

【2020安洵杯】EasyCM WriteUp

原创

古月浪子 于 2020-12-17 23:39:41 发布 225 收藏 1

文章标签: CTF

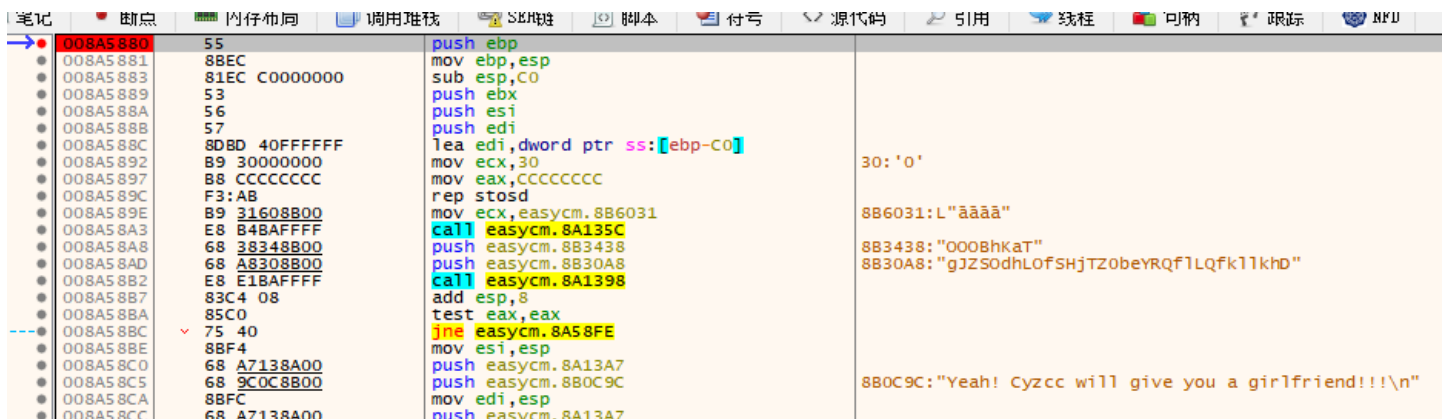
版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: <https://blog.csdn.net/tqdydq/article/details/110123052>

版权

这题从早上9点开始, 做了快3小时, 最后被人领先8分钟拿了一血, 只拿到一个二血, 呜呜呜/(T_o_T)/~~

这题IDA有点白给, 直接上x32dbg动调



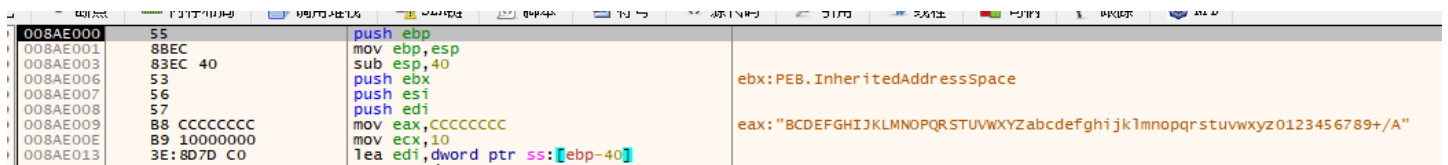
```
008A5880 55          push ebp
008A5881 8BEC       mov ebp,esp
008A5883 81EC C0000000 sub esp,C0
008A5889 53          push ebx
008A588A 56          push esi
008A588B 57          push edi
008A588C 8DBD 40FFFFFF lea edi,dword ptr ss:[ebp-C0]
008A5892 B9 30000000 mov ecx,30
008A5897 B8 CCCCCCCC mov eax,CCCCCCCC
008A589C F3:AB      rep stosd
008A589E B9 31608B00 mov ecx,easygm.8B6031
008A58A3 E8 B4BAFFFF call easygm.8A135C
008A58A8 68 38348B00 push easygm.8B3438
008A58AD 68 A8308B00 push easygm.8B30A8
008A58B2 E8 E1BAFFFF call easygm.8A1398
008A58B7 83C4 08     add esp,8
008A58BA 85C0       test eax,eax
008A58BC 75 40      jne easygm.8A58FE
008A58BE 8BF4       mov esi,esp
008A58C0 68 A7138A00 push easygm.8A13A7
008A58C5 68 9C0C8B00 push easygm.8B0C9C
008A58CA 8BFC       mov edi,esp
008A58CC 68 A7138A00 push easygm.8A13A7
```

可以看到要比较的字符串

我输入的是111d0g, 经过某种加密以后变成了OO0BhKaT

前面有判断输入是不是3的倍数, 不是则用-补齐的操作

推测是3位一组, 一组输出4个字符, 有点像base64



```
008AE000 55          push ebp
008AE001 8BEC       mov ebp,esp
008AE003 83EC 40     sub esp,40
008AE006 53          push ebx
008AE007 56          push esi
008AE008 57          push edi
008AE009 B8 CCCCCCCC mov eax,CCCCCCCC
008AE00E B9 10000000 mov ecx,10
008AE013 3E:8D7D C0 lea edi,dword ptr ss:[ebp-40]
```

加密操作在这个函数里, 会调用length/3次, 每次输出4个字符

这个函数是动态解密的, 而且dump下来后idapython写IDA里patch后也没法F5, 懒得修花指令了

读一读汇编能写出他的加密算法

解密脚本:

```

char table[] = "BCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/A";

char getC1(char* s, int i)
{
    int idx = ((s[0] & 0xC0) | (s[1] & 0x30) | ((s[2] & 0x3) << 2)) >> 2;
    return table[(idx + i) % 64];
}

char getC2(char* s, int i)
{
    int idx = ((s[1] & 0xC0) | (s[2] & 0x30) | ((s[0] & 0x3) << 2)) >> 2;
    return table[(idx + i) % 64];
}

char getC3(char* s, int i)
{
    int idx = ((s[2] & 0xC0) | (s[0] & 0x30) | ((s[1] & 0x3) << 2)) >> 2;
    return table[(idx + i) % 64];
}

char getC4(char* s, int i)
{
    int idx = (((s[0] << 4) & 0xC0) | ((s[1] << 2) & 0x30) | (s[2] & 0xC)) >> 2;
    return table[(idx + i) % 64];
}

int main()
{
    char t[4];
    t[3] = 0;
    char flag[] = "gJZS0dhLOfSHjTZ0beYRQf1LQfk1lkhD";
    for (size_t i = 0; i < 32; i += 4)
        for (size_t j = 0; j < 128; j++)
            for (size_t k = 0; k < 128; k++)
                for (size_t l = 0; l < 128; l++)
                {
                    t[0] = j;
                    t[1] = k;
                    t[2] = l;
                    if (getC1(t, i / 4) == flag[i] && getC2(t, i / 4) == flag[i + 1] && getC3(t, i / 4) == flag[i + 2] && getC4
(t, i / 4) == flag[i + 3])
                        cout << t;
                }
    return 0;
}

```