ICS I reverselab 报告

中国人民大学 sheriyuo

摘 要

RUC 2023 计算机系统基础 I reverselab 的解题思路和实现。
选做题部分挺有意思的，只不过 9,10 懒得卷了。
使用 IDA Pro 的 F5 是从 bomblab 到 bomblab 2.0 的伟大进化。

1 easystr

```
mov     [rsp+68h+var_48], 0
jmp     short loc_140001089
loc_140001089:
cmp     [rsp+68h+var_48], 14h
jge     short loc_1400010DE
```

结合 `loc_14000107F` 的 `inc eax` 可以发现此为循环结构，`[rsp+68h+var_48]` 为 `i`。

```
movsxd  rax, [rsp+68h+var_48]
lea     rcx, Str2       ; "Welcome_to_the_reverse_world!"
movsx   eax, byte ptr [rcx+rax]
cmp     eax, 6Ch ; 'l'
jnz     short loc_1400010B7
movsxd  rax, [rsp+68h+var_48]
lea     rcx, Str2       ; "Welcome_to_the_reverse_world!"
mov     byte ptr [rcx+rax], 31h ; '1'
jmp     short loc_1400010DC
```

循环内，判断 `*(Str2 + i)` 是否等于 `'l'`，如果等于，将其替换为 `'1'`。`Str2` 初值为 `"Welcome_to_the_reverse_world!"`。

```
loc_1400010B7:
movsxd  rax, [rsp+68h+var_48]
lea     rcx, Str2       ; "Welcome_to_the_reverse_world!"
movsx   eax, byte ptr [rcx+rax]
cmp     eax, 6Fh ; 'o'
jnz     short loc_1400010DC
movsxd  rax, [rsp+68h+var_48]
lea     rcx, Str2       ; "Welcome_to_the_reverse_world!"
mov     byte ptr [rcx+rax], 30h ; '0'
```

若 *(Str2 + i) 不等于 'l'，判断其是否等于 'o'，如果等于，将其替换为 '0'。执行 for(i = 0; i < 20; i++) 的循环，满足 jge 的条件时跳转。循环操作后的 Str2 为 "We1c0me_t0_the_reverse_world!"。

```
loc_1400010DE:
lea     rax, [rsp+68h+Str1]
mov     rdi, rax
xor     eax, eax
mov     ecx, 28h ; '('
rep stosb
lea     rcx, Buffer     ; "input the flag:"
call    cs:puts
mov     r8d, 28h ; '('
lea     rdx, [rsp+68h+Str1]
lea     rcx, aS         ; "%s"
call    sub_140001170
lea     rdx, Str2       ; "Welcome_to_the_reverse_world!"
lea     rcx, [rsp+68h+Str1] ; Str1
call    strcmp
test    eax, eax
jnz     short loc_140001137
lea     rcx, aRight     ; "right!"
call    cs:puts
jmp     short loc_140001144
```

循环结束后，首先 puts("input the flag:")，随后调用函数 sub_140001170 并压栈 "%s"，结合程序运行即可发现是在执行 scanf("%s", Str1)。随后调用 strcmp 函数，在 if(!strcmp(Str1, Str2)) 时 puts("right!")。

那么输入的 flag 即为 "We1c0me_t0_the_reverse_world!"。

2 xorrr

```
lea     rax, aX1j3y5a7u9t ; "X1j3y5a7u9t;`=|"
mov     [rsp+58h+var_28], rax
lea     rax, a5w7n9LNA  ; ";5w7n9 ;l=n?)A-"
mov     [rsp+58h+var_20], rax
lea     rax, aS9JAxchej ; "s9?;}=j?|AxChEj"
mov     [rsp+58h+var_18], rax
mov     [rsp+58h+var_38], 0
jmp     short loc_1400010C2
loc_1400010C2:
cmp     [rsp+58h+var_38], 18h
jge     short loc_14000112D
```

写入 3 个字符串，开始循环，枚举 `for(i = 0; i < 24; i++)`。

```
mov     eax, [rsp+58h+var_38]
cdq
mov     ecx, 3
idiv    ecx
mov     eax, edx
cdqe
mov     [rsp+58h+var_30], rax
```

首先提出 `i`，将 `i % 3` 的结果存储在 `[rsp+58h+var_30]` 中。

```
mov     eax, [rsp+58h+var_38]
cdq
mov     ecx, 3
idiv    ecx
shl     eax, 1
cdqe
```

再计算 `(i / 3) << 1` 的值，结果存储在 `rax` 中。

```
mov     rcx, [rsp+58h+var_30]
mov     rcx, [rsp+rcx*8+58h+var_28]
movsx   eax, byte ptr [rcx+rax]
mov     ecx, [rsp+58h+var_38]
```

```
add     ecx, 8
xor     eax, ecx
movsxd  rcx, [rsp+58h+var_38]
lea     rdx, byte_140003040
movsx   ecx, byte ptr [rdx+rcx]
sub     eax, ecx
cmp     eax, 2
jz      short loc_14000112B
```

将 `[rsp+i*8+58h+var_28]` 加载到 `rcx` 中，再 `movsx eax, byte ptr [rcx+rax]`，得到第 `i % 3` 个字符串的第 `(i / 3) << 1` 个字符。

将字符值异或上 `(i + 8)`，再减去读入字符串的第 `i` 个字符，如果结果不为 2，则跳转到 `break`。

按照如下代码构造字符串即可，结果为 `"N0w_y0u_kn0w_what_x0r_1s"`。

```
__int64 s[3];
s[0] = (__int64)"X1j3y5a7u9t;`=|";
s[1] = (__int64)";5w7n9 ;l=n?)A-";
s[2] = (__int64)"s9?;}=j?|AxChEj";
for(int i = 0; i < 24; i++) {
    char c = ((i + 8) ^ *(char *)(s[i % 3] + i / 3 * 2)) - 2;
    putchar(c);
}
```

3 maze

从此题开始，使用 IDA Pro 的反编译功能。

`v3` 的类型是 `FILE*`，可以推测出其为 `stdin`。读入的字符串 `Buffer` 长度应该为 10，`while` 遍历每一位 `v9 = Buffer[v10] - 97`。

如果 `v9 == 0`，`v11--`，`goto LABEL_11`；如果 `v9 == 3`，`v11++`，`goto LABEL_11`；如果 `v9 == 18`，`v11 += 5`，`goto LABEL_11`；如果 `v9 == 22`，`v11 -= 5`，`goto LABEL_11`。

`LABEL_11` 中，如果 `v11 > 0x18` 或 `byte_402180[v11] == 48`，跳转到 `puts("Wrong")`；否则 `++v10`，继续循环。

如果 `v9` 为其他值，也会跳转到 `puts("Wrong")`。

`if(byte_402180[v11] == 42) puts("Right! The flag is md5(your input)")` 在循环结束后执行，访问地址得到 `char byte_402180[] = "#11000011000001000110*1100"`，采用以下 dfs 代码求解。

```
vector<string>ans[0x19];
char byte_402180[] = "#1100001100001000110*1100";
void dfs(int i, int v11, string s) {
    if(v11 > 0x18 || v11 < 0 || byte_402180[v11] == 48) return;
    if(i == 10) {
        ans[v11].push_back(s);
        return;
    }
    dfs(i + 1, v11 - 1, s + 'a');
    dfs(i + 1, v11 + 1, s + 'd');
    dfs(i + 1, v11 + 5, s + 's');
    dfs(i + 1, v11 - 5, s + 'w');
}
int main() {
    dfs(0, 0, "");
    for(int i = 0; i <= 0x18; i++)
        if(ans[i].size())
            cout << i << ":" << ans[i].size() << ":" << ans[i].front() << endl;
}
```

求出 v1 == 20 有唯一解，答案正确，输入为 "ddsdssasaa"。

md5 后得到 flag 为 "0e6321aa4d31bfc66b83c0406885ce86"。

4 array

要求读入的字符串 byte_4033FC 长度为 20，且 sub_401080(byte_4033FC) 返回 1，条件为 byte_403000[*(char *)(i + byte_4033FC)] 等于 byte_403080[i]。

char byte_403000[4] = {0xff, 0xff, 0xff, 0xff};

char byte_403080[] = "82=7#+1;?50:9&?9=+%!";

模拟带 align 的内存溢出即可，代码如下。

```
#include <stdio.h>
char byte_403000[4] = {0xff, 0xff, 0xff, 0xff};
unsigned int dword_403004 = 1;
long long x = 0;
unsigned int dword_403010 = 1;
unsigned int __security_cookie = 0xBB40E64E;
unsigned int dword_403018 = 0x44BF19B1;
unsigned int dword_40301C = 1;
char aZyxwvutsrqponm[] = "~}|{zyxwvutsrqponmlkjihgfedcba`_^]\\[ZYXWVUTSRQPONMLKJIHGFEDCBA@
    ?>=<;:9876543210/.-,+*)('&%$#\"! ";
char byte_403080[] = "82=7#+1;?50:9&?9=+%!";
unsigned int _x = 0;
unsigned int dword_403098 = 0;
char unk_40309C[4] = {0, 0, 0, 0};
char byte_4030A0 = 0;
```

```
char byte_4030A1 = 0;
char __x[] = {0, 0};
int main() {
    for(int i = 0; i < 20; i++)
    for(int j = 0; j <= 0xff; j++) {
        char c = j;
        if(byte_403000[c] == byte_403080[i]) {
            putchar(c);
            break;
        }
    }
}
```

flag 为 "flag{smc_index_easy}"。

5 rome

char Destination[] = "Zxb3xo_qe4_Dob@q"，读入长度为 16 的字符串 Str，按照以下规则构造 char* v5:

- 如果 Str[i] 不是小写或大写字母，v5[i] = Str[i];
- 如果 Str[i] 是小写字母，v5[i] = 'a' + (Str[i] - 'd' + 26) % 26;
- 如果 Str[i] 是大写字母，v5[i] = 'A' + (Str[i] - 'D' + 26) % 26。

所以反向构造即可，答案为 "Cae3ar_th4_Gre@t"。

```
#include <stdio.h>
char Destination[] = "Zxb3xo_qe4_Dob@q";
int main() {
    for(int i = 0; i < 16; i++) {
        char c = Destination[i];
        if(c >= 'A' && c <= 'Z')
            putchar('A' + (c - 62 + 26) % 26);
        else if(c >= 'a' && c <= 'z')
            putchar('a' + (c - 94 + 26) % 26);
        else
            putchar(c);
    }
}
```

6 equation

输入一个长度为 32 的字符串 Str,取前 4 个字符进行一系列操作得到 dword_1400040A8，要求全 0，于是可以枚举反向构造出前 4 个字符，得到前 4 位为 "QTEM"。

```
int a[6], v6;
char Str[4] = "";
void check() {
    memset(a, 0, sizeof(a));
    a[4] += 16; a[3] += Str[3] * a[4];
    a[4] += 3; a[3] += Str[2] * a[4];
    a[4] -= 10; a[0] += Str[3] * a[4];
    a[4] -= 2; a[1] += Str[2] * a[4];
    a[4] += 13; a[0] += Str[1] * a[4];
    a[4] -= 8; a[2] += Str[1] * a[4];
    a[4] -= 7; a[2] -= 3481;
    a[4] += 3; a[1] -= 2422;
    a[4] += 9; a[0] += Str[2] * a[4];
    a[4] -= 2; a[2] += Str[2] * a[4];
    a[4] -= 6; a[0] -= 4518;
    a[4] += 7; a[3] -= 5006;
    a[4] -= 9; a[1] += Str[0] * a[4];
    a[4] += 5; a[0] += Str[0] * a[4];
    v6 = a[4] + 1;
    a[4] = v6; a[2] += Str[0] * v6;
    a[4] -= 8; a[2] += Str[3] * a[4];
    a[4] += 14; a[3] += Str[0] * a[4];
    a[4] -= 11; a[1] += Str[3] * a[4];
    a[4] += 3; a[3] += Str[1] * a[4];
    a[4] -= 2; a[1] += Str[1] * a[4];
}
int main() {
    for(int i = 0; i < 128; i++)
    for(int j = 0; j < 128; j++)
    for(int k = 0; k < 128; k++)
    for(int l = 0; l < 128; l++) {
        Str[0] = i, Str[1] = j, Str[2] = k, Str[3] = l;
        check();
        if(!a[0] && !a[1] && !a[2] && !a[3])
            puts(Str);
    }
}
```

要求 (Str[j % 4] ^ Str[j + 4]) 等于 *((char *)&qword_140004040 + 2 * j)，再次反向构造即可，需要在 x64 下运行。flag 为 "QTEM5D91sCjyBGNvNOJMOQ7q3KsINzwr"。

```
int dword_1400040A8[6], v6;
long long qword_140004040;
char Str[35] = "QTEM";
int main() {
```

```
    qword_140004040 = 0x7C007C00100064;
    *(&qword_140004040 + 1) = 0x34002F00170022;
    *(&qword_140004040 + 2) = 0x3B000B00130013;
    *(&qword_140004040 + 3) = 0xF001B001F;
    *(&qword_140004040 + 4) = 0x3C00720005001E;
    *(&qword_140004040 + 5) = 0x40036001F0062;
    *(&qword_140004040 + 6) = 0x3F0032002E001F;
    *(&qword_140004040 + 7) = 0;
    for(int i = 0; i < 28; i++)
        Str[i + 4] = Str[i % 4] ^ *((char *)&qword_140004040 + 2 * i);
    puts(Str);
}
```

7 click

给定一个用 Qt 写的 `click.exe`，点击 1000 次即可获得 flag `"abcde-12345-ghijkl"`，but is it true?

利用 IDA Pro 查看 Strings 表，找到 `"You got a flag now, but is it true?"` 所在交叉引用，发现调用了函数 `QByteArray::fromBase64`。

Strings 表中，位于其地址之前有 5 个字符串，对位于 `.rdata:0000000140005658` 的字符串 `"UXRmdW4tMTAwODYtR1VJdG9v"` 进行 base64 解密得到 flag `"Qtfun-10086-GUItoo"`。

8 junkcode

利用 Debugger 单步调试，在 `scanf` 后，先调用了 `sub_141160` 返回一个字符串结果，然后调用了多次 `strncpy` 和 `sub_141000`，传参分别为 `v4=11,a3=1`，`v4=11,a3=3`，`v4=11,a3=5`，`v4=11,a3=7`。

随后，`"P1Ekb1UxW9ErWC6ZVUKiKgMaLSEgS5gpyZOrSQG3tP8g"` 入栈，调用 `strcmp`，随后判断 `"wrong!"`，应是比较修改后的字符串是否相等。

于是可以反向构造出调用 `sub_141000` 之前的字符串结果。

```
char Str[] = "P1Ekb1UxW9ErWC6ZVUKiKgMaLSEgS5gpyZOrSQG3tP8g";
int v4 = 44;
for (int i = 0; i < v4; ++i) {
    char c;
    int a3 = 1 + 2 * (i / 11);
    if (Str[i] < 65 || Str[i] > 90) {
        if (Str[i] < 97 || Str[i] > 122) {
            if (Str[i] < 48 || Str[i] > 57)
            c = Str[i];
            else
```

```
            c = (Str[i] - 48 + a3) % 10 + 48;
        } else {
            int j = (Str[i] - 97 + a3) % 26;
            c = j + 97;
        }
    } else {
        int k = (Str[i] - 65 + a3) % 26;
        c = k + 65;
    }
    putchar(c);
}
```

得 "Q2Flc2VyX0FuZF9CYXNlNjRfQXJlX0ludGVyZXN0aW5n"，问题来到 sub_141160 对输入的字符串进行了什么操作。

在单步调试的过程中，发现出现了字符 '='，结合循环后的判断，合理推测其为 base64，于是解码得到 flag "Caeser_And_Base64_Are_Interesting"。