

XCTF嘉年华 re1 Writeup

原创

ChengKaoAO 于 2017-10-17 16:00:59 发布 974 收藏

分类专栏: [CTF CTF](#) 文章标签: [XCTF Reverse](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY](#) 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/ZmeiXuan/article/details/78261545>

版权



CTF 同时被 2 个专栏收录

5 篇文章 0 订阅

订阅专栏

[CTF](#)

11 篇文章 0 订阅

订阅专栏

摘要

CTF解题 > XCTF嘉年华体验赛

>> re1

Reverse 题目描述

Github <https://github.com/Cherishao>

it's easy

文件后缀是个.ppp, 感觉是个二进制文件, 放到kali下查看一下(用file命令查看)。

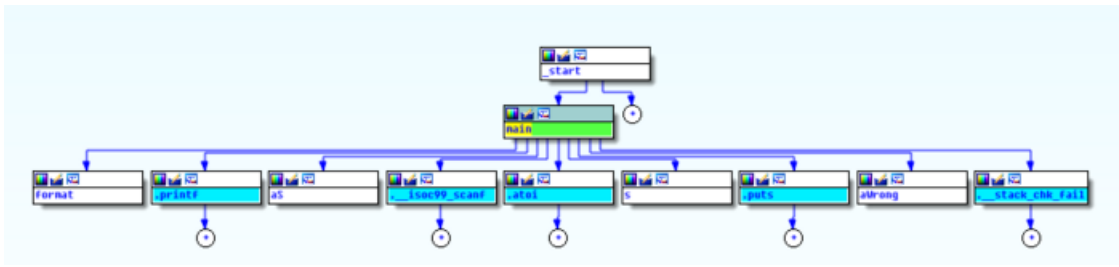
```
root@kali:file 1bc87216-beb5-4e56-a7b5-2955d2a200b6.ppp
```

由结果知, 是一个32位的二进制文件。

```
root@ChengKaoAo: ~/CTF/XCTF嘉年华体验赛/re1
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
root@ChengKaoAo:~/CTF/XCTF嘉年华体验赛/re1# file 1bc87216-beb5-4e56-a7b5-2955d2a200b6.ppp
1bc87216-beb5-4e56-a7b5-2955d2a200b6.ppp: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked,
interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.24, BuildID[sha1]=a409cfda7dfd88ea622eecd29f50ca894ad5b5f, not stri
pped
root@ChengKaoAo:~/CTF/XCTF嘉年华体验赛/re1#
```

测试运行情况, 报错, 知需要输入一个正确的值。

64位IDA pro打开, 查看到文件的整体结构, 找到main函数, 双击函数地址, 按F5, 把汇编变成C语言。



找到main函数，双击函数地址，按F5，把汇编变成C语言，这个函数可以看出，软件的运行流程大致是，先输入一个字符串，然后看这个字符串是否符合一定运算规则，符合运算规则正确，不符合的错误。

```

1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     int result; // eax@6
4     int v4; // edi@8
5     int v5; // [sp+24h] [bp-84h]@2
6     int v6; // [sp+30h] [bp-78h]@2
7     int v7; // [sp+38h] [bp-70h]@1
8     int v8; // [sp+9Ch] [bp-Ch]@1
9
10    v8 = *MK_FP(__GS__, 20);
11    printf("please input the key:");
12    memset(&v7, 0, 0x64u);
13    isoc99_scanf("%s", &v7);
14    if ( strlen((const char *)&v7) != 4
15        || (v5 = atoi((const char *)&v7), v6 = v5 % 100 / 10, v5 % 10 + v6 + v5 / 1000 + v5 % 1000 / 100 != 23)
16        || v6 / (v5 % 10) != 2
17        || v5 % 1000 / 100 - v6 != -1
18        || v5 / 1000 % v6 != 3 )
19    {
20        puts("wrong!");
21        result = 1;
22    }
23    else
24    {
25        puts("correct!");
26        result = 1;
27    }
28    v4 = *MK_FP(__GS__, 20) ^ v8;
29    return result;
30 }

```

在运算规则中，可以看书第一条是检测输入的字符串长度是否为四位数。

- 因此我们就可以写个程序，从1000开始计算，一直循环到9999。

```

#include <stdio.h>

int main(void)
{
    int count = 0;
    int i = 0, j = 0;
    for (count = 1000; count <= 9999; count++)
    {
        if ((i = count, j = i % 100 / 10, i % 10 + j + i / 1000 + i % 1000 / 100 != 23)
            || (j / (i % 10) != 2)
            || (i % 1000 / 100 - j != -1)
            || (i / 1000 % j != 3))
        {
            continue;
        }
        else
        {
            printf("%d\n", i);
        }
    }

    printf("Hello World!\n");
    return 0;
}

```

直接运行会终端因为有的数字不符合规则，调试碰到错误，跳过这个数字继续。最终我们得到的答案是：

```

root@kali:~/Desktop# ./1bc87216-beb5-4e56-a7b5-2955d2a200b6.ppp
please input the key:9563
correct!
root@kali:~/Desktop#

```