




南邮CTF逆向 ReadAsm2 writeup

原创

[dittozz](#)  于 2018-11-19 17:47:50 发布  575  收藏 1

分类专栏: [逆向](#) 文章标签: [逆向](#) [ctf](#) [汇编基础](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/qq_43394612/article/details/84257246

版权



[逆向](#) 专栏收录该内容

3 篇文章 0 订阅

订阅专栏

菜鸡刚学汇编, 找道逆向题练练手, 详细写下怎么解题的。由于是菜鸡, 写的很啰嗦。

主函数如下。

```
int main(int argc, char const *argv[])
{
    char input[] = {0x0, 0x67, 0x6e, 0x62, 0x63, 0x7e, 0x74,
0x62, 0x69, 0x6d,
                    0x55, 0x6a, 0x7f, 0x60, 0x51, 0x66, 0x63,
0x4e, 0x66, 0x7b,
                    0x71, 0x4a, 0x74, 0x76, 0x6b, 0x70, 0x79,
0x66 , 0x1c};
    func(input, 28);
    printf("%s\n",input+1);
    return 0;
}
```

https://blog.csdn.net/qq_43394612

函数汇编代码如下

```
0000000004004e6 <func>:
4004e6: 55                push   rbp
4004e7: 48 89 e5          mov    rbp, rsp
4004ea: 48 89 7d e8       mov    QWORD PTR [rbp-0x18], rdi
4004ee: 89 75 e4          mov    DWORD PTR [rbp-0x1c], esi
4004f1: c7 45 fc 01 00 00 00 mov    DWORD PTR [rbp-0x4], 0x1
4004f8: eb 28            jmp    400522 <func+0x3c>
4004fa: 8b 45 fc          mov    eax, DWORD PTR [rbp-0x4]
4004fd: 48 63 d0          movsxd rdx, eax
400500: 48 8b 45 e8       mov    rax, QWORD PTR [rbp-0x18]
400504: 48 01 d0          add    rax, rdx
400507: 8b 55 fc          mov    edx, DWORD PTR [rbp-0x4]
40050a: 48 63 ca          movsxd rcx, edx
40050d: 48 8b 55 e8       mov    rdx, QWORD PTR [rbp-0x18]
400511: 48 01 ca          add    rdx, rcx
400514: 0f b6 0a         movzx  ecx, BYTE PTR [rdx]
400517: 8b 55 fc          mov    edx, DWORD PTR [rbp-0x4]
40051a: 31 ca            xor    edx, ecx
40051c: 88 10            mov    BYTE PTR [rax], dl
40051e: 83 45 fc 01      add    DWORD PTR [rbp-0x4], 0x1
400522: 8b 45 fc          mov    eax, DWORD PTR [rbp-0x4]
400525: 3b 45 e4          cmp    eax, DWORD PTR [rbp-0x1c]
400528: 7e d0            jle   4004fa <func+0x14>
40052a: 90                nop
40052b: 5d                pop    rbp
40052c: c3                ret
```

https://blog.csdn.net/qq_43394612

```
4004e6: 55                push   rbp
4004e7: 48 89 e5          mov    rbp, rsp
```

前两句 push rbp 将rbp压栈，再将 rsp的值给rbp，提升堆栈的。

```

4004ea: 48 89 7d e8      mov     QWORD PTR [rbp-0x18],rdi
4004ee: 89 75 e4         mov     DWORD PTR [rbp-0x1c],esi
4004f1: c7 45 fc 01 00 00 00  mov     DWORD PTR [rbp-0x4],0x1

```

这三句理解为参数的传递，和局部变量的声明，

由后面的代码段(后面会分析下)可以了解到 [rbp-0x4] 里存放的值是相当于c语言的 `for(int i=1;i<=28;i++)` 中的 i，
 DWORD PTR [rbp-0x1c] 中存放的值是 28，控制循环次数的。

RDI 就是数组第一个元素的地址。

```

4004f8: eb 28           jmp     400522 <func+0x3c>

```

跳转到 400522，分析下

```

400522: 8b 45 fc      mov     eax,DWORD PTR [rbp-0x4]
400525: 3b 45 e4      cmp     eax,DWORD PTR [rbp-0x1c]
400528: 7e d0        jle     4004fa <func+0x14>
40052a: 90           nop
40052b: 5d           pop     rbp
40052c: c3           ret

```

第一句将 (DWORD PTR [rbp-0x4]) 赋值给 eax，eax=1;

第二句 比较 eax 和 (DWORD PTR [rbp-0x1c]) 的大小，如果 eax 小于等于后者 (28)，则跳转到4004fa，否则 函数结束返回。由第二句可以知道 DWORD PTR [rbp-0x1c] 中的存放的值是 控制循环次数的
 即main函数中 func(input, 28) 中的28。而eax 即是 i。

下面分析下 4004fa 处。

```

4004fa: 8b 45 fc      mov     eax,DWORD PTR [rbp-0x4] // eax =1
4004fd: 48 63 d0      movsxd rdx,eax //rdx = 1

400500: 48 8b 45 e8      mov     rax,QWORD PTR [rbp-0x18]
// [rbp-0x18]中放的是数组第一个元素的地址
400504: 48 01 d0      add     rax,rdx
// rax 是第二个元素的地址，用rax保存地址，方便后面将异或的值赋值给数组
400507: 8b 55 fc      mov     edx,DWORD PTR [rbp-0x4] //edx = 1
40050a: 48 63 ca      movsxd rcx,edx
40050d: 48 8b 55 e8      mov     rdx,QWORD PTR [rbp-0x18]
400511: 48 01 ca      add     rdx,rcx // rdx = ([rbp-0x18] ) +1
//RDX 中保存了 第二个元素的地址
400514: 0f b6 0a      movzx  ecx,BYTE PTR [rdx]
//ecx 中 保存的是 数组第二个元素的值

400517: 8b 55 fc      mov     edx,DWORD PTR [rbp-0x4]
// i =edx=1
40051a: 31 ca      xor     edx,ecx
//将 第二个元素 和 1 异或
40051c: 88 10      mov     BYTE PTR [rax],dl
40051e: 83 45 fc 01    add     DWORD PTR [rbp-0x4],0x1
//这里相当于 c 语言 i++

```

上面的注释都是以第一次循环写的，后面的依次类推。

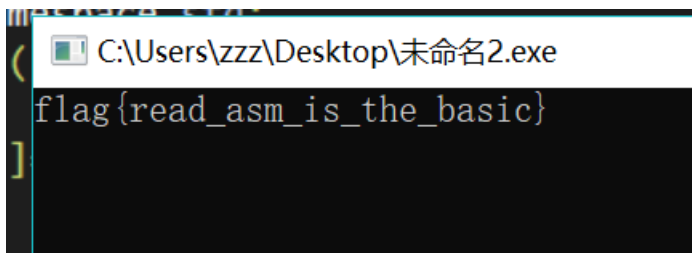
函数功能就是将数组第二个元素到第29个元素分别和 i 进行异或。

写个c++

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int a[] = {0x0, 0x67, 0x6e, 0x62, 0x63, 0x7e, 0x74, 0x62, 0x69, 0x6d,
              0x55, 0x6a, 0x7f, 0x60, 0x51, 0x66, 0x63, 0x4e, 0x66, 0x7b,
              0x71, 0x4a, 0x74, 0x76, 0x6b, 0x70, 0x79, 0x66, 0x1c};
    for(int i=1; i<=28; i++)
    {
        a[i]=a[i]^i;
    }
    for(int i=1; i<=28; i++)
        cout<<char(a[i]);
    return 0;
}
```

https://blog.csdn.net/qq_43394612

输出结果即是 flag



```
C:\Users\zzz\Desktop\未命名2.exe
flag{read_asm_is_the_basic}
```