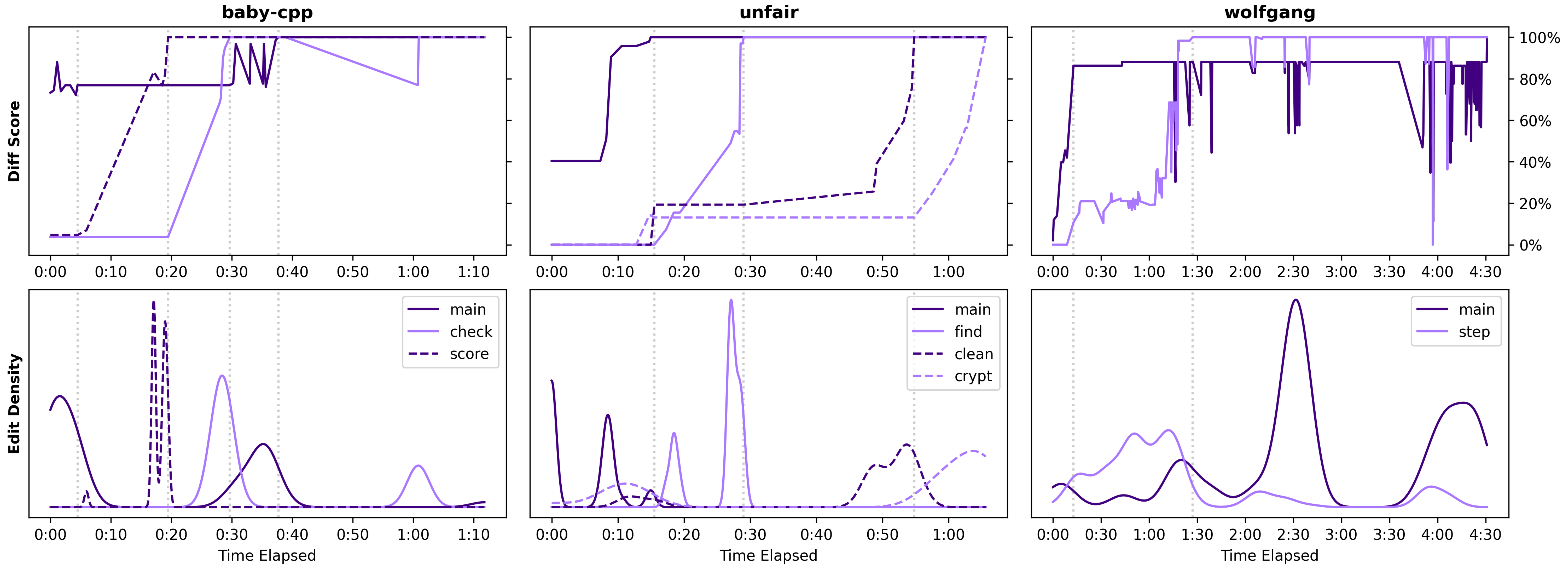
data from decompetition

329

perfect submissions, made by
91 of 188 users, or
66 of 139 teams

The Reversing Process

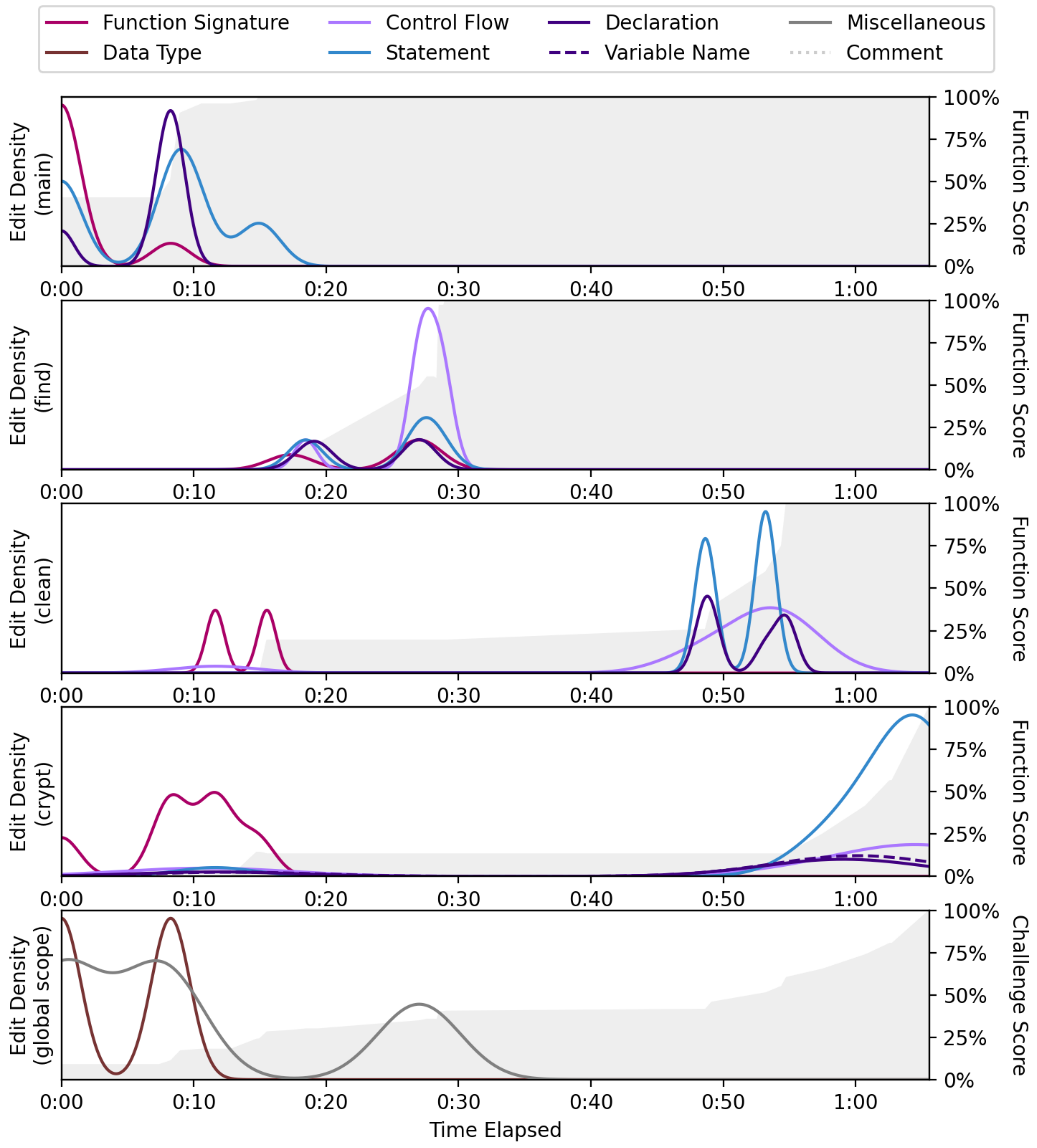
one function at a time



The Reversing Process

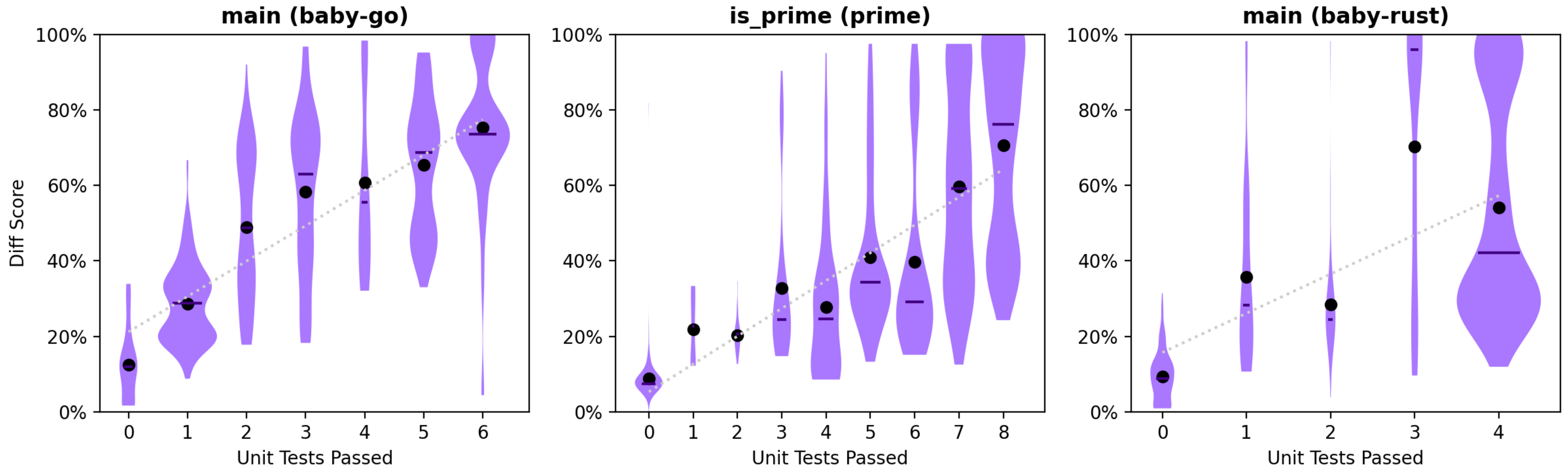
familiar phases

- Get an overview of the binary; stub out functions
- Decide what functions to focus on (not very relevant in Decompetition)
- Make hypotheses about function behavior (as source code); submit to confirm or refute



Diff Scores

reasonably correlated with test scores



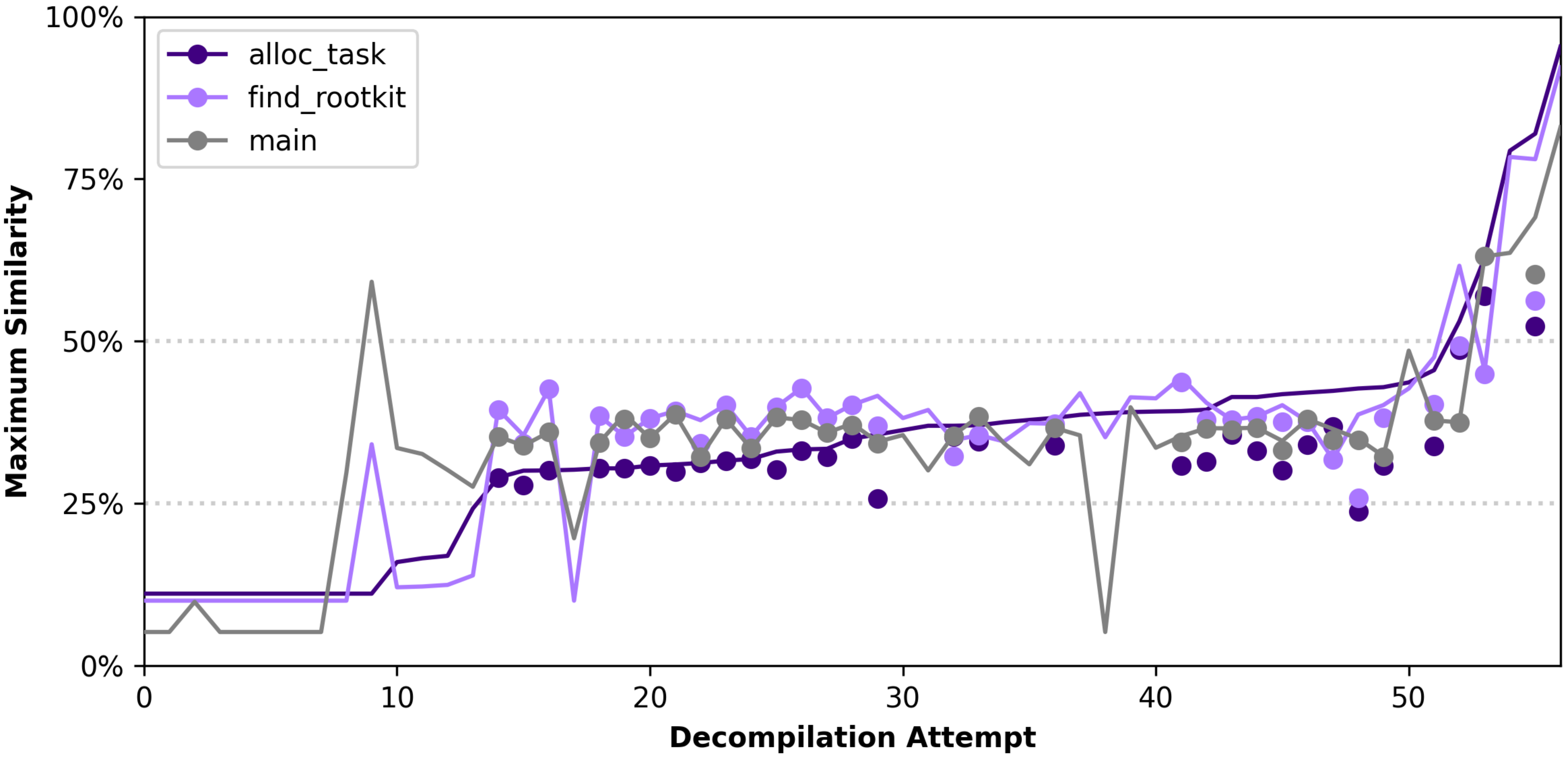
In Summary

decomperson in brief

- Largest reverse engineering study to date
- Used perfect decompilation as a quantitative measure of understanding
- Followed the reversing process programmatically
- Code, challenges, and data available online
  https://github.com/decompetition
  https://decompetition.io
- Thanks for listening!

Existing Decompilers

not yet capable of perfect decompilation


alloc_task
find_rootkit
main
Maximum Similarity
Decompilation Attempt