

万众一芯黑客马拉松

解题技术分享

参赛选手：芮志清 刘通 石珺锶 吕浩

指导老师：凌祥 吴敬征

中国科学院软件研究所 HASSLab 战队

2025.12.07

赛题简介

黑客马拉松2512



本次活动提供了15个人工注入的Bug供大家进行发现和分析，共有两个赛道：找Bug赛道和Token效率赛道。

- **Bug分级：** 每个模块含有注入5个bug，分别对应5个RTL原文件，简单难度2个、中等难度1个、困难难度2个。
- **Bug积分：** 简单难度 100分/个，中等难度300分/个，困难500分/个。
- **Token效率：** 效率 $E = \text{Bug个数} / \text{消耗总 Token}$

赛道介绍

- **找Bug赛道：** 对参赛队伍按获得的Bug积分进行排名 (bug数需要大于5个)，前三名将获得证书 + 现金奖励，其他名次将获得证书。
- **效率赛道：** 利用UCAgent API模式分析Bug，按Token效率进行排名(bug数需不少于12个)，前三名将获得证书 + 现金奖励，其他名次将获得证书。
- **额外奖励：** 首位发现Origin版本中的非人工注入Bug，获得现金奖励：500-1000元/个 + 证书。具体金额由UT设计者给出的Bug等级来定。

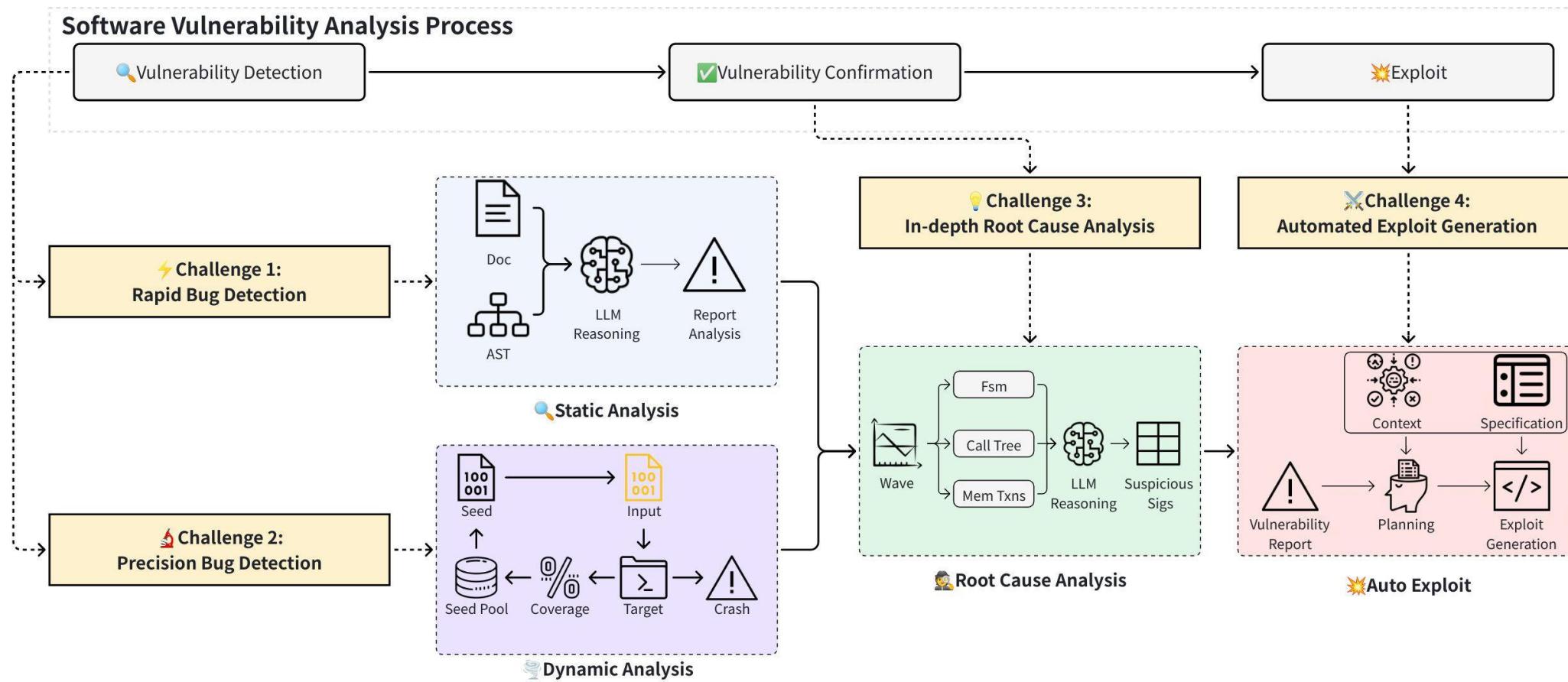
注：除了效率赛道外，其他赛道可用您任何擅长的工具或者方法，不仅限于UCAgent技术路线。

开源地址：<https://github.com/XS-MLVP/hackathon2512>

方法论

■ 基于软件漏洞挖掘中的思路

- 静态审计代码找到潜在BUG点，深入分析原因以确认bug是否存在，最终实现动态利用



Methodology Framework

静态审计

- 赛题提供的代码经过混淆，肉眼阅读困难
- 大模型针对变量名混淆已有较强的分析能力，能够找到大多数BUG
- 对于难以找到的BUG，对照原始代码和主办方的提示进行更细节的分析

```
''
SRT16Div_t dwa0umq6VFkny0F (
    .clock      (clock),
    .reset      (reset),
    .io_sign    (UshydUvI),
    .io_dividend
        (daRDdBa6jKZprCqTAnGiB
         ? {24'h0, Z99bC[103:96]}
         : Rm8i3Kn9BlG8bFJV6zZB1h6 ? {16'h0, Z99bC[79:64]} : Z99bC[31:0]),
    .io_divisor
        (daRDdBa6jKZprCqTAnGiB
         ? {24'h0, kzZJI[103:96]}
         : Rm8i3Kn9BlG8bFJV6zZB1h6 ? {16'h0, kzZJI[79:64]} : kzZJI[31:0]),
    .io_flush    (io_flush),
    .io_sew     (iK6868r),
    .io_div_in_valid (nJwf7EZY[1]),
    .io_div_out_ready (nJwf7EZY[2]),
    .io_div_out_valid (fxJ6zKRfqCD4Mu0t7NggB8hLfMcVxnRbg),
    .io_div_out_q   (e8bQQDF7jurQZfpsS5EhddkP1s11H),
    .io_div_out_rem (xpuEsAqUebvef5zMpjWREhONe1Bj6Ze)
);

```

被混淆的代码

这段被混淆后的代码被人工注入了一个隐藏bug，请找出这个bug。

显示思路 ▾

经过静态代码分析，在 `VectorIdiv` 模块及其子模块中发现了一个严重的**异常标志位屏蔽 (Exception Flag Suppression) **类型的 Bug。

1. 缺陷实证与代码定位

经查证源代码，该缺陷位于 `SRT4qdsCons` 模块的 `dQTN` 信号定义处。该信号用于定义 SRT4 除法算法中商位选择函数 (QDS) 的正向选择阈值。

Bug 位置与详情

- 所在模块: `VectorIdiv` (Top Level)
- 源代码行号:
- Bug 类型: 标志位/异常处理削弱 / 硬编码逻辑

问题代码片段:

Verilog

```
wire [7:0][8:0] dQTN =
  '{9'h1A4, 9'h1AC, 9'h1B0, 9'h1B8, 9'h1BC, 9'h1C4, 9'h1D0, 9'h1D8};
```

Verilog

```
assign io_d_zero = 16'h0; // 在bug版本中被强制移除,原版本 ass
```

特征分析: 数组最后两项均为 `9'h1D0`。在标准的 SRT4 算法实现中，随着截断除数 (truncated divisor) 索引值的增加，选择常量应呈单调递增趋势以匹配 P-D 图 (Robertson

基于大模型找到的BUG

原理分析

- 由于仅靠代码分析无法准确得到BUG原理，因此通过文档分析获得Spec，用于指导测例生成
- 首先找到相关Spec
 - 基于文档分析，找到相关的文档和出处
 - 对于文档中不存在的逻辑，直接给出相关算法逻辑作为技术参考



向量整数除法有符号数处理逻辑原始规格说明

本文档专门针对 **向量整数除法 (Vector Integer Division)** 中有符号数 (Signed Integer) 的处理逻辑进行规范溯源分析。依据 RISC-V 标准，有符号除法在处理除零和溢出时的行为有特定规定。

1. 向量扩展规范 (Vector Spec)

向量规范明确声明向量整数除法指令 (vdiv.vv, vdiv.vx 等) 的行为等同于标量整数除法。

- 来源文档: riscv-v-spec-1.0.pdf
- 章节: 11. Vector Integer Arithmetic Instructions
- 小节: 11.11. Vector Integer Divide Instructions
- 关键原文: "The divide and remainder instructions are equivalent to the RISC-V standard scalar integer multiply/divides, with the same results for extreme inputs."

译文: 除法和取余指令等同于 RISC-V 标准标量整数乘/除法，对于极端输入（如除零、溢出）具有相同的结果。

分析: 这意味着我们需要参考标量规范来确定有符号除法的具体行为，特别是针对有符号数的溢出情况。

基于规范文档获得Spec

SRT-4 除法算法设计与实现逻辑

1. 概述

SRT (Sweeney, Robertson, and Tocher) 除法算法属于数字递归 (Digit Recurrence) 类算法，广泛应用于高性能浮点运算单元 (FPU) 中。SRT-4表示该算法采用基数4 (Radix-4)，即每次迭代生成2位商。相较于传统的恢复余数法 (Restoring) 或不恢复余数法 (Non-Restoring)，SRT算法的核心优势在于引入了冗余符号位集合 (**Redundant Signed-Digit, RSD**)，使得商位选择过程不再依赖于全精度的部分余数比较，从而显著缩短了关键路径延迟。

2. 数学原理

2.1 递归公式

SRT-4除法的核心迭代公式如下：

$$\$\$P_{j+1} = r \cdot P_j - q_{j+1} \cdot D \$\$$$

其中：

- P_j : 第 j 次迭代的部分余数 (Partial Remainder)。
- P_0 : 初始部分余数，即被除数。
- r : 基数，对于SRT-4， $r = 4$ 。
- $q_j + 1$: 第 $j + 1$ 次选取的商位。
- D : 除数。

基于模型知识获得Spec

原理分析

■ 进一步分析BUG原理及影响

⚠ 缺陷影响分析

1. 直接后果

商位选择偏差：

- 当部分余数落在 456 到 464 之间时，原本应选择 +2 的情况被错误判断
- 导致特定除数模式下的商值计算错误

2. 行为异常示例

测试场景：特定除数模式下的除法运算

输入除数模式	预期商位选择	实际 BUG 输出	差异
特定高位模式(索引 6)	+2	+1 或其他值	商值错误 ✕

典型错误链：

```

部分余数 = 460 (在456-464区间)
↓
正确判断: quotient_digit = +2
↓
错误判断: quotient_digit = +1 (由于阈值被提高)
↓
后续迭代基于错误商值计算
↓
最终商和余数均错误

```

🔍 信号溯源分析

信号定义链路追踪

dQTN (即 _GEN) 的完整使用路径：

```

// 第162行 - 查找表定义
wire [7:0][8:0] dQTN = '{...};

// 第171行 - 信号使用
assign io_m_pos_2 = dQTN[io_d_trunc_3];

// 上游模块通过 io_d_trunc_3 索引选择对应阈值

```

依赖信号解析：

信号名	含义	数据来源
io_d_trunc_3	除数高位截断值(3 位)	上游 SRT 算法模块
io_m_pos_2	+2 商位选择阈值输出	当前模块输出

触发BUG

■ 基于BUG原理，首先构思测试用例实现思路

2. 构建测试用例 (Test Case Construction)

为了验证此 Bug, 我们需要构造特定的 **被除数 (Dividend)** 和 **除数 (Divisor)**,
使其满足以下两个条件:

条件 A: 命中特定的除数区间

查找表索引为 6, 对应 `io_d_trunc_3` 为 `3'b110`。

在 `SRT16Divint` 中, `io_d_trunc_3` 连接自 `norm_d[14:12]`。

由于 `norm_d` 是归一化的, 最高有效位 (第15位) 总是1。

因此, 我们需要构造一个除数, 其二进制形式高位模式为 `1.110...`。

- 对于 16位 整数除法:

除数应在 `0xE000` 到 `0xFFFF` 之间 (或其他移位后能归一化到此范围的数)。

- 例如: `16'hE000` (二进制 `1110 0000...`)。

条件 B: 命中特定的部分余数区间

我们需要在除法的某一次迭代中, 使得移位后的部分余数 (High part of P-D plot) 落在 `[0x1C8, 0x1D0)` 也就是 `[456, 464)` 之间。

这通常发生在部分余数大约是除数的 1.75 到 1.8 倍附近。

触发BUG

基于UCAgent搭建测试环境，构建基础测试用例

Mission

VectorIdiv芯片验证任务

```

0 1-requirement_analysis_and_planning-需求分析与验证规划 (0 fails, 16m 07s)
1 2-dut_function_understanding-VectorIdiv功能理解 (0 fails, 02m 02s)
2 3.1-dut_function_grouping-DUT功能分组与层次划分[7] (0 fails, 02m 31s)
3 3.2-function_point_definition-具体功能点识别与定义[26] (0 fails, 02m 17s)
4 3.3-check_point_design-检测点设计与定义[131] (0 fails, 03m 21s)
5 3-functional_specification_analysis-功能规格分析与测试点定义 (0 fails, 24s)
6 4.1-dut_creation_implementation-DUT创建函数实现 (0 fails, 02m 21s)
7 4.2-pytest_fixture_dut_implementation-dut fixture实现 (0 fails, 49s)
8 5.1-coverage_group_creation-功能覆盖组创建 (0 fails, 01m 09s)
9 5.2.1-implemente_function_checks_in_batch-分批功能点检查函数实现 [131/131] (6 fails, 12m 30s)
10 5.2-coverage_point_implementation-覆盖率检查点实现 (0 fails, 22s)
11 *6.1-human_check_env_specification-人工检查env规格说明 (skipped)
12 6.2-bundle_wrapper_design-引脚封装设计 (0 fails, 01m 31s)
13 6.3-mock_design_and_implementation-Mock组件设计与实现 (skipped)
14 6.4-env_fixture_implementation-env fixture实现 (0 fails, 01m 05s)
15 6.5-evaluate_env_fixture-env fixture实现测试与评估 (1 fails, 03m 33s)
16 *6.6-human_check_env_implementation-人工检查env实现质量 (skipped)
17 7-basic_apiImplementation-基础API实现 (0 fails, 01m 56s)
18 8-basic_api_functional_test-基础API功能正确性测试 (3 fails, 32m 06s)
19 9-create_test_case_templates-创建测试用例模板 [131/131,122] (5 fails, 15m 06s)
20 10.1-test_case_implementation_in_batch-分批测试用例实现与对应bug分析 [110/122] (12 fails, 1h 07m 41s)
21 10-comprehensive_verification_and_bug_analysis-全面验证与缺陷分析 (0 fails)
22 11-line_coverage_analysis_and_improvement-代码行覆盖率分析与提升(90.00) (skipped)
23 12-verification_review_and_summary-验证审查与总结 (0 fails)

Changed Files
2025-12-06 12:09:37 (00:00:08): .pytest_cache/v/cache/nodeids

```

Welcome back!

Tips for getting started
Run ./init to create a CLAUDE.md file with instructions for Claude

Recent activity
No recent activity

Sonnet 4.5 - API Usage Billing
~/Work/2512-hackathon/hackathon2512/output/5011

/model to try Opus 4.5

Conversation compacted · ctrl+o for history

- I'll continue by running the 10 tests from Batch 4 that were just implemented.
- ucagent-div - RunTestCases (MCP)**:target: "test_VectorIdiv_error_and_flow_control.py::test_flush_then_new_operation test_VectorIdiv_error_and_flow_control.py::test_continuous_flush test_VectorIdiv_error_and_flow_control.py::test_signed_overflow_16 test_VectorIdiv_error_and_flow_control.py::test_signed_overflow_132 test_VectorIdiv_error_and_flow_control.py::test_signed_overflow_164 test_VectorIdiv_error_and_flow_control.py::test_signed_overflow_output test_VectorIdiv_error_and_flow_control.py::test_flush_during_calculation test_VectorIdiv_signed_division.py::test_signed_div_s16_pos_div_pos test_VectorIdiv_signed_division.py::test_signed_div_s16_neg_div_pos", timeout: 120
- REPORT:
run_test_success: true
tests:
+141 lines (ctrl+o to expand)
- Excellent! Batch 4 tests all passed (10/10). Let me check the overall progress now.
- ucagent-div - Check (MCP)**:timeout: 60
- check_info:
- name: UnityChipCheckerBatchTestsImplementation
count_pass: 0
+12 lines (ctrl+o to expand)
- Progress: 40/122 tests complete (32.8%). Moving to Batch 5 - implementing 10 more signed division tests.

Name: UCAgent-25.11.22.dev25+ga8fe443da
Email: unitychip@bosc.ac.cn

VectorIdiv Test Report

Started 2025-12-06 12:56:19
Ended 2025-12-06 12:56:49
Duration 0:00:30.152576
Total run time 0:00:30.063872

139 passed

Line Coverage				Functional Coverage			
Coverage Rate	Hint Lines	Total Lines	Detail	Coverage Rate	Hint Points	Total Points	Marked Points
100.0%	2066	2066	View Details	100.0%	26	26	26/26 (100.0%) View Details

WARNING <frozen importlib._bootstrap>:488
builtin type SwigPyPacked has no __module__ attribute
WARNING <frozen importlib._bootstrap>:488
builtin type SwigPyObject has no __module__ attribute
WARNING <frozen importlib._bootstrap>:488
builtin type SwigPyPacked has no __module__ attribute
WARNING <frozen importlib._bootstrap>:488
builtin type SwigPyObject has no __module__ attribute
WARNING <frozen importlib._bootstrap>:488
builtin type swigvarlink has no __module__ attribute

Marked Functions: 139/139 (100.0%)

> tests/test_VectorIdiv_api_basic.py	PASSED	17	3
> test_api_VectorIdiv_reset_basic	PASSED	1	
> test_api_VectorIdiv_reset_custom_cycles	PASSED	1	
> test_api_VectorIdiv_reset_parameter_validation	PASSED	1	
> test_api_VectorIdiv_reset_outputs	PASSED	1	
> test_api_VectorIdiv_divide_basic_unsigned	PASSED	1	

0:00:03.680559
0:00:00.280137
0:00:00.215672
0:00:00.207146
0:00:00.202520
0:00:00.207049

触发BUG

- 然后参考已有基础测试用例，构建针对对应 BUG 的测试用例
- 在 bug 版本和 origin 版本上分别测试，根据运行结果调试测试用例代码，以触发 BUG

1. 整体架构

测试脚本采用 约束随机验证 (Constrained Random Verification) 方法：

```
iterations = 50000 # 大量迭代以覆盖边界情况
for i in range(iterations):
    # 生成测试向量
    # 驱动硬件
    # 检查结果
```

2. 关键设计：有偏随机数生成

2.1 除数生成策略

测试脚本的核心在于有偏的除数生成，专门针对触发 Bug 的条件：

```
r = random.random()
if r < 0.5: # 50% 概率
    # 目标 Index 6: MSB 4 bits = 1110 (0xE)
    base = random.getrandbits(60)
    divisor = (0xE << 60) | base
elif r < 0.8: # 30% 概率
    # 随机 64-bit
    divisor = random.getrandbits(64)
else: # 20% 概率
    # 小数字或其他模式
    divisor = random.randint(1, 100000)
```

```
cd 2_test_cases
python bug/VectorIdiv/example.py
echo "----"
python origin/VectorIdiv/example.py
```

- (py3) → [VectorIdiv_bug_2](#) cd 2_test_cases
- (py3) → [2_test_cases ls](#) [bug](#) [origin](#) 运行方式及
- ② (py3) → [2_test_cases python bug/VectorIdiv/example.py](#)
Ready after reset: 1
Starting verification loop...
Mismatch at iteration 308
Dividend: 17360576827843868672 (0xf0ed28a50782f800)
Divisor: 40627 (0x9eb3)
Exp Q: 427316238655176 (0x184a4509c9ec8)
Act Q: 427316401036592 (0x184a45a4a5d30)
Exp R: 33320 (0x8228)
Act R: 12144 (0x2f70)
- (py3) → [2_test_cases echo "----"](#)
- (py3) → [2_test_cases python origin/VectorIdiv/example.py](#)
Ready after reset: 1
Starting verification loop...
Passed 50000 iterations.

未解决的问题

- 人机协同增强：UCAgent 框架的设计理念和使用细节还未充分掌握，因此调试跑通整个流程较为困难，经常遇到失败的情况。如何更好地和大模型人机协同，使其能准确生成 test env 和 test case？
- 资源开销的平衡：性能更强的大模型能做到所有流程一遍通过，实现更少的交互调试，生成更准确的测试用例，但价格开销较高；性能一般的大模型需要更多的交互调试，耗时耗力。
- 设计文档补全：单一渠道无法收集到足够全面的设计文档，要求验证人员对于模块设计有深入理解。

总结

- 找到了并验证了所有的共15个BUG，获得4500分
- 学习到很多关于香山、RTL验证、MCP、Agent的知识，极大扩展了知识面
- 感谢万众一芯黑客马拉松主办方的精心准备，感谢队友们的通力合作！

DOMjudge Home Problemset Scoreboard Enable Notifications Submit Logout 01:36

万众一芯-黑客马拉松(第一期) started: 09:00 - ends: 15:30

Filter ▾

RANK	TEAM	SCORE	ADDER_BUG1	ADDER_BUG2	ADDER_BUG3	ADDER_BUG4	ADDER_BUG5	ADDER_RAW	FMA_Bug1	FMA_Bug2	FMA_Bug3	FMA_Bug4	FMA_Bug5
1	2512-team-HASSLab	4500	100 1 try	100 1 try	300 1 try	500 1 try	500 1 try		100 1 try	100 1 try	300 1 try	500 1 try	500 1 try
2	2512-individual-高明	1400	100 1 try		300 1 try	500 1 try			100 1 try	100 1 try			
3	2512-individual-小金	500	100 1 try	100 2 tries	300 2 tries		1 try	0 + 1 tries					
4	2512-team-香山真香	300		1 try					1 try				
5	2512-team-智验芯小队	100											
6	2512-individual-任嘉楠	100		2 tries									
7	2512-individual-不报错也是一种报错	0											
	2512-individual-任冬冬	0											
	2512-individual-余桂强	0											
	2512-individual-刘子宸	0											
	2512-individual-刘昕	0											
	2512-individual-刘馨鸿	0											
	2512-individual-卢超	0											
	2512-individual-史塔克	0											

200 @team_scoreboard 118 ms 8.0 MiB ↓ 20 ⌂ 1 ⌂ 25 in 0.44 ms ⌂ 13569315527 ⌂ 20 ms ⌂ 20 in 16.80 ms ⌂ S:0 D:0 6.4.23 ×

谢谢！

汇报人: 范志清
zhiqing@iscas.ac.cn



微信号: m2kar-cn