永信至诚 春秋 GAME

WRITEUP 文档规范

WRITEUP 书写规范

一、定义：

书面解题思路（WRITEUP，简称“WP”）是参赛选手将自己解题的思路，包括猜想、实践过程，以及必要的工具、方法、代码、资源等用书面的形式记录下来。供其他参赛选手学习和分享，以及供裁判组审查对这道题原创的解决能力。

二、特点：

1. 书面性：

要求以文档的形式进行记录：包括 word、markdown 或其他文档工具生成的可阅读的 pdf 格式。如果有更方便理解的需要，可以辅助以图片、语音及录像等形式。但是书面的文档内容必须记录完整的解题思路及过程的必备要素。

2. 完整性：

WP 文档必须完整的包含参赛战队队名、战队排名、战队整体答题情况，每道题的答题人，以及对每道成功解决（成功得分）的题目的分析和破解的过程，如果 WP 中有关键步骤缺失导致无法复现解题经

过，则视为 WP 不完整。

3. 原创性：

WP 文档需要能体现参赛选手所在团队有独立解决该问题的能力。如果在解决问题的过程中，用到了某种工具，需要注明工具的来源，只能是来源可追的或者自研工具。如果是自研工具，需要附上对解决问题有帮助的代码。

4. 可读性：

WP 文档需要语句通顺，逻辑严谨，格式及排版规范，可以通过 WP 逐步推导出来对问题的分析以及对正确答案的获取。

三、注意事项

1. 书面解题文档需要包含战队的队伍信息，解题列表，以及战队在本场比赛中所解决的全部题目的解题步骤和思考过程。
2. WP 文件名请包含战队队名。
3. 解题过程中，关键步骤不可省略，不可含糊其辞、一笔带过。
4. 解题过程中如是自己编写的脚本，不可省略，不可截图（代码字体可以调小；而如果代码太长，则贴关键代码函数）。
5. 您队伍所有解出的题目都必须书写 WRITEUP，缺少一个则视该 WRITEUP 无效，队伍成绩将无效。
6. WRITEUP 如过于简略和敷衍，导致无法形成逻辑链条推断出战队

对题目有分析和解决的能力，该 WRITEUP 可能被视为无效，队伍成绩将无效。

7. WRITEUP 书写过程中请注意格式规范，排版干净，语句通顺，以及用语文明。如果影响阅读可能会被判为无效。
8. WRITEUP 请务必按时提交，平台将在规定时间后停止收集 WP。

附件：WRITEUP 模板

码农烧烤战队 WRITEUP

一、 战队信息

战队名称：码农烧烤

战队排名：42

二、 解题情况

请粘贴战队排名截图和答题情况截图：

示例的操作流程：

“排行榜”→ “输入框输入自己对队伍名称”→“找到您所在队伍”→“截图”

（提交的时候请把下图替换为您队伍排行榜上的排名截图）

三、 解题过程

题目序号 题目名称

（题目序号 请参考解题总榜上面的序号）

Reverse—VVM

签到

操作内容：

（请输入操作内容）

如该题使用自己编写的脚本代码请详细写出，不允许截图

签到题：

```
s = 'zhofrph_fkdqjfkhqjehl'
for c in s:
    print(chr(ord(c) - 3), end = '')
```

VVM：

```
/* This file was generated by the Hex-Rays decompiler version 8.3.0.230608.
   Copyright (c) 2007-2021 Hex-Rays <info@hex-rays.com>

   Detected compiler: GNU C++
*/

// #include <defs.h>
#include <cstring>
#include <cstdio>

// using __int64 = long long;
using _DWORD = unsigned int;

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

// void sub_1020();
// void sub_1030();
// void sub_1040();
```

```
// void sub_1050();
// void sub_1060();
// void sub_1070();
// int __fastcall _cxa_finalize(void *);
// size_t strlen(const char *s);
// int printf(const char *format, ...);
// void *memset(void *s, int c, size_t n);
// __int64 __isoc99_scanf(const char *, ...); weak
// void __noreturn exit(int status);
void *sub_1110();
__int64 sub_1140();
void *sub_1180();
__int64 sub_11C0();
__int64 __fastcall sub_11C9(unsigned int a1, int a2);
__int64 sub_11F8();
_DWORD *__fastcall sub_1217(_DWORD *a1, int a2);
_DWORD *__fastcall sub_1232(_DWORD *a1, _DWORD *a2);
_DWORD *__fastcall sub_1251(_DWORD *a1, int a2);
int *__fastcall sub_1274(int *a1);
__int64 __fastcall sub_1297(int *a1);
_DWORD *__fastcall sub_12B2(_DWORD *a1);
__int64 __fastcall sub_12CD(__int64 a1);
int __fastcall main(int a1, char **a2, char **a3); // idb
void fini(void); // idb
void term_proc();
// int __fastcall _libc_start_main(int (__fastcall *main)(int, char **, char **), int argc, char **ubp_av,
void (*init)(void), void (*fini)(void), void (*rtld_fini)(void), void *stack_end);
// int __fastcall __cxa_finalize(void *);
// __int64 _gmon_start__(void); weak

//-------------------------------------------------------------------------
// Data declarations

// _UNKNOWN init;
// __int64 (__fastcall *off_3D98)() = &sub_11C0; // weak
// __int64 (__fastcall *off_3DA0)() = &sub_1180; // weak
void *off_4008 = &off_4008; // idb
_DWORD dword_4020[24] =
{
  126,
  120,
  117,
  127,
  107,
```

```
    82,
    117,
    114,
    109,
    119,
    78,
    121,
    121,
    121,
    119,
    68,
    98,
    36,
    96,
    113,
    115,
    96,
    53,
    105
}; // weak
_DWORD dword_4080[9] = { 204, 10, 11, 12, 13, 14, 15, 170, 187 }; // weak
// _UNKNOWN unk_40A8; // weak
char byte_40B0; // weak
_DWORD unk_40B4; // weak
_DWORD dword_40BC; // weak
int dword_40C0; // weak
int dword_40C4; // weak
int dword_40C8; // weak
char s[24]; // idb


//----- (0000000000001000) ----------------------------------------------------
// __int64 (**init_proc())(void)
// {
//   __int64 (**result)(void); // rax

//   result = &_gmon_start__;
//   if ( &_gmon_start__ )
//     return (__int64 (**)(void))_gmon_start__();
//   return result;
// }
// 4128: using guessed type __int64 _gmon_start__(void);

//----- (0000000000001020) ----------------------------------------------------
```

```
// void sub_1020()
// {
//     JUMPOUT(0LL);
// }
// 1026: control flows out of bounds to 0

// //----- (0000000000001030) ----------------------------------------------------
// void sub_1030()
// {
//     sub_1020();
// }

// //----- (0000000000001040) ----------------------------------------------------
// void sub_1040()
// {
//     sub_1020();
// }

// //----- (0000000000001050) ----------------------------------------------------
// void sub_1050()
// {
//     sub_1020();
// }

// //----- (0000000000001060) ----------------------------------------------------
// void sub_1060()
// {
//     sub_1020();
// }

// //----- (0000000000001070) ----------------------------------------------------
// void sub_1070()
// {
//     sub_1020();
// }

//----- (00000000000010E0) ----------------------------------------------------
// positive sp value has been detected, the output may be wrong!
// void __fastcall __noreturn start(__int64 a1, __int64 a2, void (*a3)(void))
// {
//     __int64 v3; // rax
//     int v4; // esi
//     __int64 v5; // [rsp-8h] [rbp-8h] BYREF
//     char *retaddr; // [rsp+0h] [rbp+0h] BYREF
```

```
//     v4 = v5;
//     v5 = v3;
//     _libc_start_main(main, v4, &retaddr, (void (*)(void))init, fini, a3, &v5);
//     __halt();
// }
// 10EA: positive sp value 8 has been found
// 10F1: variable 'v3' is possibly undefined

//----- (0000000000001110) ----------------------------------------------------
// void *sub_1110()
// {
//     return &unk_40A8;
// }

//----- (0000000000001140) ----------------------------------------------------
__int64 sub_1140()
{
  return 0LL;
}

//----- (0000000000001180) ----------------------------------------------------
// void *sub_1180()
// {
//     void *result; // rax

//     if ( !byte_40B0 )
//     {
//         if ( &__cxa_finalize )
//             _cxa_finalize(off_4008);
//         result = sub_1110();
//         byte_40B0 = 1;
//     }
//     return result;
// }
// 40B0: using guessed type char byte_40B0;

//----- (00000000000011C0) ----------------------------------------------------
// attributes: thunk
__int64 sub_11C0()
{
  return sub_1140();
}
```

```
//----- (00000000000011C9) ----------------------------------------------------
__int64 __fastcall sub_11C9(unsigned int a1, int a2)
{
  __int64 result; // rax

  result = a1;
  dword_40C4 = a1 == a2;
  return result;
}
// 40C4: using guessed type int dword_40C4;

//----- (00000000000011F8) ----------------------------------------------------
__int64 sub_11F8()
{
  __int64 result; // rax

  result = (unsigned int)dword_40C4;
  if ( !dword_40C4 )
    dword_40C0 = 0;
  return result;
}
// 40C0: using guessed type int dword_40C0;
// 40C4: using guessed type int dword_40C4;

//----- (0000000000001217) ----------------------------------------------------
_DWORD *__fastcall sub_1217(_DWORD *a1, int a2)
{
  _DWORD *result; // rax

  result = a1;
  *a1 = a2;
  return result;
}

//----- (0000000000001232) ----------------------------------------------------
_DWORD *__fastcall sub_1232(_DWORD *a1, _DWORD *a2)
{
  _DWORD *result; // rax

  result = a1;
  *a1 = *a2;
  return result;
}
```

```
//----- (0000000000001251) ----------------------------------------------------
_DWORD *__fastcall sub_1251(_DWORD *a1, int a2)
{
  _DWORD *result; // rax

  result = a1;
  *a1 += a2;
  return result;
}

//----- (0000000000001274) ----------------------------------------------------
int *__fastcall sub_1274(int *a1)
{
  int *result; // rax

  result = a1;
  *a1 = (*a1 + 2) ^ 0x16;
  return result;
}

//----- (0000000000001297) ----------------------------------------------------
__int64 __fastcall sub_1297(int *a1)
{
  __int64 result; // rax

  result = (unsigned int)*a1;
  dword_40C8 = *a1;
  return result;
}
// 40C8: using guessed type int dword_40C8;

//----- (00000000000012B2) ----------------------------------------------------
_DWORD *__fastcall sub_12B2(_DWORD *a1)
{
  _DWORD *result; // rax

  result = a1;
  *a1 = dword_40C8;
  return result;
}
// 40C8: using guessed type int dword_40C8;

//----- (00000000000012CD) ----------------------------------------------------
__int64 __fastcall sub_12CD(__int64 a1)
```

```
{
  __int64 result; // rax
  int v2; // [rsp+14h] [rbp-4h]

  if ( dword_4080[dword_40C0] == 204 )
    sub_1217(&dword_40BC, 0);
  ++dword_40C0;
  do
  {
    v2 = dword_4080[dword_40C0];
    if ( v2 == 221 )
    {
      sub_1251(&unk_40B4, 1);
    }
    else if ( v2 <= 221 )
    {
      if ( v2 == 187 )
      {
        sub_11F8();
      }
      else if ( v2 <= 187 )
      {
        if ( v2 > 15 )
        {
          if ( v2 == 170 )
            sub_11C9(dword_40BC, 24);
        }
        else if ( v2 >= 10 )
        {
          switch ( v2 )
          {
            case 10:
              sub_1232(&unk_40B4, (_DWORD *)(dword_40BC + a1));
              break;
            case 11:
              sub_1297((int *)&unk_40B4);
              break;
            case 12:
              sub_1274(&dword_40C8);
              break;
            case 13:
              sub_12B2(&unk_40B4);
              break;
            case 14:
```

```
                    sub_1232((_DWORD *)(dword_40BC + a1), &unk_40B4);
                    break;
                case 15:
                    sub_1251(&dword_40BC, 1);
                    break;
                default:
                    break;
            }
        }
      }
    }
    result = (unsigned int)++dword_40C0;
  }
  while ( dword_40C0 <= 8 );
  return result;
}
// 4080: using guessed type _DWORD dword_4080[9];
// 40BC: using guessed type int dword_40BC;
// 40C0: using guessed type int dword_40C0;
// 40C8: using guessed type int dword_40C8;

//----- (00000000000014A2) ----------------------------------------------------
int __fastcall main(int a1, char **a2, char **a3)
{
  int i; // [rsp+8h] [rbp-8h]

  memset(s, 0, sizeof(s));
  printf("please input your flag:");
  scanf("%s", s);
  char ps[24];
  memcpy(ps, s, sizeof(s));
  if ( (unsigned int)strlen(s) != 24 )
  {
    printf("failed!");
    return 0;
  }
  sub_12CD((__int64)s);
  // for ( i = 0; i < 24; ++i )
  // {
  //     if ( dword_4020[i] != s[i] )
  //         return printf("failed!\n");
  // }
  printf("{");
  for (int i = 0; i < 24; ++i)
```

```
    {
        printf("%d: '%c',", (int)s[i], ps[i]);
    }
    printf("}\n");
    printf("success!");
    return 0;
}
// 10C0: using guessed type __int64 __isoc99_scanf(const char *, ...);
// 4020: using guessed type _DWORD dword_4020[24];

//----- (00000000000015A0) ----------------------------------------------------
// void __fastcall init(unsigned int a1, __int64 a2, __int64 a3)
// {
//     signed __int64 v4; // rbp
//     __int64 i; // rbx

//     init_proc();
//     v4 = &off_3DA0 - &off_3D98;
//     if ( v4 )
//     {
//         for ( i = 0LL; i != v4; ++i )
//             ((void (__fastcall *)(_QWORD, __int64, __int64))*(&off_3D98 + i))(a1, a2, a3);
//     }
// }
// 3D98: using guessed type __int64 (__fastcall *off_3D98)();
// 3DA0: using guessed type __int64 (__fastcall *off_3DA0)();

//----- (0000000000001610) ----------------------------------------------------
void fini(void)
{
    ;
}

//----- (0000000000001618) ----------------------------------------------------
void term_proc()
{
    ;
}

// nfuncs=39 queued=25 decompiled=25 lumina nreq=0 worse=0 better=0
// ALL OK, 25 function(s) have been successfully decompiled
```

flag 值：

flag{welcome_changchengbei}（签到）

flag{Baby_Vmmm_Pr0tect!}（VVM）