QuickBinDiff: Efficient Binary Code Diff Tool Based on Command Line Interface

Background

- Source code diff
  - Version control and collaboration
  - Code review
  - Bug tracking and fixing
  - Detect plagirarism and infringement
  - Detect bug or vulnerabilities
- We also use it extensively:
  - Git diff when commit/push/pull/merge

Background

- Source code diff
  - Upside-down view

```
11c11,12
< export CTNG_CONF_PATH CTNG_BIN CTNG_PATH CTNG_TARBALL_PATH
---
> EXTRA_DEP_PATH="$TOOL_PATH/extra_dep"
> export CTNG_CONF_PATH CTNG_BIN CTNG_PATH CTNG_TARBALL_PATH EXTRA_DEP_PATH
14c15
< NUM_JOBS=8
---
> NUM_JOBS=8
```

Background

- Source code diff
  - Side-by-side view

```
CTNG_PATH="$TOOL_PATH/crosstool-ng"                                  CTNG_PATH="$TOOL_PATH/crosstool-ng"
CTNG_CONF_PATH="$PROJ_ROOT/ctng_conf"                                CTNG_CONF_PATH="$PROJ_ROOT/ctng_conf"
CTNG_TARBALL_PATH="$TOOL_PATH/ctng_tarballs"                         CTNG_TARBALL_PATH="$TOOL_PATH/ctng_tarballs"
export CTNG_CONF_PATH CTNG_BIN CTNG_PATH CTNG_TARBALL_PATH         | EXTRA_DEP_PATH="$TOOL_PATH/extra_dep"
                                                                   > export CTNG_CONF_PATH CTNG_BIN CTNG_PATH CTNG_TARBALL_PATH EX
mkdir -p "$CTNG_TARBALL_PATH"                                        mkdir -p "$CTNG_TARBALL_PATH"

NUM_JOBS=8                                                         | NUM_JOBS=8
MAX_JOBS=8                                                           MAX_JOBS=8
export NUM_JOBS MAX_JOBS                                             export NUM_JOBS MAX_JOBS
```

Background

- Binary code diff
  - Malware detection and analysis
  - Code plagirarism and infringement
  - Find bug or vulnerability
  - binary code similarity
  - binary diversifying

Problems

1. The absence of a free, open source, CLI-based diff tools focused on assembly code diff

- GUI-based diff tools focus on **pseudo code diff**
- Slow and automation is not possible
- Most tools require a paid license
  - DarunGrim
    - IDA Pro required
  - diaphora
    - IDA Pro required
  - bindiff
    - IDA Pro required to utilize all features
    - IDA free, Binary Ninja, Ghidra required to utilize limited features

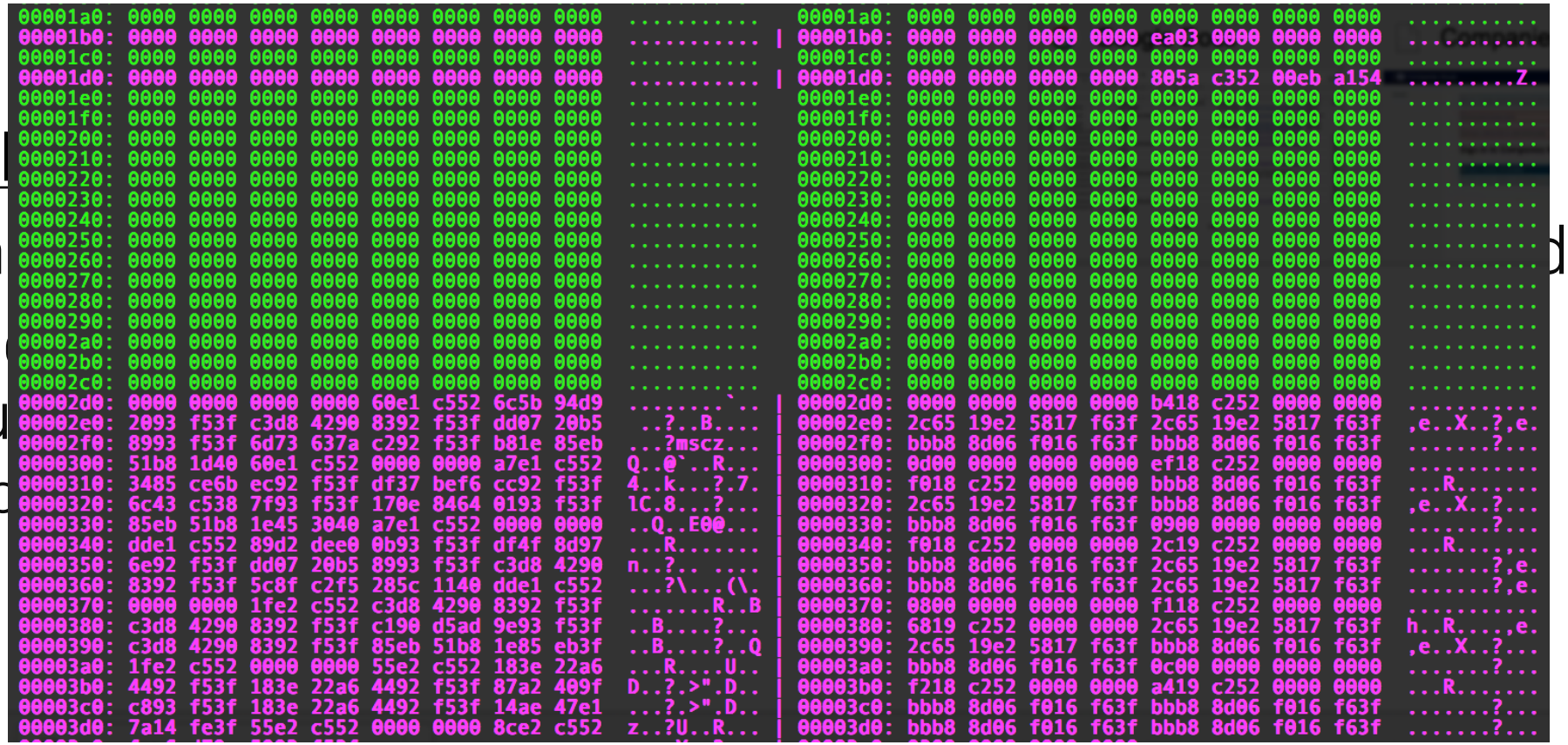
Problems

1. The absence of a free, open-source, CLI-based diff tools focused on assembly code diff

- GLI-based diff tools focus on **bytecode diff**
- Significant loss of information and difficult to understand.
- The only CLI-based assembly code diff tool (elf_diff) requires source code
  - compilation needed including debugging information

Problems

1. The absence of a free, open-source, CLI-based diff tools focused on assembly code diff



- GLI-
- Sign
- The
  requ
    - co

Problems

2. Difficult to automatically handle a large quantity of binaries

- Assuming the license has been purchased
- To check if the target binary group has additional pop-ret gadgets,
  - Open GUI-based diff tool
  - Load two binaries
  - Wait for analysis
  - Manually verify and record the findings

references here

Problems

3. The lack of research on the readability of binary code diff algorithms

- Research on readability in source code diff from a software engineering perspective
  - “How Different Are Different diff Algorithms in Git?”
  - The histogram algorithm excels in the readability of source code diff
- There is a absence of such research for binary code, despite the relative difficulty in understanding binary code

Problems

1. The absence of a free, open source, CLI-based diff tools focused on assembly code diff
2. Difficult to automatically handle a large quantity of binaries
3. The lack of research on the readability of binary code diff algorithms

Goal

- Low-quality CLI tools ← **QuickBinDiff** → High-quality GUI tools

- Why not CLI tools?
  - Only suppurt bytecode diff or need source code to recompile
- Why not GUI tools?
  - Requires a license, slow, and not automatable
  - "Even though we have IDA Pro, we often use gdb or objdump”

Goal

- Free, open source and CLI-based binary code diff tool
  - More detail than bytecode diff
  - More faster then pseudo-code diff
- For Efficiency, utilize text diff algorithm
  - Select an algorithm for assembly matching offers good readability.
- Support cross-platform, various file foramt and multi architecture
  - Platform: Linux, Windows, OS X
  - File format: PE, ELF, Match-O
  - Architecture: X86, X64, ARM, MIPS
- Utilize various information from the binary
  - including Sections, function boundaries, instructions

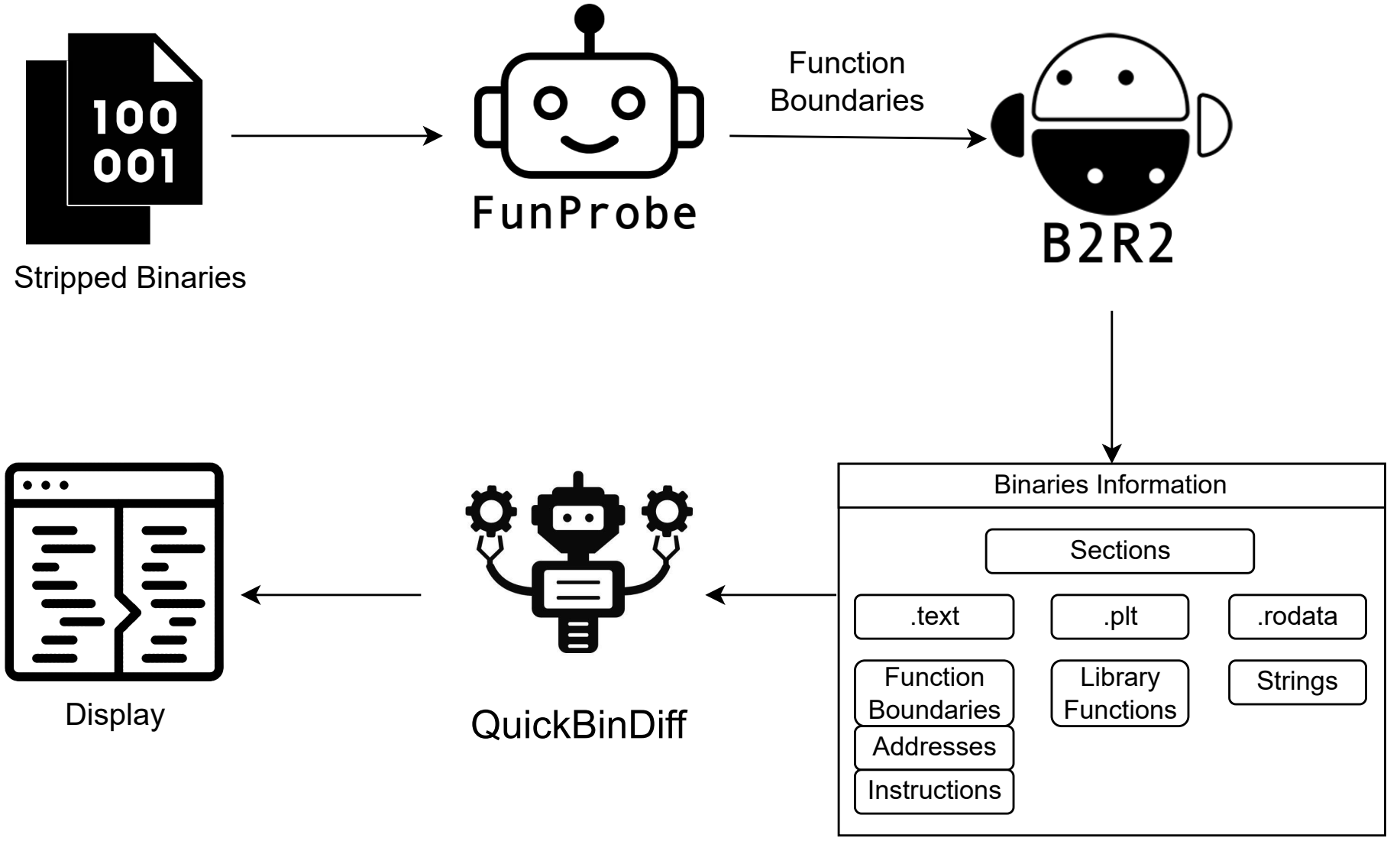
Expected Result

- Fast, free, open source and CLI-based binary code diff tool
- Efficiently perform assembly code diff of two binaries even in limited environments such as servers
- Analyze two binarie codes in detail
- Automatically extract features from a large dataset

Example of Expected Result

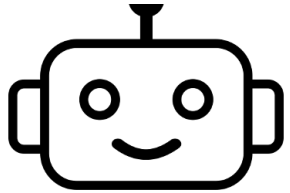
```
<main>                                          |     <main>
..skip..
0x1176 | mov EDI, 0x5                           |     mov EDI, 0x5                          | 0x117a
0x117b | call <func1> ; -0x43                   |     call <func1> ; -0x47                  | 0x117f
0x1180 | mov dword ptr [RBP-0x4], EAX           |     mov dword ptr [RBP-0x4], EAX          | 0x1184
       |                                        | [+] mov dword ptr [RBP-0x8], 0x0          | 0x1187
0x1183 | jmp +0xa ; <main+0x2a>                 |     jmp +0xa ; <main+0x2a>                | 0x118e
0x1185 | add dword ptr [RBP-0x8], 0x1     [-]   |                                           |
0x1189 | sub dword ptr [RBP-0x4], 0x1           |     sub dword ptr [RBP-0x4], 0x1          | 0x1190
       |                                        | [+] add dword ptr [RBP-0x8], 0x1          | 0x1194
0x118d | cmp dword ptr [RBP-0x4], 0x0           |     cmp dword ptr [RBP-0x4], 0x0          | 0x1198
0x1191 | jnz -0xc ; <main+0x22>           [-]   | [+] jz -0xc ; <main+0x29>                 | 0x119c
0x1193 | mov EAX, dword ptr [RBP-0x8]           |     mov EAX, dword ptr [RBP-0x8]          | 0x119e
0x1196 | leave                                  |     leave                                 | 0x11a1
0x1197 | ret                                    |     ret                                   | 0x11a2
0x1198 | nop dword ptr [RAX+RAX+0x0]      [-]   | [+] nop word ptr [CS:RAX+RAX+0x0]         | 0x11a3
                                                | [+] nop dword ptr [RAX]                   | 0x11ad
```

Overview


100
001
Stripped Binaries
FunProbe
Function
Boundaries
B2R2
Binaries Information
Sections
.text
.plt
.rodata
Function
Boundaries
Library
Functions
Strings
Addresses
Instructions
QuickBinDiff
Display

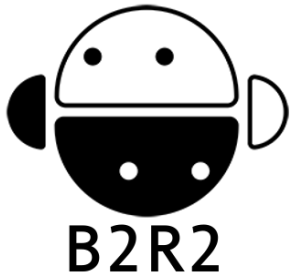
Discover Function Boundary

- A tool for recovering user-defined functions from stripped binaries
- Compiler-agnostic (including GCC, LLVM)
- Architecture-agnostic (including x86, x64, ARM, MIPS)
- A state-of-the-art tool proposed in 2023
  - higher accuracy than existing tools (97% for MIPS-clang, 99% for others)
  - 6 times faster on average (6 seconds for GNU coreutils binary)



Disassemble and Gather Data

- Binary Analysis tool
- Free, open-source, CLI-based
- Cross platform (Linux, Windows, OS X)
- Various file format (including ELF, PE, Mach-O)
- Architecture-agnostic (including x86, x64, ARM, MIPS)
- Providing various information necessary for diff



Binary Diffing

- Binary, section, function, basic block or snippet sliced by address
- Gathering instructions and breaking them down into finer granularity such as opcode and operands
- There are several issues that need to be resolved

Binary Diffing – Resolve Expected Issues

- Separating addresses and instructions
  - The addresses should be included in the diff result, but including them in the diff introduces many errors.

```
──────────────────────────────────────────────────────────────────────────────────────────────────────────
                                        ..skip..
0x1176 | mov EDI, 0x5                        |     mov EDI, 0x5                        | 0x117a
0x117b | call <func1> ; -0x43                |     call <func1> ; -0x47                | 0x117f
0x1180 | mov dword ptr [RBP-0x4], EAX        |     mov dword ptr [RBP-0x4], EAX        | 0x1184
       |                                     | [+] mov dword ptr [RBP-0x8], 0x0        | 0x1187
0x1183 | jmp +0xa ; <main+0x2a>              |     jmp +0xa ; <main+0x2a>              | 0x118e
0x1185 | add dword ptr [RBP-0x8], 0x1  [-] |                                         |
0x1189 | sub dword ptr [RBP-0x4], 0x1        |     sub dword ptr [RBP-0x4], 0x1        | 0x1190
       |                                     | [+] add dword ptr [RBP-0x8], 0x1        | 0x1194
                                        ..skip..
──────────────────────────────────────────────────────────────────────────────────────────────────────────
```

Binary Diffing – Resolve Expected Issues

- Resolve function address
  - Function addresses can also vary due to “address shift”
  - They may point to semantically equivalent functions but with different syntax

```
──────────────────────────────────────────────────────────────────────────────────────────────────────
                                        ..skip..
0x1176 | mov EDI, 0x5                        |     mov EDI, 0x5                          | 0x117a
0x117b | call <func1> ; -0x43               |     call <func1> ; -0x47                  | 0x117f
0x1180 | mov dword ptr [RBP-0x4], EAX       |     mov dword ptr [RBP-0x4], EAX          | 0x1184
       |                                    | [+] mov dword ptr [RBP-0x8], 0x0          | 0x1187
0x1183 | jmp +0xa ; <main+0x2a>             |     jmp +0xa ; <main+0x2a>                | 0x118e
                                        ..skip..
──────────────────────────────────────────────────────────────────────────────────────────────────────
```

Binary Diffing – Resolve Expected Issues

- Resolve address with function and offset
  - Even if the target addresses differ, consider them semantically equivalent if the offset of the operand, such as +0xa, is the same relative to the function address, like main+0x2a."

```
──────────────────────────────────────────────────────────────────────────────────────────────────
                                    ..skip..
0x1176 | mov EDI, 0x5                    |     mov EDI, 0x5                        | 0x117a
0x117b | call <func1> ; -0x43            |     call <func1> ; -0x47                | 0x117f
0x1180 | mov dword ptr [RBP-0x4], EAX    |     mov dword ptr [RBP-0x4], EAX        | 0x1184
       |                                 | [+] mov dword ptr [RBP-0x8], 0x0        | 0x1187
0x1183 | jmp +0xa ; <main+0x2a>          |     jmp +0xa ; <main+0x2a>              | 0x118e
                                    ..skip..
──────────────────────────────────────────────────────────────────────────────────────────────────
```

Binary Diffing – Resolve Expected Issues

- Resolve address with function and offset
  - Both appear as "jnz -0xc," indicating identical syntax

```
──────────────────────────────────────────────────────────────────────────────────────────────────────────
                                        ..skip..
0x118d | cmp dword ptr [RBP-0x4], 0x0       |     cmp dword ptr [RBP-0x4], 0x0     | 0x1198
0x1191 | jnz -0xc ; <main+0x22>        [-] | [+] jz -0xc ; <main+0x29>            | 0x119c
0x1193 | mov EAX, dword ptr [RBP-0x8]      |     mov EAX, dword ptr [RBP-0x8]     | 0x119e
0x1196 | leave                             |     leave                            | 0x11a1
0x1197 | ret                               |     ret                              | 0x11a2
──────────────────────────────────────────────────────────────────────────────────────────────────────────
```

Binary Diffing – Resolve Additional Issues

- Resolve additional issues

- Instruction pointing string in .rodata section
  - same address points to different strings
  - different addresses point to the same string

Display Result

- Section, function, basic block or code snippet sliced by address
- Side-by-side view
- Red, green, background-highlight
  - Fine-grained diff in are highlighted with background color replaced lines
- Explicit mark (`[+]`, `[-]`)

```
                                        ..skip..
0x118d | cmp dword ptr [RBP-0x4], 0x0       |     cmp dword ptr [RBP-0x4], 0x0     | 0x1198
0x1191 | jnz -0xc ; <main+0x22>         [-] | [+] jz -0xc ; <main+0x29>            | 0x119c
0x1193 | mov EAX, dword ptr [RBP-0x8]      |     mov EAX, dword ptr [RBP-0x8]     | 0x119e
0x1196 | leave                             |     leave                            | 0x11a1
0x1197 | ret                               |     ret                              | 0x11a2
```

Prototype (Demo)

- Implement diff algorithm and separate address and instruction

```
    (.text)

    <_start>
    0000000000001040: endbr64
    0000000000001044: xor EBP, EBP
    0000000000001046: mov R9, RDX
    0000000000001049: pop RSI
    000000000000104a: mov RDX, RSP
    000000000000104d: and RSP, 0xfffffffffffffff0
    0000000000001051: push RAX
    0000000000001052: push RSP
[-] 0000000000001053: lea R8, qword ptr [RIP+0x1b6]
[-] 000000000000105a: lea RCX, qword ptr [RIP+0x13f]
[-] 0000000000001061: lea RDI, qword ptr [RIP+0xfb]
    0000000000001068: call qword ptr [RIP+0x2f72]
    000000000000106e: hlt
    000000000000106f: nop
```

```
    (.text)

    <_start>
    0000000000001040: endbr64
    0000000000001044: xor EBP, EBP
    0000000000001046: mov R9, RDX
    0000000000001049: pop RSI
    000000000000104a: mov RDX, RSP
    000000000000104d: and RSP, 0xfffffffffffffff0
    0000000000001051: push RAX
    0000000000001052: push RSP
[+] 0000000000001053: lea R8, qword ptr [RIP+0x1c6]
[+] 000000000000105a: lea RCX, qword ptr [RIP+0x14f]
[+] 0000000000001061: lea RDI, qword ptr [RIP+0xff]
    0000000000001068: call qword ptr [RIP+0x2f72]
    000000000000106e: hlt
    000000000000106f: nop
```

A Study on Readability

- Implement the well-known four diff algorithms
- Resolve issues to maintain semantics
- Analyze various cases using a sample dataset
- (Empirical) Identify cases where readability differs and conduct a case study
- (Theoretical) Analyze the phenomenon based on the operational principles of the algorithm.
- Select the best diff algorithm for readability

Researcher's Capabilities

- Proficient in diff algorithms
  - Implement the well-known four diff algorithms
  - I have discovered that the algorithm's name and its implementation in git diff have not been consistent since 2005
    - The names and implementations of Myers and Minimal have been swapped
  - Microsoft and Git's Developers, contributors, and users are all unaware this truth
- Proficient in the F# language
  - I have a basic understanding of the B2R2
  - If I implement a diff tool, migration can be done easily.
- Large dataset for research
  - 6 architectures, 20 compilers (gcc-4~11, clang3-13), and 6 optimization levels (O0-O3, Os, Ofast)
  - I have compiled 200 unique source codes, resulting in 200K binaries
  - It is easy to construct ground truth for the diff tool and there is enough data to conduct empirical research on readability

Summary

- Binary diff is a technology required in various fields.
  - The absence of a free, open-source, CLI-based diff tools focused on assembly code diff
  - Difficult to automatically handle a large quantity of binaries
  - The lack of research on the readability of binary code diff algorithms
- Fast, free, open sourced and CLI-based binary code diff tool
  - Support cross-platform, various file foramt and multi architecture