全国大学生信息安全竞赛writeup--暗号(reverse300)

原创

Anciety 于 2016-07-10 22:48:22 发布 5849 收藏 4

分类专栏： android security ctf 文章标签： 信息安全

本文链接：https://blog.csdn.net/qq_29343201/article/details/51873914

描述

George是一名FBI特工，昨天他获得了一个命令，在今天晚上纽约林肯中心的大都会歌剧院的演唱会上，将会有恐怖分子出没。George下午结果了一名恐怖分子，在他的身上发现了一个U盘，里面有这样一个程序。George看了程序之后，决定将自己伪装成刚解决掉的恐怖分子的身份。请你帮助George获得这名恐怖分子的姓名和暗号。

注：为MadFrog生成注册码

tips：flag是flag{暗号}

附件描述：

文件名：确认姓名和暗号的程序

校验（SHA1）：DC0CB7AF31E70514CD63DEC7B2E57CAAB4E8888E

思路

打开发现还是一个登陆，逆向发现有so。check函数在so里，要看把check函数看懂需要一些功夫。注意要把一些已知的JNIEnv *env参数改回来，IDA在逆向的时候可能当做int处理了。不然有一些带偏移的函数指针会不知道是什么。 基本上把这个改了之后就好懂多了，SO创建了一个服务器一个客户端。

服务器接收客户端的数据，并和hhxptgdlffojwztpewc进行比较，观察是否一样，如果一样，发送返回的数据hehehhehe。
客户端将输入的字符串经过一连串加密之后发送已加密的数据给服务器端，如果结果是hehehhehe则成功。

加密算法大致如下

```
bool check(char *str, int len)
{
    char encode[202], needstr[202], src;
    char checkstr[202] = "hhxptgdlffojwztpewc";
    int i, b, c, e, idx, which;
    char *strPiece;
    char *finalStr = (char *)malloc(2020);
    memset(finalStr, 0, sizeof(char) * 1111);

    if (!has[len])
    {
        printf("checking len:%d\n", len);
        has[len] = true;
    }
    len = strlen(str);
    i = 0;
    while (i < len)
    {
        sprintf(&encode[3 * i], "%03d", str[i]);
        i++;
    }

    puts(encode);
    memset(needstr, '0', sizeof(needstr));

    idx = 0;
    which = 0;
    for (i = 0;;i = 0)
    {
        while (i < len)
        {
            b = 100 * encode[3 * i] + 1000 * idx;
            c = 3 * i;
            i++;
            strPiece = &encode[c];
            e = b + 10 * encode[c + 1] + encode[c + 2] - 5328;
            printf("b:%d c:%d e:%d idx:%d\n", b, c, e, idx);
            idx = e % 26;
            e /= 26;
            encode[c] = e / 100 + 48;
            strPiece[1] = e % 100 / 10 + 48;
            strPiece[2] = e % 10 + 48;
            printf("after: %c%c%c\n", encode[c], strPiece[1], strPiece[2]);
            getchar();
        }
        sprintf(&src, "%c", idx + 97);
        which++;
        strcat(finalStr, &src);
        puts(encode);
        if (!memcmp(encode, needstr, 3 * len))
            break;
        idx = 0;
    }
    printf("hey, the string is \"%s\"\n", finalStr);
    return true;
}
```

这是我还原的加密算法，中间有一些用来观察数据特征的输出，可以删掉。 不过根据这样的观察，我们会发现（当然也可以用数学推理，稍显复杂），其加密过程大致如下。
每一轮得到一个字符。首先将一个字符串还原成ASCII的字符串，比如abc 会变为 097098099 三位一组。然后开始进行一轮运算，每一轮运算，又包括从左至右的n组运算，每一组运算将ascii值提出来，比如097就提为97，然后对26取模，得19，于是将19作为千位再加上下一组ascii值，继续进行运算。而当前位将由ascii值+上一位遗留作为千位的值 再除26得到商，商即是该位的值，余数即是留给下一位的值。下面举例说明:
097 098 099 第一次运算后 得到98%26 = 19，作为千位加下一位得19098，第一位为98/26 = 3，
下一位19098%26=14，作为千位加下一位得14099，第二位为19098/26=734，最后第三位为14099/26 = 542. 第三位得到的余数14099%26=7，即为第一轮得到的字母在字母表中的索引，即’a’+7 = ‘h’
下一轮运算则由 003 734 542 继续进行一样的运算，直到所有位都为0.

逆向这个算法，每一位有用的值都为除法后的商和余数，另x为这个值，(x - y) * t = 原值，最后由于全为0，所以最后的余数就是最后一位在字母表中的索引，根据这个逆推就可以了，逆推方式，最后一个值作为余数，倒数第二个字母在字母表中的索引即为商，然后一轮一轮逆推。
其他不懂的见逆推代码

```cpp
#include <iostream>
#include <cstdio>
#include <cstring>
using namespace std;
string password = "hhxptgdlffojwztpewc";
#define GETIDX(X) (X - 'a')
void solve(int len)
{
    int num[10];
    bool init = true;
    int delta = 0;
    memset(num, 0, sizeof(num));
    for (int i = password.length() - 1;i >= 0;i--)
    {
        if (init)
        {
            init = false;
            num[len - 1] = GETIDX(password[i]);
            goto nextLoop;
        }
        for (int j = len - 1;j >= 0;j--)
        {
            if (true)
            {
                delta = (j == len - 1)?GETIDX(password[i]):delta;
                num[j] *= 26;
                num[j] += delta;
                delta = num[j] / 1000;
                num[j] %= 1000;
                printf("j:%d delta:%d\n", j, delta);
            }
        }

nextLoop:
        for (int i = 0;i < len;i++)
            printf("%d ", num[i]);
        printf("\n");
    }

    for (int i = 0;i < len;i++)
        printf("%c", num[i]);
    printf("\n");

}
int main()
{
    solve(9);
    return 0;
}
```

后记（吐槽）

真是逼出了我的算法老本。