ctf up怎么写 write_Hgame Ctf write up (3&4）

原创

Han H 于 2021-01-14 10:59:37 发布 309 收藏
文章标签： ctf up怎么写 write

本帖最后由 yechen123 于 2019-3-9 21:06 编辑

这是第三周和第四周的逆向题

好像没有师傅写wp

恰巧第一次碰到webassembly 想记录一下

题目.rar

(135.21 KB, 下载次数: 31)

2019-3-6 22:35 上传

点击文件名下载附件

下载积分: 吾爱币 -1 CB

第三周

0x00 easy_math

[Asm] 纯文本查看 复制代码int __cdecl main(int argc, const char **argv, const char **envp)

{

__int64 v3; // rax

__int64 v4; // rdx

unsigned int v5; // ebx

__int64 v6; // rdx

__int64 v7; // rax

__int64 v8; // rax

__int64 v9; // rdx

__int64 v10; // rax

__int64 v11; // rax

__int64 v12; // rdx

__int64 v13; // rax

```
__int64 v14; // rdx
__int64 v15; // rax
__int64 v16; // rax
__int64 v17; // rdx
__int64 v18; // rax
unsigned int input; // [rsp-48h] [rbp-48h]
unsigned int v21; // [rsp-44h] [rbp-44h]
__int64 flags; // [rsp-40h] [rbp-40h]
unsigned __int64 v23; // [rsp-18h] [rbp-18h]
v23 = __readfsqword(0x28u);
v3 = std::operator<<<:char_traits>>(
(__int64)&std::cout,
(__int64)"to continue, you have to guess the value of my dice first!",
(__int64)envp);
std::ostream::operator<>);
v21 = rolling_dice();
std::operator<<<:char_traits>>(
(__int64)&std::cout,
(__int64)"now the dice have been rolled, guess what it is: ",
v4);
std::istream::operator>>(&std::cin, &input);
v5 = input;
v7 = std::operator<<<:char_traits>>((__int64)&std::cout, (__int64)"expected: ", v6);
v8 = std::ostream::operator<
v10 = std::operator<<<:char_traits>>(v8, (__int64)", guess: ", v9);
v11 = std::ostream::operator<
std::ostream::operator<>);
if ( input != v21 )
{
v13 = std::operator<<<:char_traits>>((__int64)&std::cout, (__int64)"you are bad at guessing dice", v12);
std::ostream::operator<>);
```

```
exit(0);
}
std::operator<<<:char_traits>>(
(__int64)&std::cout,
(__int64)"wow, you are good at dice-guessing, now give me your flag: ",
v12);
std::__cxx11::basic_string,std::allocator>::basic_string(&flags);
std::operator>>,std::allocator>(&std::cin, &flags);
if ( std::__cxx11::basic_string,std::allocator>::length(&flags) != 32 )
{
v15 = std::operator<<<:char_traits>>((__int64)&std::cout, (__int64)"assert len(flag) == 32", v14);
std::ostream::operator<>);
exit(0);
}
v16 = std::operator<<<:char_traits>>((__int64)&std::cout, (__int64)"now the math part...", v14);
std::ostream::operator<>);
if ( (unsigned __int8)math_part((__int64)&flags) )
v18 = std::operator<<<:char_traits>>(
(__int64)&std::cout,
(__int64)"wow, you are good at doing math too, you deserve to have the flag, just submit it!",
v17);
else
v18 = std::operator<<<:char_traits>>((__int64)&std::cout, (__int64)"you are bad at doing math", v17);
std::ostream::operator<>);
std::__cxx11::basic_string,std::allocator>::~basic_string(&flags);
return 0;
}
```

提示用户先输入一个数字，要与rolling_dice函数返回的数值相同才能进行下一步

由于不涉及到flag 所以可以不理

当用户输入flag后会进入核心函数math_part()

[Asm] 纯文本查看 复制代码signed __int64 __fastcall math_part(__int64 flag_s)

```
{
int v1; // et1
int v2; // edx
int v3; // ecx
int v4; // et1
int v5; // et1
int v6; // et1
int v7; // edx
int v8; // edx
int v9; // et1
int v10; // ecx
int v11; // edx
int v12; // et1
int v13; // edx
int v14; // edx
int v15; // ecx
int v16; // edx
int v17; // et1
int v18; // edx
int v19; // et1
int v20; // ecx
int v21; // edx
int v22; // ecx
int v23; // et1
int v24; // edx
int v25; // edx
int v26; // ecx
int v27; // ecx
int v28; // ecx
int v29; // et1
int v30; // ecx
```

```
int v31; // ecx
int v32; // edx
signed __int64 result; // rax
char *flag; // [rsp-8h] [rbp-8h]
flag = (char *)std::__cxx11::basic_string,std::allocator>::c_str(flag_s);
v1 = 76 * flag[21]
+ 31 * flag[9]
+ 87 * flag[28]
+ 54 * flag[2]
+ 74 * flag[5]
+ 99 * flag[26]
+ 94 * flag[3]
+ 84 * flag[19]
+ 32 * flag[15]
+ 90 * flag[27]
+ 16 * flag[14]
+ 19 * flag[8]
+ 33 * flag[20]
+ 35 * flag[31]
+ 65 * flag[29]
+ 47 * flag[12]
+ 3 * flag[1]
+ 57 * flag[7]
+ 5 * flag[17]
+ 70 * flag[13]
+ 28 * flag[24]
+ 79 * flag[11]
+ 63 * flag[23]
+ 66 * flag[30]
+ 28 * flag[10]
+ flag[4];
```

```
if ( 82 * flag[16] + 58 * flag[25] + v1 + 81 * flag[6] + 61 * flag[18] + 31 * flag[22] + 71 * *flag != 0x237F5 )
goto LABEL_37;
v2 = 55 * flag[6]
+ 38 * flag[9]
+ 39 * flag[18]
+ 73 * flag[24]
+ 86 * flag[13]
+ 18 * flag[11]
+ 40 * flag[21]
+ 40 * flag[26]
+ 54 * flag[14]
+ 81 * flag[10]
+ 71 * flag[27]
+ 20 * flag[8]
+ 16 * flag[28]
+ 65 * flag[30]
+ 87 * flag[3]
+ 14 * flag[16]
+ flag[5]
+ 41 * *flag
+ 58 * flag[15]
+ 73 * flag[2]
+ 46 * flag[23]
+ 7 * flag[19]
+ 89 * flag[17]
+ 65 * flag[25]
+ 43 * flag[7]
+ 6 * flag[20];
if ( v2 + 60 * flag[12] + 40 * flag[31] + 57 * flag[29] + 40 * flag[4] + 30 * flag[1] + 63 * flag[22] != 0x1F21D )
goto LABEL_37;
v3 = 53 * flag[10]
```

```
+ 82 * flag[14]
+ 70 * flag[5]
+ 84 * flag[2]
+ 57 * flag[19]
+ 92 * flag[27]
+ 57 * flag[11]
+ 77 * flag[4]
+ 49 * flag[8]
+ 62 * flag[29]
+ 97 * flag[22]
+ 47 * flag[1]
+ 30 * flag[16]
+ 45 * flag[30]
+ 94 * flag[28]
+ 6 * flag[9]
+ 83 * flag[20]
+ 18 * flag[23]
+ 97 * flag[15]
+ 11 * flag[12]
+ 35 * flag[7]
+ 81 * flag[26]
+ 67 * flag[13]
+ 11 * flag[31]
+ 84 * flag[24];
if ( 28 * flag[6] + 17 * flag[21] + 18 * flag[3] + v3 + 63 * flag[25] + 61 * flag[18] != 0x22863 )
goto LABEL_37;
v4 = 14 * flag[24]
+ 46 * flag[6]
+ 56 * flag[7]
+ 13 * flag[2]
+ 82 * flag[11]
```

```
+ 49 * flag[30]
+ 97 * flag[18]
+ 50 * flag[14]
+ 83 * flag[27]
+ 38 * flag[13]
+ 49 * flag[29]
+ 9 * flag[4]
+ 91 * flag[20]
+ 33 * flag[25]
+ 4 * flag[22]
+ 5 * flag[17]
+ 61 * flag[15]
+ 65 * flag[3]
+ 68 * flag[28]
+ 6 * flag[16]
+ (flag[8] << 6)
+ 56 * flag[9]
+ 67 * flag[10]
+ 5 * flag[5]
+ flag[21]
+ 10 * flag[19];
if ( 86 * flag[23] + 52 * flag[1] + v4 + 83 * flag[12] + 37 * flag[26] + 85 * *flag != 0x1CA87 )
goto LABEL_37;
v5 = 9 * flag[28]
+ 63 * flag[5]
+ 20 * flag[4]
+ 96 * flag[8]
+ 39 * flag[11]
+ 91 * flag[1]
+ 40 * flag[9]
+ 85 * flag[14]
```

```
+ 62 * flag[16]
+ 95 * flag[19]
+ 34 * flag[22]
+ 67 * flag[31]
+ 51 * flag[27]
+ 45 * flag[26]
+ 92 * flag[15]
+ 91 * flag[21]
+ 85 * flag[13]
+ 12 * flag[7]
+ 26 * flag[23]
+ 56 * flag[30]
+ 82 * flag[18]
+ 72 * flag[17]
+ 54 * flag[6]
+ 17 * flag[12]
+ 84 * flag[29]
+ 17 * *flag;
if ( 53 * flag[3] + 91 * flag[2] + 57 * flag[25] + 66 * flag[20] + v5 + 8 * flag[24] + 63 * flag[10] != 0x261F8 )
goto LABEL_37;
v6 = 88 * flag[9]
+ 48 * flag[4]
+ 83 * flag[13]
+ 66 * flag[7]
+ 60 * flag[30]
+ 57 * flag[6]
+ 85 * flag[17]
+ 71 * flag[28]
+ 98 * flag[24]
+ 83 * flag[10]
+ 12 * flag[1]
```

```
+ 72 * flag[31]
+ 12 * flag[22]
+ 80 * flag[20]
+ 15 * flag[19]
+ 81 * flag[21]
+ 87 * *flag
+ 37 * flag[16]
+ 4 * flag[15]
+ 41 * flag[3]
+ 84 * flag[26]
+ 56 * flag[25]
+ 84 * flag[14]
+ 41 * flag[27]
+ 98 * flag[18]
+ 18 * flag[2];
if ( 55 * flag[23] + v6 + 95 * flag[11] + 33 * flag[29] + 66 * flag[8] != 0x245E3 )
goto LABEL_37;
v7 = 57 * flag[21]
+ 63 * flag[12]
+ 4 * flag[14]
+ 59 * flag[31]
+ 15 * flag[23]
+ 12 * flag[25]
+ 58 * flag[5]
+ 40 * flag[4]
+ 26 * flag[30]
+ 8 * flag[15]
+ 25 * flag[6]
+ 97 * flag[10]
+ 12 * flag[28]
+ 74 * flag[26]
```

```
+ 65 * flag[8]
+ 93 * flag[27]
+ 18 * flag[22]
+ 84 * flag[2]
+ 7 * flag[1]
+ 22 * flag[18]
+ 9 * flag[17]
+ 89 * flag[19]
+ 72 * flag[13]
+ 47 * flag[20]
+ 7 * flag[29];
if ( 43 * flag[16] + 47 * *flag + 53 * flag[24] + 75 * flag[11] + v7 + 8 * flag[9] + 24 * flag[7] + 75 * flag[3] != 121517 )
goto LABEL_37;
v8 = 86 * flag[17]
+ 74 * *flag
+ 72 * flag[4]
+ 27 * flag[20]
+ 88 * flag[9]
+ (flag[21] << 6)
+ 52 * flag[15]
+ 4 * flag[19]
+ 8 * flag[1]
+ 16 * flag[13]
+ 54 * flag[25]
+ 8 * flag[29]
+ 52 * flag[23]
+ 14 * flag[10]
+ 88 * flag[18]
+ 33 * flag[8]
+ 99 * flag[27]
```

```
+ 65 * flag[14]
+ 66 * flag[5]
+ 36 * flag[6]
+ 58 * flag[16]
+ 63 * flag[22]
+ 93 * flag[3]
+ 96 * flag[11]
+ 26 * flag[26]
+ 65 * flag[12];
if ( 77 * flag[30] + 89 * flag[31] + 55 * flag[7] + v8 + 42 * flag[28] + 14 * flag[2] + 57 * flag[24] != 0x24F96 )
goto LABEL_37;
v9 = 51 * flag[7]
+ 42 * flag[4]
+ 78 * flag[8]
+ 45 * flag[25]
+ 63 * flag[30]
+ 85 * flag[26]
+ 30 * flag[29]
+ 83 * flag[14]
+ 62 * flag[31]
+ 71 * flag[22]
+ 45 * flag[17]
+ (flag[6] << 6)
+ 87 * flag[23]
+ 49 * flag[28]
+ 14 * *flag
+ 4 * flag[21]
+ 63 * flag[5]
+ 53 * flag[13]
+ 19 * flag[19]
+ 44 * flag[16]
```

```
+ 5 * flag[3]
+ 74 * flag[15]
+ 19 * flag[18]
+ 89 * flag[11]
+ 11 * flag[20]
+ 34 * flag[12];
if ( 53 * flag[24] + 95 * flag[27] + v9 + 14 * flag[1] + 87 * flag[10] + 63 * flag[9] + 70 * flag[2] != 142830 )
goto LABEL_37;
v10 = 13 * flag[29]
+ 11 * flag[22]
+ 41 * flag[5]
+ 38 * flag[13]
+ 90 * flag[31]
+ 68 * flag[7]
+ 56 * flag[14]
+ 4 * flag[23]
+ 66 * flag[28]
+ 28 * flag[1]
+ 6 * flag[12]
+ 91 * flag[16]
+ 59 * flag[3]
+ 81 * flag[17]
+ 44 * flag[2]
+ 33 * flag[24]
+ 34 * flag[19]
+ 17 * flag[18]
+ 77 * flag[25]
+ 25 * flag[8]
+ 8 * flag[6]
+ 10 * flag[30]
+ 66 * flag[20];
```

```
if ( 69 * *flag
+ 67 * flag[9]
+ 57 * flag[15]
+ 77 * flag[10]
+ 67 * flag[26]
+ 94 * flag[11]
+ v10
+ 41 * flag[27]
+ 29 * flag[21] != 0x1DED9 )
goto LABEL_37;
v11 = 23 * flag[25]
+ 32 * flag[3]
+ 72 * flag[15]
+ 41 * flag[26]
+ 33 * flag[30]
+ 82 * flag[13]
+ 20 * *flag
+ 7 * flag[12]
+ 25 * flag[29]
+ 39 * flag[21]
+ 57 * flag[14]
+ 14 * flag[16]
+ 24 * flag[24]
+ 37 * flag[22]
+ 71 * flag[10]
+ 65 * flag[23]
+ 46 * flag[8]
+ 40 * flag[19]
+ 77 * flag[27]
+ 80 * flag[18]
+ 88 * flag[6]
```

```
+ 20 * flag[31]
+ 83 * flag[11]
+ 73 * flag[1]
+ 8 * flag[5]
+ 15 * flag[20];
if ( 31 * flag[9] + 17 * flag[4] + 6 * flag[28] + v11 + 70 * flag[7] + 24 * flag[17] + 16 * flag[2] != 0x19B4D )
goto LABEL_37;
v12 = 41 * flag[24]
+ 45 * flag[30]
+ 82 * flag[20]
+ 86 * flag[19]
+ 99 * flag[9]
+ 96 * flag[22]
+ 85 * flag[28]
+ 70 * flag[5]
+ 77 * flag[23]
+ 80 * flag[11]
+ 40 * flag[31]
+ 66 * flag[12]
+ 12 * flag[2]
+ 77 * flag[15]
+ 72 * flag[4]
+ 42 * flag[26]
+ 81 * flag[27]
+ 90 * flag[13]
+ 37 * flag[16]
+ 29 * flag[17]
+ 20 * flag[29]
+ 85 * flag[6]
+ 6 * flag[7]
+ 2 * *flag
```

```
+ 72 * flag[1]
+ 75 * flag[14];
if ( 25 * flag[21] + 79 * flag[3] + v12 + 40 * flag[25] + 29 * flag[8] + 25 * flag[10] != 0x2519A )
goto LABEL_37;
v13 = 42 * flag[31]
+ 95 * flag[30]
+ 58 * flag[8]
+ 47 * flag[13]
+ 65 * flag[15]
+ 24 * flag[17]
+ 97 * flag[10]
+ 24 * flag[21]
+ 28 * *flag
+ 77 * flag[5]
+ 97 * flag[6]
+ 24 * flag[26]
+ 32 * flag[12]
+ 5 * flag[25]
+ 55 * flag[28]
+ 9 * flag[23]
+ 85 * flag[4]
+ 6 * flag[9]
+ 61 * flag[19]
+ 12 * flag[3]
+ 76 * flag[7]
+ 36 * flag[27]
+ 77 * flag[24]
+ 24 * flag[29]
+ 67 * flag[14]
+ 19 * flag[16];
if ( 83 * flag[11] + 75 * flag[1] + v13 + 47 * flag[20] + 13 * flag[22] != 125609 )
```

```
goto LABEL_37;
v14 = flag[1]
+ 88 * flag[3]
+ 90 * *flag
+ 4 * flag[23]
+ 46 * flag[7]
+ 54 * flag[16]
+ 16 * flag[6]
+ 89 * flag[22]
+ 76 * flag[27]
+ 38 * flag[17]
+ 3 * flag[5]
+ 70 * flag[14]
+ 3 * flag[24]
+ 24 * flag[13]
+ 54 * flag[2]
+ 20 * flag[8]
+ 83 * flag[12]
+ 21 * flag[15]
+ 77 * flag[18]
+ 31 * flag[19]
+ 59 * flag[21]
+ 33 * flag[20]
+ 84 * flag[11]
+ 19 * flag[29]
+ 38 * flag[26]
+ 63 * flag[31];
if ( 30 * flag[25] + 41 * flag[28] + 65 * flag[10] + v14 + 16 * flag[30] + 15 * flag[4] + 39 * flag[9] != 123069 )
goto LABEL_37;
v15 = 27 * flag[18]
+ 48 * flag[4]
```

```
+ 13 * flag[20]
+ 44 * flag[10]
+ 70 * flag[12]
+ 44 * flag[17]
+ 22 * flag[23]
+ 55 * flag[14]
+ 73 * flag[26]
+ 55 * flag[8]
+ 58 * flag[11]
+ 31 * flag[30]
+ 78 * flag[29]
+ 19 * flag[25]
+ 52 * flag[31]
+ 27 * flag[21]
+ 38 * flag[27]
+ 40 * flag[28]
+ 35 * flag[1]
+ 48 * flag[22]
+ 71 * flag[15]
+ 24 * flag[6]
+ 89 * flag[16]
+ 37 * flag[3]
+ 78 * flag[2];
if ( 6 * flag[9] + 19 * flag[19] + v15 + 3 * flag[5] + 52 * flag[24] + 40 * flag[7] != 113842 )
goto LABEL_37;
v16 = 31 * flag[12]
+ 35 * flag[10]
+ 54 * flag[20]
+ 26 * flag[29]
+ 29 * flag[3]
+ 2 * flag[23]
```

```
+ 46 * *flag
+ 30 * flag[26]
+ 56 * flag[27]
+ 100 * flag[11]
+ 43 * flag[1]
+ 15 * flag[4]
+ 79 * flag[17]
+ 12 * flag[5]
+ 38 * flag[9]
+ 3 * flag[30]
+ 16 * flag[21]
+ 19 * flag[13]
+ 67 * flag[19]
+ 37 * flag[28]
+ flag[7]
+ 73 * flag[16]
+ 85 * flag[6]
+ 17 * flag[14]
+ 90 * flag[22]
+ 15 * flag[2];
if ( 95 * flag[8] + 92 * flag[18] + 84 * flag[31] + v16 + 43 * flag[25] + 96 * flag[24] != 119824 )
goto LABEL_37;
v17 = 92 * flag[20]
+ 43 * flag[23]
+ 16 * flag[19]
+ 92 * flag[5]
+ 49 * flag[26]
+ 44 * flag[2]
+ 26 * flag[29]
+ (flag[25] << 6)
+ 45 * flag[24]
```

```
+ 99 * flag[11]
+ 43 * flag[4]
+ 75 * flag[21]
+ 53 * flag[31]
+ 18 * flag[18]
+ 11 * flag[13]
+ 52 * *flag
+ 16 * flag[8]
+ 9 * flag[7]
+ 77 * flag[16]
+ 33 * flag[10]
+ 86 * flag[1]
+ 33 * flag[3]
+ 29 * flag[9]
+ 6 * flag[12]
+ 91 * flag[14]
+ 36 * flag[15];
if ( 36 * flag[22] + 69 * flag[28] + 77 * flag[6] + v17 + 94 * flag[27] + 13 * flag[30] + 89 * flag[17] != 135873 )
goto LABEL_37;
v18 = 68 * flag[2]
+ 83 * flag[10]
+ 47 * flag[5]
+ 85 * flag[13]
+ 22 * flag[8]
+ 92 * flag[27]
+ 75 * flag[28]
+ 43 * flag[3]
+ 29 * flag[22]
+ 92 * *flag
+ 54 * flag[16]
+ 17 * flag[30]
```

```
+ 78 * flag[18]
+ 7 * flag[23]
+ 69 * flag[21]
+ 63 * flag[31]
+ 71 * flag[4]
+ 10 * flag[6]
+ 66 * flag[14]
+ 25 * flag[26]
+ 32 * flag[1]
+ 48 * flag[19]
+ 86 * flag[11]
+ 20 * flag[25]
+ 78 * flag[20]
+ 25 * flag[17];
if ( 16 * flag[7] + flag[15] + 82 * flag[9] + 60 * flag[29] + v18 + 76 * flag[12] + 13 * flag[24] != 142509 )
goto LABEL_37;
v19 = 77 * flag[9]
+ 56 * flag[30]
+ 79 * flag[2]
+ 71 * flag[29]
+ 95 * flag[28]
+ 87 * flag[24]
+ 62 * flag[16]
+ 85 * flag[26]
+ 43 * flag[20]
+ 67 * flag[15]
+ 97 * flag[8]
+ 80 * *flag
+ 23 * flag[3]
+ 95 * flag[25]
+ 82 * flag[21]
```

```
+ 66 * flag[31]
+ 5 * flag[4]
+ 66 * flag[27]
+ 25 * flag[12]
+ 4 * flag[5]
+ 12 * flag[7]
+ 85 * flag[1]
+ 10 * flag[6]
+ 45 * flag[11]
+ 28 * flag[18]
+ 26 * flag[19];
if ( 88 * flag[22] + 23 * flag[13] + 18 * flag[14] + v19 + 48 * flag[23] + 45 * flag[17] != 148888 )
goto LABEL_37;
v20 = 81 * flag[30]
+ 21 * flag[6]
+ 72 * flag[11]
+ 48 * flag[18]
+ 2 * flag[19]
+ 42 * flag[10]
+ 22 * flag[24]
+ 99 * flag[2]
+ 78 * flag[22]
+ 83 * flag[12]
+ 60 * flag[9]
+ 59 * flag[13]
+ 15 * flag[5]
+ 25 * flag[20]
+ 43 * flag[15]
+ 56 * flag[28]
+ 33 * flag[25]
+ 71 * flag[23]
```

```
+ 31 * *flag
+ 95 * flag[3]
+ 73 * flag[17]
+ 86 * flag[14]
+ 15 * flag[21]
+ 61 * flag[7]
+ 12 * flag[29]
+ 95 * flag[26];
if ( 25 * flag[8] + v20 + 13 * flag[1] + 100 * flag[16] + 11 * flag[4] + 79 * flag[27] != 138023 )
goto LABEL_37;
v21 = 53 * flag[27]
+ 52 * flag[29]
+ 70 * flag[22]
+ 35 * flag[30]
+ 50 * flag[16]
+ 59 * flag[8]
+ 75 * flag[10]
+ 55 * flag[20]
+ 23 * *flag
+ 52 * flag[17]
+ 47 * flag[3]
+ 91 * flag[13]
+ 46 * flag[7]
+ 42 * flag[14]
+ 79 * flag[26]
+ 87 * flag[21]
+ 30 * flag[6]
+ 26 * flag[1]
+ 57 * flag[31]
+ 33 * flag[12]
+ 51 * flag[9]
```

```
+ 56 * flag[24]
+ 59 * flag[11]
+ 36 * flag[23]
+ 88 * flag[4]
+ 28 * flag[2];
if ( 37 * flag[28] + 62 * flag[25] + 42 * flag[18] + v21 + 44 * flag[15] + 19 * flag[19] + 74 * flag[5] != 142299 )
goto LABEL_37;
v22 = 80 * flag[21]
+ 43 * flag[31]
+ 67 * flag[16]
+ 55 * flag[13]
+ 95 * flag[24]
+ 46 * flag[28]
+ 93 * flag[5]
+ 75 * flag[20]
+ 14 * flag[25]
+ 24 * flag[26]
+ 50 * flag[29]
+ 70 * flag[15]
+ 63 * flag[30]
+ 77 * flag[23]
+ 96 * flag[19]
+ 66 * flag[11]
+ 72 * flag[27]
+ 94 * flag[4]
+ 63 * flag[22]
+ 69 * flag[3]
+ 73 * flag[1]
+ 60 * flag[7]
+ 9 * flag[2]
+ 39 * flag[17]
```

```
+ 25 * *flag
+ 49 * flag[14];
if ( v22 + 48 * flag[8] + 86 * flag[9] + 72 * flag[10] + 23 * flag[18] + 21 * flag[6] != 155777 )
goto LABEL_37;
v23 = 27 * flag[11]
+ 40 * flag[8]
+ 53 * flag[15]
+ 40 * flag[18]
+ 56 * flag[3]
+ 2 * flag[2]
+ 32 * flag[4]
+ 90 * flag[1]
+ 54 * flag[16]
+ 20 * flag[9]
+ 86 * flag[17]
+ 82 * flag[31]
+ 43 * flag[25]
+ 43 * flag[13]
+ 86 * flag[21]
+ 17 * *flag
+ (flag[14] << 6)
+ 6 * flag[30]
+ 86 * flag[5]
+ 15 * flag[7]
+ 46 * flag[12]
+ 21 * flag[26]
+ 90 * flag[20]
+ 19 * flag[6]
+ 93 * flag[23]
+ 31 * flag[27];
if ( 25 * flag[24] + 11 * flag[22] + v23 + 62 * flag[29] + 21 * flag[19] + 42 * flag[10] != 117687 )
```

```
goto LABEL_37;
v24 = flag[27]
+ 66 * flag[18]
+ 40 * flag[17]
+ 17 * *flag
+ 27 * flag[19]
+ 26 * flag[31]
+ 57 * flag[24]
+ 35 * flag[3]
+ 80 * flag[1]
+ 67 * flag[5]
+ 85 * flag[6]
+ 7 * flag[15]
+ 93 * flag[8]
+ 3 * flag[22]
+ 77 * flag[12]
+ 12 * flag[28]
+ 4 * flag[2]
+ 27 * flag[9]
+ 53 * flag[25]
+ 37 * flag[30]
+ 43 * flag[23]
+ 33 * flag[4]
+ 39 * flag[26]
+ 7 * flag[7]
+ 75 * flag[10]
+ 15 * flag[14];
if ( 89 * flag[21] + 100 * flag[13] + v24 + 45 * flag[20] + 36 * flag[29] + 78 * flag[11] + 31 * flag[16] != 117383 )
goto LABEL_37;
v25 = 71 * flag[16]
+ 4 * flag[1]
```

```
+ 77 * flag[31]
+ 83 * flag[2]
+ 11 * flag[30]
+ 53 * flag[19]
+ 85 * flag[12]
+ 67 * flag[13]
+ 39 * flag[8]
+ 45 * flag[24]
+ 84 * flag[22]
+ 99 * flag[14]
+ 38 * flag[3]
+ 29 * flag[4]
+ 90 * flag[9]
+ 61 * flag[18]
+ 40 * flag[7]
+ (flag[17] << 6)
+ 9 * flag[25]
+ 86 * flag[29]
+ 80 * flag[21]
+ 4 * flag[15]
+ 96 * flag[23]
+ 99 * flag[10]
+ 40 * flag[27];
if ( 73 * flag[20] + 16 * flag[26] + 100 * flag[5] + 71 * flag[28] + v25 + 4 * *flag + 56 * flag[11] != 155741
|| (v26 = 87 * flag[2]
+ 86 * flag[24]
+ 76 * flag[14]
+ 38 * flag[23]
+ 85 * flag[3]
+ 71 * flag[22]
+ 42 * flag[29]
```

+ 85 * flag[30]

+ 14 * flag[10]

+ 17 * flag[13]

+ 42 * flag[25]

+ 11 * flag[19]

+ 44 * flag[15]

+ 21 * flag[4]

+ 60 * flag[16]

+ 28 * flag[6]

+ 46 * flag[20]

+ 25 * flag[9]

+ 77 * flag[31]

+ 21 * flag[8]

+ 85 * flag[7]

+ 36 * flag[1]

+ 91 * flag[27]

+ 21 * flag[28]

+ 38 * flag[17],

(flag[12] << 6) + 76 * *flag + 5 * flag[11] + v26 + 3 * flag[26] + 61 * flag[21] + 15 * flag[5] + 32 * flag[18] != 132804)

|| (v27 = 36 * flag[1]

+ 60 * flag[3]

+ 84 * flag[11]

+ 19 * flag[26]

+ 76 * flag[27]

+ 86 * flag[16]

+ 92 * flag[8]

+ 96 * flag[14]

+ 60 * flag[21]

+ 23 * flag[4]

+ 60 * flag[12]

+ 50 * flag[23]

+ 78 * flag[22]

+ 45 * flag[9]

+ 42 * flag[18]

+ 10 * flag[2]

+ 60 * flag[20]

+ 24 * flag[24]

+ 77 * flag[7]

+ 41 * flag[6]

+ 29 * flag[13]

+ 33 * flag[5]

+ 2 * flag[15]

+ 33 * flag[29]

+ 39 * flag[31],

95 * flag[30] + 75 * flag[28] + 3 * flag[10] + v27 + 41 * flag[25] + 100 * flag[19] + 9 * flag[17] + 79 * *flag != 145568)

|| (v28 = 25 * flag[26]

+ 98 * flag[24]

+ 15 * flag[6]

+ 50 * flag[18]

+ 88 * flag[20]

+ 74 * flag[11]

+ 83 * flag[1]

+ 86 * flag[21]

+ 52 * flag[7]

+ 39 * flag[10]

+ 40 * flag[13]

+ 82 * flag[28]

+ 37 * flag[3]

+ 45 * *flag

+ 18 * flag[25]

```
+ 2 * flag[29]
+ 6 * flag[12]
+ 78 * flag[31]
+ 37 * flag[2]
+ 57 * flag[23]
+ 3 * flag[4]
+ 59 * flag[8]
+ 73 * flag[15]
+ flag[22]
+ 18 * flag[9]
+ 35 * flag[14],
68 * flag[5] + 98 * flag[27] + 98 * flag[16] + 10 * flag[19] + v28 + 20 * flag[17] + 54 * flag[30] != 130175)
|| (v29 = 68 * flag[23]
+ 60 * flag[18]
+ 93 * flag[20]
+ 100 * flag[11]
+ 98 * flag[14]
+ 32 * flag[3]
+ 15 * flag[21]
+ 79 * *flag
+ 6 * flag[24]
+ 62 * flag[26]
+ 96 * flag[6]
+ 68 * flag[22]
+ 9 * flag[7]
+ 88 * flag[5]
+ 18 * flag[27]
+ 70 * flag[9]
+ 96 * flag[25]
+ 89 * flag[4]
+ 14 * flag[31]
```

+ 83 * flag[17]

+ 19 * flag[15]

+ 44 * flag[1]

+ 96 * flag[8]

+ 87 * flag[16]

+ 48 * flag[2]

+ 95 * flag[13],

60 * flag[10] + 50 * flag[12] + 30 * flag[29] + 90 * flag[19] + v29 + 73 * flag[28] + 92 * flag[30] != 171986)

|| (v30 = 86 * flag[9]

+ 20 * flag[7]

+ 29 * flag[16]

+ 31 * flag[14]

+ 83 * flag[26]

+ 11 * flag[4]

+ 29 * flag[19]

+ 82 * flag[13]

+ 84 * flag[10]

+ 70 * flag[1]

+ 52 * flag[12]

+ 40 * flag[6]

+ 91 * flag[8]

+ 6 * flag[17]

+ 77 * flag[28]

+ 56 * flag[5]

+ 86 * flag[23]

+ 63 * flag[31]

+ 26 * flag[27]

+ 19 * flag[22]

+ 50 * flag[3]

+ 15 * flag[15]

+ 67 * flag[2]

```
+ 37 * flag[24]
+ 84 * flag[18],
53 * flag[30] + 87 * flag[25] + 23 * flag[29] + 80 * flag[20] + v30 + 81 * flag[21] + 93 * *flag != 151676)
|| (v31 = 12 * flag[11]
+ 82 * flag[24]
+ 100 * flag[8]
+ 29 * flag[26]
+ 97 * flag[12]
+ 32 * flag[6]
+ 26 * flag[27]
+ 46 * flag[19]
+ 8 * (flag[25] + 9 * *flag + 2 * flag[17])
+ 63 * flag[10]
+ 39 * flag[29]
+ 81 * flag[15]
+ 51 * flag[13]
+ 31 * flag[30]
+ 49 * flag[4]
+ 3 * flag[22]
+ 26 * flag[28]
+ 15 * flag[20]
+ 89 * flag[2]
+ 5 * flag[31]
+ 47 * flag[18]
+ 19 * flag[23]
+ 98 * flag[9],
29 * flag[3] + 93 * flag[5] + 67 * flag[21] + v31 + 15 * flag[16] + 49 * flag[1] != 128223)
|| (v32 = 84 * flag[25]
+ 91 * flag[10]
+ 67 * flag[22]
+ 77 * flag[15]
```

```
+ 23 * flag[26]
+ 38 * flag[4]
+ 3 * flag[31]
+ 76 * flag[13]
+ 50 * *flag
+ 74 * flag[11]
+ 45 * flag[28]
+ 58 * flag[29]
+ 39 * flag[5]
+ 95 * flag[9]
+ 26 * flag[16]
+ 23 * flag[8]
+ 28 * flag[24]
+ 89 * flag[1]
+ 88 * flag[18]
+ 3 * flag[3]
+ 59 * flag[20]
+ 80 * flag[23]
+ 49 * flag[17]
+ 56 * flag[21]
+ 32 * flag[27]
+ 24 * flag[2],
13 * flag[14] + 73 * flag[19] + 99 * flag[7] + 76 * flag[12] + v32 + 77 * flag[30] + 18 * flag[6] != 138403) )
{
LABEL_37:
result = 0LL;
}
else
{
result = 1LL;
}
```

return result;

}

因为z3这个库的类型问题 可以公式里边的<<6 改成*64

判断flag的 可以用z3来解得到flag

[Asm] 纯文本查看 复制代码from z3 import *

x = Solver()

flag = [Int('flag%d'%i) for i in range(32)]

x.add((82 * flag[16] + 58 * flag[25] + 76 * flag[21] + 31 * flag[9]+ 87 * flag[28]+ 54 * flag[2]+ 74 * flag[5]+ 99 * flag[26]+ 94 * flag[3]+ 84 * flag[19]+ 32 * flag[15]+ 90 * flag[27]+ 16 * flag[14]+ 19 * flag[8]+ 33 * flag[20]+ 35 * flag[31]+ 65 * flag[29]+ 47 * flag[12]+ 3 * flag[1]+ 57 * flag[7]+ 5 * flag[17]+ 70 * flag[13]+ 28 * flag[24]+ 79 * flag[11]+ 63 * flag[23]+ 66 * flag[30]+ 28 * flag[10]+ flag[4] + 81 * flag[6] + 61 * flag[18] + 31 * flag[22] + 71 * flag[0]) == 0x237F5)

x.add((55 * flag[6]+ 38 * flag[9]+ 39 * flag[18]+ 73 * flag[24]+ 86 * flag[13]+ 18 * flag[11]+ 40 * flag[21]+ 40 * flag[26]+ 54 * flag[14]+ 81 * flag[10]+ 71 * flag[27]+ 20 * flag[8]+ 16 * flag[28]+ 65 * flag[30]+ 87 * flag[3]+ 14 * flag[16]+ flag[5]+ 41 * flag[0]+ 58 * flag[15]+ 73 * flag[2]+ 46 * flag[23]+ 7 * flag[19]+ 89 * flag[17]+ 65 * flag[25]+ 43 * flag[7]+ 6 * flag[20] + 60 * flag[12] + 40 * flag[31] + 57 * flag[29] + 40 * flag[4] + 30 * flag[1] + 63 * flag[22]) == 0x1F21D)

x.add((28 * flag[6] + 17 * flag[21] + 18 * flag[3] + 53 * flag[10]+ 82 * flag[14]+ 70 * flag[5]+ 84 * flag[2]+ 57 * flag[19]+ 92 * flag[27]+ 57 * flag[11]+ 77 * flag[4]+ 49 * flag[8]+ 62 * flag[29]+ 97 * flag[22]+ 47 * flag[1]+ 30 * flag[16]+ 45 * flag[30]+ 94 * flag[28]+ 6 * flag[9]+ 83 * flag[20]+ 18 * flag[23]+ 97 * flag[15]+ 11 * flag[12]+ 35 * flag[7]+ 81 * flag[26]+ 67 * flag[13]+ 11 * flag[31]+ 84 * flag[24] + 63 * flag[25] + 61 * flag[18]) == 0x22863)

x.add((86 * flag[23] + 52 * flag[1] + 14 * flag[24]+ 46 * flag[6]+ 56 * flag[7]+ 13 * flag[2]+ 82 * flag[11]+ 49 * flag[30]+ 97 * flag[18]+ 50 * flag[14]+ 83 * flag[27]+ 38 * flag[13]+ 49 * flag[29]+ 9 * flag[4]+ 91 * flag[20]+ 33 * flag[25]+ 4 * flag[22]+ 5 * flag[17]+ 61 * flag[15]+ 65 * flag[3]+ 68 * flag[28]+ 6 * flag[16]+ (flag[8] * 64)+ 56 * flag[9]+ 67 * flag[10]+ 5 * flag[5]+ flag[21]+ 10 * flag[19] + 83 * flag[12] + 37 * flag[26] + 85 * flag[0]) == 0x1CA87)

x.add(53 * flag[3] + 91 * flag[2] + 57 * flag[25] + 66 * flag[20] + 9 * flag[28]+ 63 * flag[5]+ 20 * flag[4]+ 96 * flag[8]+ 39 * flag[11]+ 91 * flag[1]+ 40 * flag[9]+ 85 * flag[14]+ 62 * flag[16]+ 95 * flag[19]+ 34 * flag[22]+ 67 * flag[31]+ 51 * flag[27]+ 45 * flag[26]+ 92 * flag[15]+ 91 * flag[21]+ 85 * flag[13]+ 12 * flag[7]+ 26 * flag[23]+ 56 * flag[30]+ 82 * flag[18]+ 72 * flag[17]+ 54 * flag[6]+ 17 * flag[12]+ 84 * flag[29]+ 17 * flag[0] + 8 * flag[24] + 63 * flag[10] == 0x261F8)

x.add(55 * flag[23] + 88 * flag[9]+ 48 * flag[4]+ 83 * flag[13]+ 66 * flag[7]+ 60 * flag[30]+ 57 * flag[6]+ 85 * flag[17]+ 71 * flag[28]+ 98 * flag[24]+ 83 * flag[10]+ 12 * flag[1]+ 72 * flag[31]+ 12 * flag[22]+ 80 * flag[20]+ 15 * flag[19]+ 81 * flag[21]+ 87 * flag[0]+ 37 * flag[16]+ 4 * flag[15]+ 41 * flag[3]+ 84 * flag[26]+ 56 * flag[25]+ 84 * flag[14]+ 41 * flag[27]+ 98 * flag[18]+ 18 * flag[2] + 95 * flag[11] + 33 * flag[29] + 66 * flag[8] == 0x245E3)

x.add(43 * flag[16] + 47 * flag[0] + 53 * flag[24] + 75 * flag[11] + 57 * flag[21]+ 63 * flag[12]+ 4 * flag[14]+ 59 * flag[31]+ 15 * flag[23]+ 12 * flag[25]+ 58 * flag[5]+ 40 * flag[4]+ 26 * flag[30]+ 8 * flag[15]+ 25 * flag[6]+ 97 * flag[10]+ 12 * flag[28]+ 74 * flag[26]+ 65 * flag[8]+ 93 * flag[27]+ 18 * flag[22]+ 84 * flag[2]+ 7 * flag[1]+ 22 * flag[18]+ 9 * flag[17]+ 89 * flag[19]+ 72 * flag[13]+ 47 * flag[20]+ 7 * flag[29] + 8 * flag[9] + 24 * flag[7] + 75 * flag[3] == 121517)

x.add(77 * flag[30] + 89 * flag[31] + 55 * flag[7] + 86 * flag[17]+ 74 * flag[0]+ 72 * flag[4]+ 27 * flag[20]+ 88 * flag[9]+ (flag[21] * 64)+ 52 * flag[15]+ 4 * flag[19]+ 8 * flag[1]+ 16 * flag[13]+ 54 * flag[25]+ 8 * flag[29]+ 52 * flag[23]+ 14 * flag[10]+ 88 * flag[18]+ 33 * flag[8]+ 99 * flag[27]+ 65 * flag[14]+ 66 * flag[5]+ 36 * flag[6]+ 58 * flag[16]+ 63 * flag[22]+ 93 * flag[3]+ 96 * flag[11]+ 26 * flag[26]+ 65 * flag[12] + 42 * flag[28] + 14 * flag[2] + 57 * flag[24] == 0x24F96)

x.add(53 * flag[24] + 95 * flag[27] + 51 * flag[7]+ 42 * flag[4]+ 78 * flag[8]+ 45 * flag[25]+ 63 * flag[30]+ 85 * flag[26]+ 30 * flag[29]+ 83 * flag[14]+ 62 * flag[31]+ 71 * flag[22]+ 45 * flag[17]+ (flag[6] * 64)+ 87 * flag[23]+ 49 * flag[28]+ 14 * flag[0]+ 4 * flag[21]+ 63 * flag[5]+ 53 * flag[13]+ 19 * flag[19]+ 44 * flag[16]+ 5 * flag[3]+ 74 * flag[15]+ 19 * flag[18]+ 89 * flag[11]+ 11 * flag[20]+ 34 * flag[12] + 14 * flag[1] + 87 * flag[10] + 63 * flag[9] + 70 * flag[2] == 142830)

x.add(69 * flag[0]+ 67 * flag[9]+ 57 * flag[15]+ 77 * flag[10]+ 67 * flag[26]+ 94 * flag[11]+ 13 * flag[29]+ 11 * flag[22]+ 41 * flag[5]+ 38 * flag[13]+ 90 * flag[31]+ 68 * flag[7]+ 56 * flag[14]+ 4 * flag[23]+ 66 * flag[28]+ 28 * flag[1]+ 6 * flag[12]+ 91 * flag[16]+ 59 * flag[3]+ 81 * flag[17]+ 44 * flag[2]+ 33 * flag[24]+ 34 * flag[19]+ 17 * flag[18]+ 77 * flag[25]+ 25 * flag[8]+ 8 * flag[6]+ 10 * flag[30]+ 66 * flag[20]+ 41 * flag[27]+ 29 * flag[21] == 0x1DED9)

x.add(31 * flag[9] + 17 * flag[4] + 6 * flag[28] + 23 * flag[25]+ 32 * flag[3]+ 72 * flag[15]+ 41 * flag[26]+ 33 * flag[30]+ 82 * flag[13]+ 20 * flag[0]+ 7 * flag[12]+ 25 * flag[29]+ 39 * flag[21]+ 57 * flag[14]+ 14 * flag[16]+ 24 * flag[24]+ 37 * flag[22]+ 71 * flag[10]+ 65 * flag[23]+ 46 * flag[8]+ 40 * flag[19]+ 77 * flag[27]+ 80 * flag[18]+ 88 * flag[6]+ 20 * flag[31]+ 83 * flag[11]+ 73 * flag[1]+ 8 * flag[5]+ 15 * flag[20] + 70 * flag[7] + 24 * flag[17] + 16 * flag[2] == 0x19B4D)

x.add(25 * flag[21] + 79 * flag[3] + 41 * flag[24]+ 45 * flag[30]+ 82 * flag[20]+ 86 * flag[19]+ 99 * flag[9]+ 96 * flag[22]+ 85 * flag[28]+ 70 * flag[5]+ 77 * flag[23]+ 80 * flag[11]+ 40 * flag[31]+ 66 * flag[12]+ 12 * flag[2]+ 77 * flag[15]+ 72 * flag[4]+ 42 * flag[26]+ 81 * flag[27]+ 90 * flag[13]+ 37 * flag[16]+ 29 * flag[17]+ 20 * flag[29]+ 85 * flag[6]+ 6 * flag[7]+ 2 * flag[0]+ 72 * flag[1]+ 75 * flag[14] + 40 * flag[25] + 29 * flag[8] + 25 * flag[10] == 0x2519A)

x.add(83 * flag[11] + 75 * flag[1] + 42 * flag[31]+ 95 * flag[30]+ 58 * flag[8]+ 47 * flag[13]+ 65 * flag[15]+ 24 * flag[17]+ 97 * flag[10]+ 24 * flag[21]+ 28 * flag[0]+ 77 * flag[5]+ 97 * flag[6]+ 24 * flag[26]+ 32 * flag[12]+ 5 * flag[25]+ 55 * flag[28]+ 9 * flag[23]+ 85 * flag[4]+ 6 * flag[9]+ 61 * flag[19]+ 12 * flag[3]+ 76 * flag[7]+ 36 * flag[27]+ 77 * flag[24]+ 24 * flag[29]+ 67 * flag[14]+ 19 * flag[16] + 47 * flag[20] + 13 * flag[22] == 125609)

x.add(30 * flag[25] + 41 * flag[28] + 65 * flag[10] + flag[1]+ 88 * flag[3]+ 90 * flag[0]+ 4 * flag[23]+ 46 * flag[7]+ 54 * flag[16]+ 16 * flag[6]+ 89 * flag[22]+ 76 * flag[27]+ 38 * flag[17]+ 3 * flag[5]+ 70 * flag[14]+ 3 * flag[24]+ 24 * flag[13]+ 54 * flag[2]+ 20 * flag[8]+ 83 * flag[12]+ 21 * flag[15]+ 77 * flag[18]+ 31 * flag[19]+ 59 * flag[21]+ 33 * flag[20]+ 84 * flag[11]+ 19 * flag[29]+ 38 * flag[26]+ 63 * flag[31] + 16 * flag[30] + 15 * flag[4] + 39 * flag[9] == 123069)

x.add(6 * flag[9] + 19 * flag[19] + 27 * flag[18]+ 48 * flag[4]+ 13 * flag[20]+ 44 * flag[10]+ 70 * flag[12]+ 44 * flag[17]+ 22 * flag[23]+ 55 * flag[14]+ 73 * flag[26]+ 55 * flag[8]+ 58 * flag[11]+ 31 * flag[30]+ 78 * flag[29]+ 19 * flag[25]+ 52 * flag[31]+ 27 * flag[21]+ 38 * flag[27]+ 40 * flag[28]+ 35 * flag[1]+ 48 * flag[22]+ 71 * flag[15]+ 24 * flag[6]+ 89 * flag[16]+ 37 * flag[3]+ 78 * flag[2] + 3 * flag[5] + 52 * flag[24] + 40 * flag[7] == 113842)

x.add(95 * flag[8] + 92 * flag[18] + 84 * flag[31] + 31 * flag[12]+ 35 * flag[10]+ 54 * flag[20]+ 26 * flag[29]+ 29 * flag[3]+ 2 * flag[23]+ 46 * flag[0]+ 30 * flag[26]+ 56 * flag[27]+ 100 * flag[11]+ 43 * flag[1]+ 15 * flag[4]+ 79 * flag[17]+ 12 * flag[5]+ 38 * flag[9]+ 3 * flag[30]+ 16 * flag[21]+ 19 * flag[13]+ 67 * flag[19]+ 37 * flag[28]+ flag[7]+ 73 * flag[16]+ 85 * flag[6]+ 17 * flag[14]+ 90 * flag[22]+ 15 * flag[2] + 43 * flag[25] + 96 * flag[24] == 119824)

x.add(36 * flag[22] + 69 * flag[28] + 77 * flag[6] + 92 * flag[20]+ 43 * flag[23]+ 16 * flag[19]+ 92 * flag[5]+ 49 * flag[26]+ 44 * flag[2]+ 26 * flag[29]+ (flag[25] * 64)+ 45 * flag[24]+ 99 * flag[11]+ 43 * flag[4]+ 75 * flag[21]+ 53 * flag[31]+ 18 * flag[18]+ 11 * flag[13]+ 52 * flag[0]+ 16 * flag[8]+ 9 * flag[7]+ 77 * flag[16]+ 33 * flag[10]+ 86 * flag[1]+ 33 * flag[3]+ 29 * flag[9]+ 6 * flag[12]+ 91 * flag[14]+ 36 * flag[15] + 94 * flag[27] + 13 * flag[30] + 89 * flag[17] == 135873)

x.add(16 * flag[7] + flag[15] + 82 * flag[9] + 60 * flag[29] + 68 * flag[2]+ 83 * flag[10]+ 47 * flag[5]+ 85 * flag[13]+ 22 * flag[8]+ 92 * flag[27]+ 75 * flag[28]+ 43 * flag[3]+ 29 * flag[22]+ 92 * flag[0]+ 54 * flag[16]+ 17 * flag[30]+ 78 * flag[18]+ 7 * flag[23]+ 69 * flag[21]+ 63 * flag[31]+ 71 * flag[4]+ 10 * flag[6]+ 66 * flag[14]+ 25 * flag[26]+ 32 * flag[1]+ 48 * flag[19]+ 86 * flag[11]+ 20 * flag[25]+ 78 * flag[20]+ 25 * flag[17] + 76 * flag[12] + 13 * flag[24] == 142509)

x.add(88 * flag[22] + 23 * flag[13] + 18 * flag[14] + 77 * flag[9]+ 56 * flag[30]+ 79 * flag[2]+ 71 * flag[29]+ 95 * flag[28]+ 87 * flag[24]+ 62 * flag[16]+ 85 * flag[26]+ 43 * flag[20]+ 67 * flag[15]+ 97 * flag[8]+ 80 * flag[0]+ 23 * flag[3]+ 95 * flag[25]+ 82 * flag[21]+ 66 * flag[31]+ 5 * flag[4]+ 66 * flag[27]+ 25 * flag[12]+ 4 * flag[5]+ 12 * flag[7]+ 85 * flag[1]+ 10 * flag[6]+ 45 * flag[11]+ 28 * flag[18]+ 26 * flag[19] + 48 * flag[23] + 45 * flag[17] == 148888)

x.add(25 * flag[8] + 81 * flag[30]+ 21 * flag[6]+ 72 * flag[11]+ 48 * flag[18]+ 2 * flag[19]+ 42 * flag[10]+ 22 * flag[24]+ 99 * flag[2]+ 78 * flag[22]+ 83 * flag[12]+ 60 * flag[9]+ 59 * flag[13]+ 15 * flag[5]+ 25 * flag[20]+ 43 * flag[15]+ 56 * flag[28]+ 33 * flag[25]+ 71 * flag[23]+ 31 * flag[0]+ 95 * flag[3]+ 73 * flag[17]+ 86 * flag[14]+ 15 * flag[21]+ 61 * flag[7]+ 12 * flag[29]+ 95 * flag[26] + 13 * flag[1] + 100 * flag[16] + 11 * flag[4] + 79 * flag[27] == 138023)

x.add(37 * flag[28] + 62 * flag[25] + 42 * flag[18] + 53 * flag[27]+ 52 * flag[29]+ 70 * flag[22]+ 35 * flag[30]+ 50 * flag[16]+ 59 * flag[8]+ 75 * flag[10]+ 55 * flag[20]+ 23 * flag[0]+ 52 * flag[17]+ 47 * flag[3]+ 91 * flag[13]+ 46 * flag[7]+ 42 * flag[14]+ 79 * flag[26]+ 87 * flag[21]+ 30 * flag[6]+ 26 * flag[1]+ 57 * flag[31]+ 33 * flag[12]+ 51 * flag[9]+ 56 * flag[24]+ 59 * flag[11]+ 36 * flag[23]+ 88 * flag[4]+ 28 * flag[2] + 44 * flag[15] + 19 * flag[19] + 74 * flag[5] == 142299)

x.add(80 * flag[21]+ 43 * flag[31]+ 67 * flag[16]+ 55 * flag[13]+ 95 * flag[24]+ 46 * flag[28]+ 93 * flag[5]+ 75 * flag[20]+ 14 * flag[25]+ 24 * flag[26]+ 50 * flag[29]+ 70 * flag[15]+ 63 * flag[30]+ 77 * flag[23]+ 96 * flag[19]+ 66 * flag[11]+ 72 * flag[27]+ 94 * flag[4]+ 63 * flag[22]+ 69 * flag[3]+ 73 * flag[1]+ 60 * flag[7]+ 9 * flag[2]+ 39 * flag[17]+ 25 * flag[0]+ 49 * flag[14] + 48 * flag[8] + 86 * flag[9] + 72 * flag[10] + 23 * flag[18] + 21 * flag[6] == 155777)

x.add(25 * flag[24] + 11 * flag[22] + 27 * flag[11]+ 40 * flag[8]+ 53 * flag[15]+ 40 * flag[18]+ 56 * flag[3]+ 2 * flag[2]+ 32 * flag[4]+ 90 * flag[1]+ 54 * flag[16]+ 20 * flag[9]+ 86 * flag[17]+ 82 * flag[31]+ 43 * flag[25]+ 43 * flag[13]+ 86 * flag[21]+ 17 * flag[0]+ (flag[14] * 64)+ 6 * flag[30]+ 86 * flag[5]+ 15 * flag[7]+ 46 * flag[12]+ 21 * flag[26]+ 90 * flag[20]+ 19 * flag[6]+ 93 * flag[23]+ 31 * flag[27] + 62 * flag[29] + 21 * flag[19] + 42 * flag[10] == 117687)

x.add(89 * flag[21] + 100 * flag[13] + flag[27]+ 66 * flag[18]+ 40 * flag[17]+ 17 * flag[0]+ 27 * flag[19]+ 26 * flag[31]+ 57 * flag[24]+ 35 * flag[3]+ 80 * flag[1]+ 67 * flag[5]+ 85 * flag[6]+ 7 * flag[15]+ 93 * flag[8]+ 3 * flag[22]+ 77 * flag[12]+ 12 * flag[28]+ 4 * flag[2]+ 27 * flag[9]+ 53 * flag[25]+ 37 * flag[30]+ 43 * flag[23]+ 33 * flag[4]+ 39 * flag[26]+ 7 * flag[7]+ 75 * flag[10]+ 15 * flag[14] + 45 * flag[20] + 36 * flag[29] + 78 * flag[11] + 31 * flag[16] == 117383)

x.add(73 * flag[20] + 16 * flag[26] + 100 * flag[5] + 71 * flag[28] + 71 * flag[16]+ 4 * flag[1]+ 77 * flag[31]+ 83 * flag[2]+ 11 * flag[30]+ 53 * flag[19]+ 85 * flag[12]+ 67 * flag[13]+ 39 * flag[8]+ 45 * flag[24]+ 84 * flag[22]+ 99 * flag[14]+ 38 * flag[3]+ 29 * flag[4]+ 90 * flag[9]+ 61 * flag[18]+ 40 * flag[7]+ (flag[17] * 64)+ 9 * flag[25]+ 86 * flag[29]+ 80 * flag[21]+ 4 * flag[15]+ 96 * flag[23]+ 99 * flag[10]+ 40 * flag[27] + 4 * flag[0] + 56 * flag[11] == 155741)

x.add((flag[12] * 64) + 76 * flag[0] + 5 * flag[11] + 87 * flag[2]+ 86 * flag[24]+ 76 * flag[14]+ 38 * flag[23]+ 85 * flag[3]+ 71 * flag[22]+ 42 * flag[29]+ 85 * flag[30]+ 14 * flag[10]+ 17 * flag[13]+ 42 * flag[25]+ 11 * flag[19]+ 44 * flag[15]+ 21 * flag[4]+ 60 * flag[16]+ 28 * flag[6]+ 46 * flag[20]+ 25 * flag[9]+ 77 * flag[31]+ 21 * flag[8]+ 85 * flag[7]+ 36 * flag[1]+ 91 * flag[27]+ 21 * flag[28]+ 38 * flag[17] + 3 * flag[26] + 61 * flag[21] + 15 * flag[5] + 32 * flag[18] == 132804)

x.add(95 * flag[30] + 75 * flag[28] + 3 * flag[10] + 36 * flag[1]+ 60 * flag[3]+ 84 * flag[11]+ 19 * flag[26]+ 76 * flag[27]+ 86 * flag[16]+ 92 * flag[8]+ 96 * flag[14]+ 60 * flag[21]+ 23 * flag[4]+ 60 * flag[12]+ 50 * flag[23]+ 78 * flag[22]+ 45 * flag[9]+ 42 * flag[18]+ 10 * flag[2]+ 60 * flag[20]+ 24 * flag[24]+ 77 * flag[7]+ 41 * flag[6]+ 29 * flag[13]+ 33 * flag[5]+ 2 * flag[15]+ 33 * flag[29]+ 39 * flag[31] + 41 * flag[25] + 100 * flag[19] + 9 * flag[17] + 79 * flag[0] == 145568)

x.add(68 * flag[5] + 98 * flag[27] + 98 * flag[16] + 10 * flag[19] + 25 * flag[26]+ 98 * flag[24]+ 15 * flag[6]+ 50 * flag[18]+ 88 * flag[20]+ 74 * flag[11]+ 83 * flag[1]+ 86 * flag[21]+ 52 * flag[7]+ 39 * flag[10]+ 40 * flag[13]+ 82 * flag[28]+ 37 * flag[3]+ 45 * flag[0]+ 18 * flag[25]+ 2 * flag[29]+ 6 * flag[12]+ 78 * flag[31]+ 37 * flag[2]+ 57 * flag[23]+ 3 * flag[4]+ 59 * flag[8]+ 73 * flag[15]+ flag[22]+ 18 * flag[9]+ 35 * flag[14] + 20 * flag[17] + 54 * flag[30] == 130175)

x.add(60 * flag[10] + 50 * flag[12] + 30 * flag[29] + 90 * flag[19] + 68 * flag[23]+ 60 * flag[18]+ 93 * flag[20]+ 100 * flag[11]+ 98 * flag[14]+ 32 * flag[3]+ 15 * flag[21]+ 79 * flag[0]+ 6 * flag[24]+ 62 * flag[26]+ 96 * flag[6]+ 68 * flag[22]+ 9 * flag[7]+ 88 * flag[5]+ 18 * flag[27]+ 70 * flag[9]+ 96 * flag[25]+ 89 * flag[4]+ 14 * flag[31]+ 83 * flag[17]+ 19 * flag[15]+ 44 * flag[1]+ 96 * flag[8]+ 87 * flag[16]+ 48 * flag[2]+ 95 * flag[13] + 73 * flag[28] + 92 * flag[30] == 171986)

x.add(53 * flag[30] + 87 * flag[25] + 23 * flag[29] + 80 * flag[20] + 86 * flag[9]+ 20 * flag[7]+ 29 * flag[16]+ 31 * flag[14]+ 83 * flag[26]+ 11 * flag[4]+ 29 * flag[19]+ 82 * flag[13]+ 84 * flag[10]+ 70 * flag[1]+ 52 * flag[12]+ 40 * flag[6]+ 91 * flag[8]+ 6 * flag[17]+ 77 * flag[28]+ 56 * flag[5]+ 86 * flag[23]+ 63 * flag[31]+ 26 * flag[27]+ 19 * flag[22]+ 50 * flag[3]+ 15 * flag[15]+ 67 * flag[2]+ 37 * flag[24]+ 84 * flag[18] + 81 * flag[21] + 93 * flag[0] == 151676)

x.add(29 * flag[3] + 93 * flag[5] + 67 * flag[21] + 12 * flag[11]+ 82 * flag[24]+ 100 * flag[8]+ 29 * flag[26]+ 97 * flag[12]+ 32 * flag[6]+ 26 * flag[27]+ 46 * flag[19]+ 8 * (flag[25] + 9 * flag[0] + 2 * flag[17])+ 63 * flag[10]+ 39 * flag[29]+ 81 * flag[15]+ 51 * flag[13]+ 31 * flag[30]+ 49 * flag[4]+ 3 * flag[22]+ 26 * flag[28]+ 15 * flag[20]+ 89 * flag[2]+ 5 * flag[31]+ 47 * flag[18]+ 19 * flag[23]+ 98 * flag[9] + 15 * flag[16] + 49 * flag[1] == 128223)

x.add(13 * flag[14] + 73 * flag[19] + 99 * flag[7] + 76 * flag[12] + 84 * flag[25]+ 91 * flag[10]+ 67 * flag[22]+ 77 * flag[15]+ 23 * flag[26]+ 38 * flag[4]+ 3 * flag[31]+ 76 * flag[13]+ 50 * flag[0]+ 74 * flag[11]+ 45 * flag[28]+ 58 * flag[29]+ 39 * flag[5]+ 95 * flag[9]+ 26 * flag[16]+ 23 * flag[8]+ 28 * flag[24]+ 89 * flag[1]+ 88 * flag[18]+ 3 * flag[3]+ 59 * flag[20]+ 80 * flag[23]+ 49 * flag[17]+ 56 * flag[21]+ 32 * flag[27]+ 24 * flag[2] + 77 * flag[30] + 18 * flag[6] == 138403)

print x.check()

print x.model()

得到flag

[Asm] 纯文本查看 复制代码hgame{H4ppY#n3w@Y3AR%fr0M-oDiDi}

0x01 Say-Muggle-Code a.k.a. SMC

[Asm] 纯文本查看 复制代码int __cdecl main(int argc, const char **argv, const char **envp)

{

```
__int64 v3; // rax
char v4; // r12
bool v5; // r13
__int64 v6; // rax
__int64 v7; // rbx
bool v8; // al
__int64 v9; // rax
signed int i; // [rsp+Ch] [rbp-E4h]
char v12; // [rsp+10h] [rbp-E0h]
char v13; // [rsp+30h] [rbp-C0h]
char v14; // [rsp+50h] [rbp-A0h]
char v15; // [rsp+70h] [rbp-80h]
char v16; // [rsp+90h] [rbp-60h]
__int64 v17; // [rsp+B0h] [rbp-40h]
__int64 v18; // [rsp+B8h] [rbp-38h]
unsigned __int64 v19; // [rsp+C8h] [rbp-28h]
v19 = __readfsqword(0x28u);
std::__cxx11::basic_string,std::allocator>::basic_string(&v12, argv, envp);
std::operator<<<:char_traits>>(&std::cout, "hello muggle, please give me your flag: ");
std::operator>>,std::allocator>(&edata, &v12);
if ( std::__cxx11::basic_string,std::allocator>::length(&v12) != 39 )
{
v3 = std::operator<<<:char_traits>>(&std::cout, "your flag has a wrong length, muggle!");
std::ostream::operator<>);
exit(0);
}
v4 = 0;
std::__cxx11::basic_string,std::allocator>::substr(&v13, &v12, 0LL, 6LL);
v5 = 1;
if ( !(unsigned __int8)std::operator!=,std::allocator>(&v13, "hgame{") )
{
```

```
std::__cxx11::basic_string,std::allocator>::substr(&v14, &v12, 38LL, -1LL);
v4 = 1;
if ( !(unsigned __int8)std::operator!=,std::allocator>(&v14, "}") )
v5 = 0;
}
if ( v4 )
std::__cxx11::basic_string,std::allocator>::~basic_string(&v14);
std::__cxx11::basic_string,std::allocator>::~basic_string(&v13);
if ( v5 )
{
v6 = std::operator<<<:char_traits>>(&std::cout, "it's not even a valid flag, muggle!");
std::ostream::operator<>);
}
else
{
std::__cxx11::basic_string,std::allocator>::substr(&v15, &v12, 6LL, 16LL);
std::__cxx11::basic_string,std::allocator>::substr(&v16, &v12, 22LL, 16LL);
v17 = 0LL;
v18 = 0LL;
v17 = *(unsigned __int8 *)std::__cxx11::basic_string,std::allocator>::operator[](
&v15,
0LL);
for ( i = 1; i <= 15; ++i )
{
v7 = *(unsigned __int8 *)std::__cxx11::basic_string,std::allocator>::operator[](
&v15,
i - 1);
*((_BYTE *)&v17 + i) = *(_BYTE *)std::__cxx11::basic_string,std::allocator>::operator[](
&v15,
i) ^ v7;
}
```

```
v8 = (unsigned __int8)check1(&v15) ^ 1 || (unsigned __int8)check2(&v16, &v17) ^ 1;
if ( v8 )
v9 = std::operator<<<:char_traits>>(&std::cout, "your flag is good, but mine is better, muggle!");
else
v9 = std::operator<<<:char_traits>>(
&std::cout,
"wow, your flag is exactly the same as mine, congratulations, just submit it!");
std::ostream::operator<>);
std::__cxx11::basic_string,std::allocator>::~basic_string(&v16);
std::__cxx11::basic_string,std::allocator>::~basic_string(&v15);
}
std::__cxx11::basic_string,std::allocator>::~basic_string(&v12);
return 0;
}
```

先判断长度是不是39位 后边还判断了"hgame{"和"}"

随后是切割 把6-22位送入check1函数

[Asm] 纯文本查看 复制代码signed __int64 __fastcall check1(__int64 a1)

```
{
int v1; // eax
int i; // [rsp+1Ch] [rbp-14h]
for ( i = 0;
i < (unsigned __int64)std::__cxx11::basic_string,std::allocator>::length(a1);
++i )
{
v1 = *(char *)std::__cxx11::basic_string,std::allocator>::operator[](a1, i);
LOBYTE(v1) = v1 ^ 0xE9;
if ( v1 != (unsigned __int8)data1[i] )
return 0LL;
}
return 1LL;
}
```

异或比较 那么可以得出6-22位是781ef0676e13e541

再看check2函数

[Asm] 纯文本查看 复制代码bool __fastcall check2(__int64 a1, __int64 a2)

{

const char *v2; // rax

char dest[8]; // [rsp+10h] [rbp-20h]

__int64 v5; // [rsp+18h] [rbp-18h]

char v6; // [rsp+20h] [rbp-10h]

unsigned __int64 v7; // [rsp+28h] [rbp-8h]

v7 = __readfsqword(0x28u);

*(_QWORD *)dest = 0LL;

v5 = 0LL;

v6 = 0;

v2 = (const char *)std::__cxx11::basic_string,std::allocator>::c_str(a1);

strcpy(dest, v2);

mprotect(&encrypt, 0x200uLL, 7);

modify(&encrypt, 0x200uLL);

((void (__fastcall *)(char *, __int64))encrypt)(dest, a2);

return strcmp(dest, data2) == 0;

}

mprotect 函数修改属性

modify修改数值 就是SMC了

直接上IDC脚本

[Asm] 纯文本查看 复制代码#include

static dec(from) {

auto i, x;

auto q=123;

for (i=0; i <= 0x200; i=i+1) {

x = Byte(from+i);

x = (x^q);

q=q+1;

PatchByte(from+i,x);

}

}

得到加密函数

[Asm] 纯文本查看 复制代码_DWORD *__fastcall encrypt(__int64 a1, _DWORD *a2)

{

_DWORD *result; // rax

int v3; // [rsp+10h] [rbp-10h]

signed int i; // [rsp+14h] [rbp-Ch]

signed int j; // [rsp+18h] [rbp-8h]

v3 = 0;

for (i = 0; i <= 31; ++i)

{

result = (_DWORD *)2654435769LL;

v3 -= 1640531527;

for (j = 0; j <= 3; j += 2)

{

*(_DWORD *)(a1 + 4LL * j) = *(_DWORD *)(4LL * j + a1)

+ ((*(_DWORD *)(4 * (j + 1LL) + a1) + v3) ^ (16 * *(_DWORD *)(4 * (j + 1LL) + a1) + *a2) ^ ((*(_DWORD *)(4 * (j + 1LL) + a1) >> 5) + a2[1]));

result = (_DWORD *)(4 * (j + 1LL) + a1);

result += ((_DWORD *)(4LL * j + a1) + v3) ^ (16 * *(_DWORD *)(4LL * j + a1) + a2[2]) ^ ((*(_DWORD *)(4LL * j + a1) >> 5)

+ a2[3]);

}

}

return result;

}

捉重点

[Asm] 纯文本查看 复制代码result = (_DWORD *)2654435769LL;

v3 -= 1640531527;

目测是TEA加密

在看看前面的参数

发现是用原来的6-22位异或之后当做密钥 然后使用TEA算法加密22-38位 再比较

直接上脚本就行 这里修改了一位师傅的脚本

[Asm] 纯文本查看 复制代码https://sh1rker.github.io/2019/02/12/HGAME2019-Say-Muggle-Code-a-k-a-SMC/

[Asm] 纯文本查看 复制代码#include

#include

```
void decrypt (DWORD* v, DWORD* k) {
DWORD v0=v[0], v1=v[1], v2=v[2],v3=v[3], sum=0xC6EF3720, i;
DWORD delta=0x9e3779b9;
DWORD k0=k[0], k1=k[1], k2=k[2], k3=k[3];
for (i=0; i<32; i++) {
v3 -= ((v2*16) + k2) ^ (v2 + sum) ^ ((v2>>5) + k3);
v2 -= ((v3*16) + k0) ^ (v3 + sum) ^ ((v3>>5) + k1);
v1 -= ((v0*16) + k2) ^ (v0 + sum) ^ ((v0>>5) + k3);
v0 -= ((v1*16) + k0) ^ (v1 + sum) ^ ((v1>>5) + k1);
sum -= delta;
}
v[0]=v0; v[1]=v1; v[2]=v2; v[3]=v3;
}
int main(void)
{
DWORD key[4] = {0x54090f37,0x01065603, 0x02545301, 0x05015056};
DWORD flags[4] ={0xd240f52f,0x728cca9d,0xb6379fd3, 0xfba1a736};
//TEA_Decrypt(flags, key);
decrypt(flags, key);
for(int i=0; i<4; ++i)
{
printf("%x ", flags[i]);
}
return 0;
}
```

注意大小端问题后可以得到flag

[Asm] 纯文本查看 复制代码hgame{781ef0676e13e541d91debef62c1946f}

0x02 helloweb

题目下载下来后发现有三个文件 hello.js hello.html hello.wasm

第一次遇到 百度了wasm

发现是

WebAssembly是一种运行在现代网络浏览器中的新型代码并且提供新的性能特性和效果。

它设计的目的不是为了手写代码而是为诸如C、C++和Rust等低级源语言提供一个高效的编译目标。

WebAssembly是一门低级的类汇编语言。

它有一种紧凑的二进制格式，使其能够以接近原生性能的速度运行并且为诸如C++和Rust等拥有低级的内存模型语言提供了一个编译目标以便它们能够在网络上运行。

也就是说 你可以用C/C++来写网页

网上对于这方面逆向不是很多 即使用wabt翻译成c代码或者是wat文件格式阅读起来都非常麻烦

然后我决定自己来写一个类似的逆一下

先上官网安装好环境https://emscripten.org/

写一个C文件

[Asm] 纯文本查看 复制代码#include

#include

int main()

{

char s[20];

printf("Input your flag:");

scanf("%s", s);

if(strcmp(s, "flag{aaa}}")==0)

{

printf("good\n");

}

else

{

printf("wrong!\n");

}

return 0;

}

然后生成网页

[Asm] 纯文本查看 复制代码emcc hellos.c -s WASM=1 -o test.html

打开后发现界面简直跟原来题目一毛一样 而且html和js里边的代码其实差不多

22.png (142.13 KB, 下载次数: 0)

2019-3-6 21:49 上传

现在试一下逆自己写的代码

直接在浏览器里边调试 用wabt反汇编出来的.wat文件阅读代码中发现

[Asm] 纯文本查看 复制代码(export "_main" (func 19))

也就是说(func (;19;) (type 4) (result i32) 其实就是main函数 在这里下断点

看一下代码

[Asm] 纯文本查看 复制代码(func (;19;) (type 4) (result i32)

(local i32 i32 i32 i32 i32 i32 i32 i32 i32 i32)

global.get 18

local.set 9

global.get 18

i32.const 64

i32.add

global.set 18

global.get 18

global.get 19

i32.ge_s

if ;; label = @1

i32.const 64

call 0

end

local.get 9

i32.const 48

i32.add

local.set 7

local.get 9

i32.const 40

i32.add

local.set 6

local.get 9

i32.const 32

i32.add

local.set 5

local.get 9

i32.const 24

i32.add

local.set 4

local.get 9

local.set 1

i32.const 0

local.set 0

i32.const 2528

local.get 4

call 86

drop

local.get 5

local.get 1

i32.store

i32.const 2545

local.get 5

call 85

drop

local.get 1

i32.const 2548

call 84

local.set 2

local.get 2

i32.const 0

i32.eq

local.set 3

local.get 3

if ;; label = @1

i32.const 2559

local.get 6

call 86

drop

local.get 9

global.set 18

i32.const 0

return

else

i32.const 2565

local.get 7

call 86

drop

local.get 9

global.set 18

i32.const 0

return

end

动态调试可以发现

call $func86 应该就是printf了

call $func85 应该就是scanf函数

call $func84 应该就是strcmp函数了

还有一些有意思的代码

[Asm] 纯文本查看 复制代码get_local $var1

i32.const 2548

call $func84

const 2548 应该就是和输入的字符串对比的字符了 2548就是偏移了

333.png (13.33 KB, 下载次数: 0)

2019-3-6 22:00 上传

是字符串"flag{aaa}}"

好了 按照刚才我的思路去逆题目

先简单看看反汇编的C代码

[Asm] 纯文本查看 复制代码static u32 _main(void) {

u32 l0 = 0, l1 = 0, l2 = 0, l3 = 0, l4 = 0, l5 = 0, l6 = 0, l7 = 0,

l8 = 0, l9 = 0, l10 = 0, l11 = 0, l12 = 0, l13 = 0, l14 = 0, l15 = 0,

l16 = 0, l17 = 0, l18 = 0, l19 = 0, l20 = 0, l21 = 0, l22 = 0;

FUNC_PROLOGUE;

u32 i0, i1, i2;

u64 j1;

i0 = g18;

l22 = i0;

i0 = g18;//6128u

i1 = 80u;

i0 += i1;

g18 = i0;

i0 = g18;

i1 = g19;

i0 = (u32)((s32)i0 >= (s32)i1);

if (i0) {

i0 = 80u;

(*Z_envZ_abortStackOverflowZ_vi)(i0);

}

i0 = l22;

i1 = 64u;

```
i0 += i1;
l20 = i0;
i0 = l22;
i1 = 32u;
i0 += i1;
l1 = i0;
i0 = l22;
l12 = i0;
i0 = 0u;
l0 = i0;
i0 = 2107u;
i0 = f71(i0);//printf("Input your flag")
i0 = l20;
i1 = l1;
i32_store(Z_envZ_memory, (u64)(i0), i1);
i0 = 2130u;
i1 = l20;
i0 = f72(i0, i1);//Input
i0 = l12;
i1 = 1024u;
j1 = i64_load(Z_envZ_memory, (u64)(i1));
i64_store(Z_envZ_memory, (u64)(i0), j1);
i0 = l12;
i1 = 8u;
i0 += i1;
i1 = 1024u;
i2 = 8u;
i1 += i2;
j1 = i64_load(Z_envZ_memory, (u64)(i1));
i64_store(Z_envZ_memory, (u64)(i0), j1);
i0 = l12;
```

```
i1 = 16u;
i0 += i1;
i1 = 1024u;
i2 = 16u;
i1 += i2;
j1 = i64_load(Z_envZ_memory, (u64)(i1));
i64_store(Z_envZ_memory, (u64)(i0), j1);
i0 = l12;
i1 = 24u;
i0 += i1;
i1 = 1024u;
i2 = 24u;
i1 += i2;
i1 = i32_load16_s(Z_envZ_memory, (u64)(i1));
i32_store16(Z_envZ_memory, (u64)(i0), i1);
i0 = l12;
i1 = 26u;
i0 += i1;
i1 = 1024u;
i2 = 26u;
i1 += i2;
i1 = i32_load8_s(Z_envZ_memory, (u64)(i1));
i32_store8(Z_envZ_memory, (u64)(i0), i1);
i0 = 0u;
l13 = i0;
L1:
i0 = l13;
l14 = i0;
i0 = l14;
i1 = 26u;
i0 = (u32)((s32)i0 < (s32)i1);//长度判断
```

```
l15 = i0;
i0 = l15;
i0 = !(i0);
if (i0) {//超过长度就跳转
goto B2;
}
i0 = l13;//计数器
l16 = i0;
i0 = 2080u;
i1 = l16;
i0 += i1;
l17 = i0;
i0 = l17;
i0 = i32_load8_s(Z_envZ_memory, (u64)(i0));//弹出栈顶的值
l18 = i0;
i0 = l18;
i1 = 24u;
i0 <<= (i1 & 31);
i1 = 24u;
i0 = (u32)((s32)i0 >> (i1 & 31));
l19 = i0;
i0 = l13;//计数器
l2 = i0;
i0 = l1;
i1 = l2;
i0 += i1;
l3 = i0;
i0 = l3;
i0 = i32_load8_s(Z_envZ_memory, (u64)(i0));
l4 = i0;
i0 = l4;
```

```
i1 = 24u;
i0 <<= (i1 & 31);
i1 = 24u;
i0 = (u32)((s32)i0 >> (i1 & 31));
l5 = i0;
i0 = l5;
i1 = l19;
i0 ^= i1;
l6 = i0;
i0 = l6;
i1 = 255u;
i0 &= i1;
l7 = i0;
i0 = l3;
i1 = l7;
i32_store8(Z_envZ_memory, (u64)(i0), i1);
i0 = l13;
l8 = i0;
i0 = l8;
i1 = 1u;
i0 += i1;
l9 = i0;
i0 = l9;
l13 = i0;
goto L1;
B2:;
i0 = l1;
i1 = l12;
i0 = f34(i0, i1);
l10 = i0;
i0 = l10;
```

```
i1 = 0u;
i0 = i0 != i1;
l11 = i0;
i0 = l11;
if (i0) {
i0 = 2144u;
i0 = f71(i0);//printf
i0 = l22;
g18 = i0;
i0 = 0u;
goto Bfunc;
} else {
i0 = 2135u;
i0 = f71(i0);//printf
i0 = l22;
g18 = i0;
i0 = 0u;
goto Bfunc;
}
UNREACHABLE;
Bfunc:;
FUNC_EPILOGUE;
return i0;
}
```

通过调试 可以发现f71是printf函数

f72是input函数

f34是strcmp函数

从整体来看 应该是输入字符串 然后跟什么东西异或再比较

在循环体中 发现了i0 = 2080u;等字眼 猜测2080存在异或需要的东西

11.png (43.62 KB, 下载次数: 1)

2019-3-6 22:12 上传

上面说过f34是strcmp函数 那么可以在这个函数前面下断点 当他们参数入栈的时候截取到字符地址 反推得到flag

最终得到

444.png (60.44 KB, 下载次数: 0)

2019-3-6 22:17 上传

第四周

0x00 real

题目提示real 还有main有个加密很复杂

[Asm] 纯文本查看 复制代码__int64 __fastcall main(__int64 a1, char **a2, char **a3)

{

__int64 v3; // r14

__int64 v4; // rcx

__int64 v5; // rdx

char flag; // [rsp+30h] [rbp-C0h]

unsigned __int64 v8; // [rsp+B8h] [rbp-38h]

v8 = __readfsqword(0x28u);

memset(&flag, 0, 0x80uLL);

puts("Hello!");

puts("Please input your flag:");

fgets(&flag, 50, stdin);

v3 = atoll(&flag);

v4 = sub_400C5C(0x3737373737373737uLL, v3, v3 >> 63, 0xF78D5C4752F8CCDBLL, -1LL);

if (v5 | v4 ^ 0x169A25615637D3A4LL)

printf("failed", 0xF78D5C4752F8CCDBLL, -1LL);

else

printf("success!", 0xF78D5C4752F8CCDBLL, -1LL);

return 0LL;

}核心代码应该不在这里

查找init函数 调用main函数之前会先调用这个函数

发现init调用的函数中 有一个很可疑

[Asm] 纯文本查看 复制代码unsigned __int64 sub_400976()

{

int v0; // eax

FILE *stream; // [rsp+0h] [rbp-1080h]

FILE *v3; // [rsp+8h] [rbp-1078h]

char modes[3]; // [rsp+10h] [rbp-1070h]

char v5; // [rsp+20h] [rbp-1060h]

char v6; // [rsp+21h] [rbp-105Fh]

char v7; // [rsp+22h] [rbp-105Eh]

char v8; // [rsp+23h] [rbp-105Dh]

char name; // [rsp+30h] [rbp-1050h]

char v10; // [rsp+31h] [rbp-104Fh]

char v11; // [rsp+32h] [rbp-104Eh]

char v12; // [rsp+33h] [rbp-104Dh]

char v13; // [rsp+34h] [rbp-104Ch]

char v14; // [rsp+35h] [rbp-104Bh]

char v15; // [rsp+36h] [rbp-104Ah]

char v16; // [rsp+37h] [rbp-1049h]

char v17; // [rsp+38h] [rbp-1048h]

char v18; // [rsp+39h] [rbp-1047h]

char v19; // [rsp+3Ah] [rbp-1046h]

char path; // [rsp+40h] [rbp-1040h]

char v21; // [rsp+41h] [rbp-103Fh]

char v22; // [rsp+42h] [rbp-103Eh]

char v23; // [rsp+43h] [rbp-103Dh]

char v24; // [rsp+44h] [rbp-103Ch]

char v25; // [rsp+45h] [rbp-103Bh]

char v26; // [rsp+46h] [rbp-103Ah]

char v27; // [rsp+47h] [rbp-1039h]

char v28; // [rsp+48h] [rbp-1038h]

```
char v29; // [rsp+49h] [rbp-1037h]
char v30; // [rsp+4Ah] [rbp-1036h]
char v31; // [rsp+4Bh] [rbp-1035h]
char v32; // [rsp+4Ch] [rbp-1034h]
char v33; // [rsp+4Dh] [rbp-1033h]
char v34; // [rsp+4Eh] [rbp-1032h]
char command; // [rsp+50h] [rbp-1030h]
char v36; // [rsp+51h] [rbp-102Fh]
char v37; // [rsp+52h] [rbp-102Eh]
char v38; // [rsp+53h] [rbp-102Dh]
char v39; // [rsp+54h] [rbp-102Ch]
char v40; // [rsp+55h] [rbp-102Bh]
char v41; // [rsp+56h] [rbp-102Ah]
char v42; // [rsp+57h] [rbp-1029h]
char v43; // [rsp+58h] [rbp-1028h]
char v44; // [rsp+59h] [rbp-1027h]
char v45; // [rsp+5Ah] [rbp-1026h]
char v46; // [rsp+5Bh] [rbp-1025h]
char v47; // [rsp+5Ch] [rbp-1024h]
char v48; // [rsp+5Dh] [rbp-1023h]
char v49; // [rsp+5Eh] [rbp-1022h]
char v50; // [rsp+5Fh] [rbp-1021h]
char v51; // [rsp+60h] [rbp-1020h]
char v52; // [rsp+61h] [rbp-101Fh]
char v53; // [rsp+62h] [rbp-101Eh]
char v54; // [rsp+63h] [rbp-101Dh]
char v55; // [rsp+64h] [rbp-101Ch]
char v56; // [rsp+65h] [rbp-101Bh]
char v57; // [rsp+66h] [rbp-101Ah]
char v58; // [rsp+67h] [rbp-1019h]
char v59; // [rsp+68h] [rbp-1018h]
```

```
char v60; // [rsp+69h] [rbp-1017h]
char v61; // [rsp+6Ah] [rbp-1016h]
char v62; // [rsp+6Bh] [rbp-1015h]
char v63; // [rsp+6Ch] [rbp-1014h]
char buf; // [rsp+70h] [rbp-1010h]
unsigned __int64 v65; // [rsp+1078h] [rbp-8h]
v65 = __readfsqword(0x28u);
modes[0] = 114;
modes[1] = 98;
modes[2] = 0;
name = 46;
v10 = 47;
v11 = 46;
v12 = 114;
v13 = 101;
v14 = 97;
v15 = 108;
v16 = 46;
v17 = 115;
v18 = 111;
v19 = 0;
v5 = 119;
v6 = 98;
v7 = 43;
v8 = 0;
command = 76;
v36 = 68;
v37 = 95;
v38 = 80;
v39 = 82;
v40 = 69;
```

```
v41 = 76;
v42 = 79;
v43 = 65;
v44 = 68;
v45 = 61;
v46 = 46;
v47 = 47;
v48 = 46;
v49 = 114;
v50 = 101;
v51 = 97;
v52 = 108;
v53 = 46;
v54 = 115;
v55 = 111;
v56 = 32;
v57 = 46;
v58 = 47;
v59 = 114;
v60 = 101;
v61 = 97;
v62 = 108;
v63 = 0;
path = 47;
v21 = 112;
v22 = 114;
v23 = 111;
v24 = 99;
v25 = 47;
v26 = 115;
v27 = 101;
```

```
v28 = 108;
v29 = 102;
v30 = 47;
v31 = 101;
v32 = 120;
v33 = 101;
v34 = 0;
if ( access(&name, 0) )
{
readlink(&path, &buf, 0x1000uLL);
stream = fopen(&buf, modes);
fseek(stream, -12936LL, 2); // 取出.so库
v3 = fopen(&name, &v5);
while ( !feof(stream) )
{
v0 = fgetc(stream);
fputc(v0, v3);
}
fclose(stream);
fclose(v3);
system(&command);
exit(0);
}
return __readfsqword(0x28u) ^ v65;
}
```

这代码的意思是

从源文件读取倒数12936个字节出来当作文件运行

也就是说 源文件可能有两个elf文件 用winhex一查果然如此

用winhex抠出来在分析

puts函数很可疑

[Asm] 纯文本查看 复制代码

```
unsigned __int64 puts()
{
```

```
char *v0; // rax
int i; // [rsp+Ch] [rbp-1014h]
char buf[4104]; // [rsp+10h] [rbp-1010h]
unsigned __int64 v4; // [rsp+1018h] [rbp-8h]
v4 = __readfsqword(0x28u);
if ( aaaaa )
{
printf("Please input your flag:");
putchar(10);
j_encrypt(10LL);
exit(0);
}
memset(buf, 0, 0x1000uLL);
getcwd(buf, 0x1000uLL);
v0 = &buf[strlen(buf)];
*(_QWORD *)v0 = 8299690328860012079LL;
*((_WORD *)v0 + 4) = 111;
unlink(buf);
printf("Hello!");
putchar(10);
mprotect(&dword_0, 0x1000uLL, 7);
for ( i = 0; i <= 309; ++i )
*((_BYTE *)encrypt + i + 20) ^= i;
aaaaa = 1;
return __readfsqword(0x28u) ^ v4;
}
```

又是smc 可以修改上面的代码拿来用

查看解密后的encrypt代码

[Asm] 纯文本查看 复制代码unsigned __int64 encrypt()

```
{
void *v0; // ST00_8
```

```
unsigned int v1; // eax
__int64 v2; // ST00_8
unsigned int v3; // eax
char s1[128]; // [rsp+10h] [rbp-190h]
char v6; // [rsp+90h] [rbp-110h]
char s2; // [rsp+110h] [rbp-90h]
unsigned __int64 v8; // [rsp+198h] [rbp-8h]
v8 = __readfsqword(0x28u);
v0 = malloc(0x100uLL);
memset(s1, 0, sizeof(s1));
s1[0] = 67;
s1[1] = 36;
s1[2] = 229;
s1[3] = 161;
s1[4] = 197;
s1[5] = 29;
s1[6] = 114;
s1[7] = 210;
s1[8] = 40;
s1[9] = 239;
s1[10] = 190;
s1[11] = 234;
s1[12] = 165;
s1[13] = 151;
s1[14] = 68;
s1[15] = 96;
s1[16] = 217;
s1[17] = 15;
s1[18] = 44;
s1[19] = 111;
s1[20] = 94;
```

```
s1[21] = 38;
s1[22] = 179;
s1[23] = 10;
s1[24] = 252;
s1[25] = 212;
s1[26] = 179;
memset(&v6, 0, 0x80uLL);
memset(&s2, 0, 0x80uLL);
scanf("%50s", &v6, &s2, v0);
v1 = strlen("hgame!@#");
unk_EB1(v2, "hgame!@#", v1);
v3 = strlen(&v6);
unk_F91(v2, (__int64)&v6, v3, (__int64)&s2);
if ( !strcmp(s1, &s2) )
printf("success!", &s2);
else
printf("failed", &s2);
putchar(10);
return __readfsqword(0x28u) ^ v8;
}
```

再看unk_EB1函数

[Asm] 纯文本查看 复制代码

```
__int64 __fastcall sub_EB1(__int64 a1, __int64 a2, int a3)
{
__int64 result; // rax
unsigned int v4; // eax
char v5; // ST24_1
signed int i; // [rsp+18h] [rbp-Ch]
signed int j; // [rsp+18h] [rbp-Ch]
int v8; // [rsp+1Ch] [rbp-8h]
for ( i = 0; i <= 255; ++i )
{
```

```
result = i;
*(_BYTE *)(a1 + i) = i;
}
v8 = 0;
for ( j = 0; j <= 255; ++j )
{
v4 = (unsigned int)((*(unsigned __int8 *)(a1 + j) + v8 + *(unsigned __int8 *)(j % a3 + a2)) >> 31) >> 24;
v8 = (unsigned __int8)(v4 + *(_BYTE *)(a1 + j) + v8 + *(_BYTE *)(j % a3 + a2)) - v4;
v5 = *(_BYTE *)(a1 + j);
*(_BYTE *)(a1 + j) = *(_BYTE *)(a1 + v8);
result = v8;
*(_BYTE *)(a1 + v8) = v5;
}
return result;
}
```

很明显了 RC4加密 密钥是"hgame!@#"

s字符串提取出来 base64加密后直接在线解密

直接反推得到flag

555.png (90.13 KB, 下载次数: 0)

2019-3-6 22:29 上传

0x01 happyVM

不懂咋讲 跟着流程图发现规律

印象中这道题也就两层异或然后比较

代码

[Asm] 纯文本查看 复制代码i = [0x84, 0x83, 0x9D, 0x91, 0x81, 0x97, 0xD7, 0xBE, 0x43, 0x72, 0x61, 0x73, 0x73, 0x0C, 0x6A, 0x70

,0x73, 0x11, 0x48, 0x2C, 0x34, 0x33, 0x31, 0x36, 0x23, 0x34, 0x3E, 0x5C, 0x23, 0x4E, 0x17, 0x11,0x19, 0x59]

flag = ""

flags = ""

cout = 0x32

```
i = i[::-1]
for q in i:
flag += chr(q^cout)
cout += 3
cout = 0x16
for q in flag:
flags+=chr(ord(q)^cout)
cout += 3
print flags[::-1]
```

[Asm] 纯文本查看 复制代码hgame{3Z_VM_W0NT_5T0P_UR_PR0GR355}