# 看雪CTF题目——流浪者

原创

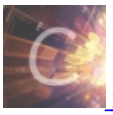
初识逆向 于 2019-08-06 21:13:33 发布 774 收藏 1

分类专栏： 逆向分析



本文链接：https://blog.csdn.net/w_g3366/article/details/98658968





逆向分析 专栏收录该内容

11 篇文章 1 订阅

订阅专栏

## 看雪CTF题目——流浪者

### 一、环境和工具

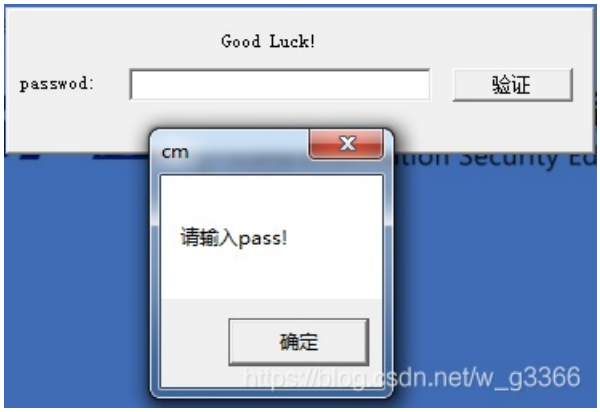
Windows7+OllyDbg

### 二、题目分析

1.先查壳，无壳程序

2.运行程序

- 不输入，之间点验证，弹出请输入pass对话框
- 随意输入123456，弹出错误，加油对话框





## 三、分析过程

提示很明显，我们直接搜索字符串可以定位关键位置了

| 地址 | HEX 数据 | 反汇编 | 注释 |
| --- | --- | --- | --- |
| 004019A3 | 8B45 F4 | MOV EAX,DWORD PTR SS:[EBP-0xC] | |
| 004019A6 | 895485 8C | MOV DWORD PTR SS:[EBP+EAX*4-0x74],EDX | |
| 004019AA | EB 05 | JMP SHORT cm.004019B1 | |
| 004019AC | E8 FFFDFFFF | CALL <cm.Fail> | 注册失败 |
| 004019B1 | 8B4D F4 | MOV ECX,DWORD PTR SS:[EBP-0xC] | |
| 004019B4 | 83C1 01 | ADD ECX,0x1 | |
| 004019B7 | 894D F4 | MOV DWORD PTR SS:[EBP-0xC],ECX | ecx=ecx+1,index+1 |
| 004019BA | E9 49FFFFFF | JMP cm.00401908 | 循环，找到循环正常结束位置 |
| 004019BF | 8D55 8C | LEA EDX,DWORD PTR SS:[EBP-0x74] | 输入为空，或者到达串尾跳到这 |
| 004019C2 | 52 | PUSH EDX | edx=拷贝后的key |
| 004019C3 | E8 28FEFFFF | CALL cm.004017F0 | |
| 004019C8 | 83C4 04 | ADD ESP,0x4 | |
| 004019CB | 5F | POP EDI | |
| 004019CC | 5E | POP ESI | |
| 004019CD | 5B | POP EBX | |
| 004019CE | 8BE5 | MOV ESP,EBP | |

这里有两个CALL，所有的跳转位置都看看，会发现如果输入的key不是数字字母就会调用第一个CALL，弹出错误对话框，满足条件的话：

- 数字：ASCII码-0x30
- A-Z：ASCII码-0x1D
- a-z：ASCII码-0x57
  将得到的值存进一个新的数组，然后调用第二个CALL

```
00401908    8B55 F8         MOV EDX,DWORD PTR SS:[EBP-0x8]          ; 如果key是数组，字母，就把key拷贝到【ebp-0x74
```

```
0040190B    0355 F4         ADD EDX,DWORD PTR SS:[EBP-0xC]
0040190E    0FBE02          MOVSX EAX,BYTE PTR DS:[EDX]             ; 取一字节
00401911    85C0            TEST EAX,EAX                            ; 判断key[0]==0?
00401913    0F84 A6000000   JE cm.004019BF                          ; 判断key是否为空或者到达末尾
00401919    8B4D F8         MOV ECX,DWORD PTR SS:[EBP-0x8]
0040191C    034D F4         ADD ECX,DWORD PTR SS:[EBP-0xC]
0040191F    0FBE11          MOVSX EDX,BYTE PTR DS:[ECX]
00401922    83FA 39         CMP EDX,0x39
00401925    7F 23           JG SHORT cm.0040194A                    ; key[0]>9?,key是数字，字母组成
00401927    8B45 F8         MOV EAX,DWORD PTR SS:[EBP-0x8]
0040192A    0345 F4         ADD EAX,DWORD PTR SS:[EBP-0xC]
0040192D    0FBE08          MOVSX ECX,BYTE PTR DS:[EAX]
00401930    83F9 30         CMP ECX,0x30
00401933    7C 15           JL SHORT cm.0040194A                    ; key[0]<0?
00401935    8B55 F8         MOV EDX,DWORD PTR SS:[EBP-0x8]
00401938    0355 F4         ADD EDX,DWORD PTR SS:[EBP-0xC]
0040193B    0FBE02          MOVSX EAX,BYTE PTR DS:[EDX]
0040193E    83E8 30         SUB EAX,0x30
00401941    8B4D F4         MOV ECX,DWORD PTR SS:[EBP-0xC]
00401944    89448D 8C       MOV DWORD PTR SS:[EBP+ECX*4-0x74],EAX   ; ebp-0x74[ecx]=key[0]
00401948    EB 67           JMP SHORT cm.004019B1
0040194A    8B55 F8         MOV EDX,DWORD PTR SS:[EBP-0x8]
0040194D    0355 F4         ADD EDX,DWORD PTR SS:[EBP-0xC]
00401950    0FBE02          MOVSX EAX,BYTE PTR DS:[EDX]
00401953    83F8 7A         CMP EAX,0x7A
00401956    7F 23           JG SHORT cm.0040197B                    ; key>z
00401958    8B4D F8         MOV ECX,DWORD PTR SS:[EBP-0x8]
0040195B    034D F4         ADD ECX,DWORD PTR SS:[EBP-0xC]
0040195E    0FBE11          MOVSX EDX,BYTE PTR DS:[ECX]
00401961    83FA 61         CMP EDX,0x61
00401964    7C 15           JL SHORT cm.0040197B                    ; key<a
00401966    8B45 F8         MOV EAX,DWORD PTR SS:[EBP-0x8]
00401969    0345 F4         ADD EAX,DWORD PTR SS:[EBP-0xC]
0040196C    0FBE08          MOVSX ECX,BYTE PTR DS:[EAX]
0040196F    83E9 57         SUB ECX,0x57
00401972    8B55 F4         MOV EDX,DWORD PTR SS:[EBP-0xC]
00401975    894C95 8C       MOV DWORD PTR SS:[EBP+EDX*4-0x74],ECX
00401979    EB 36           JMP SHORT cm.004019B1
0040197B    8B45 F8         MOV EAX,DWORD PTR SS:[EBP-0x8]
0040197E    0345 F4         ADD EAX,DWORD PTR SS:[EBP-0xC]
00401981    0FBE08          MOVSX ECX,BYTE PTR DS:[EAX]
00401984    83F9 5A         CMP ECX,0x5A
00401987    7F 23           JG SHORT cm.004019AC                    ; key>Z
00401989    8B55 F8         MOV EDX,DWORD PTR SS:[EBP-0x8]
0040198C    0355 F4         ADD EDX,DWORD PTR SS:[EBP-0xC]
0040198F    0FBE02          MOVSX EAX,BYTE PTR DS:[EDX]
00401992    83F8 41         CMP EAX,0x41
00401995    7C 15           JL SHORT cm.004019AC                    ; key<A
00401997    8B4D F8         MOV ECX,DWORD PTR SS:[EBP-0x8]
0040199A    034D F4         ADD ECX,DWORD PTR SS:[EBP-0xC]
0040199D    0FBE11          MOVSX EDX,BYTE PTR DS:[ECX]
004019A0    83EA 1D         SUB EDX,0x1D
004019A3    8B45 F4         MOV EAX,DWORD PTR SS:[EBP-0xC]
004019A6    895485 8C       MOV DWORD PTR SS:[EBP+EAX*4-0x74],EDX
004019AA    EB 05           JMP SHORT cm.004019B1
004019AC    E8 FFFDFFFF     CALL <cm.Fail>                          ; 注册失败
004019B1    8B4D F4         MOV ECX,DWORD PTR SS:[EBP-0xC]
004019B4    83C1 01         ADD ECX,0x1
004019B7    894D F4         MOV DWORD PTR SS:[EBP-0xC],ECX          ; ecx=ecx+1,index+1
```

```
004019BA   ^ E9 49FFFFFF     JMP cm.00401908                          ; 循环，找到循环正常结束位置
```

上面分析完也就是对我们的输入字符串做了一定处理，得到一个新的数组。分析第二个CALL，可以明显看到strcmp字符串比较，根据判断结果调用成功或失败函数，这里看strcmp函数的参数，

- 参数一：KanXueCTF2019JustForhappy
- 参数二：adfgsd（随便写的）：没见过，肯定是计算出来的

可以确定了最终的目标字符串就是它了：KanXueCTF2019JustForhappy

```
00401862  .  50            PUSH EAX                          004035C0  KanXueCTF2019JustForhappy
00401863  .  8D4D DC       LEA ECX,[LOCAL.9]                 ecx=key计算后得到的字符串
00401866  .  51            PUSH ECX                          s1
00401867  .  E8 84070000   CALL <JMP.&MSVCRT.strcmp>         strcmp
0040186C  .  83C4 08       ADD ESP,0x8
0040186F  .  85C0          TEST EAX,EAX
00401871  .v 75 07         JNZ SHORT cm.0040187A             判断key是否正确?
00401873  .  E8 F8FEFFFF   CALL <cm.Success>
00401878  .v EB 05         JMP SHORT cm.0040187F
0040187A  >  E8 31FFFFFF   CALL <cm.Fail>
0040187F  >  5F            POP EDI
```

F8走一遍可以发现比较函数上面的代码就是一个循环，并且出现了新的字符串
abcdefghiABCDEFGHIJKLMNjklmn0123456789opqrstuvwxyzOPQRSTUVWXYZ
这么长并且有规律，应该是做加密用的，逐行分析

```
0040181B  |> /8B45 FC       /MOV EAX,[LOCAL.1]                    ;  算法部分，eax=i=0
0040181E  |. |8B4D 08       |MOV ECX,[ARG.1]                      ;  ecx=key
00401821  |. |833C81 3E     |CMP DWORD PTR DS:[ECX+EAX*4],0x3E    ;  ecx[i]
00401825  |. |7D 30         |JGE SHORT cm.00401857                ;  key[i]>=0x3E,即加密字符串数组长度=0x3E,就直接GG
00401827  |. |8B55 FC       |MOV EDX,[LOCAL.1]
0040182A  |. |8B45 08       |MOV EAX,[ARG.1]
0040182D  |. |833C90 00     |CMP DWORD PTR DS:[EAX+EDX*4],0x0     ;  key[i]
00401831  |. |7C 24         |JL SHORT cm.00401857                 ;  key[i]<0也直接GG
00401833  |. |8B4D FC       |MOV ECX,[LOCAL.1]
00401836  |. |8B55 08       |MOV EDX,[ARG.1]
00401839  |. |8B048A        |MOV EAX,DWORD PTR DS:[EDX+ECX*4]     ;  eax=key[i]
0040183C  |. |8B4D FC       |MOV ECX,[LOCAL.1]
0040183F  |. |8B95 50FFFFFF |MOV EDX,[LOCAL.44]                   ;  key=string
00401845  |. |8A0402        |MOV AL,BYTE PTR DS:[EDX+EAX]         ;  al=string[eax]
00401848  |. |88440D DC     |MOV BYTE PTR SS:[EBP+ECX-0x24],AL    ;  buff=[ebp-0x24],buff[i]=al
0040184C  |. |8B4D FC       |MOV ECX,[LOCAL.1]
0040184F  |. |83C1 01       |ADD ECX,0x1                          ;  i++
00401852  |. |894D FC       |MOV [LOCAL.1],ECX
00401855  |.^\EB C4         \JMP SHORT cm.0040181B                ;  循环，计算正确的字符串
```

**总结：**

- 根据输入的key计算，得到一个索引数组:index=题中输入字符串 数字:ASCII-0x30 A-Z:ASCII-0x1D a-z:ASCII-0x57
- 根据索引数组和加密字符串数组,得到加密后的key
- 判断加密key是否是KanXueCTF2019JustForhappy

## 四、计算key

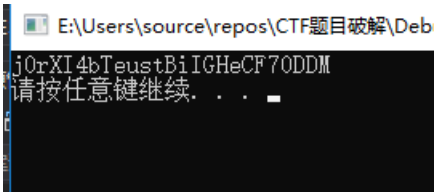
```
bool check(char index,char ch)
{
 char tmp = index + ch;
 if (tmp >= '0'&&tmp <= '9'&&ch==0x30)
 {
  return true;
 }
 else if (tmp >= 'A'&&tmp <= 'Z'&&ch == 0x1D)
 {
  return true;
 }
 else if (tmp >= 'a'&&tmp <= 'z'&&ch == 0x57)
 {
  return true;
 }
 return false;
}

int main()
{
 //看雪CTF:流浪者
 string szBuff = "abcdefghiABCDEFGHIJKLMNjklmn0123456789opqrstuvwxyzOPQRSTUVWXYZ";
 string szString = "KanXueCTF2019JustForhappy";
 char myKey[50] = { 0 };
 //1.计算索引:index=题中输入字符串 数字:ASCII-0x30 A-Z:ASCII-0x1D a-z:ASCII-0x57
 //2.取出正确值:temp[i]=szBuff[index]

 //逆向
 //1.计算索引:
 //2.计算输入字符:字符=索引+0x30
 int dwlen = szString.length();
 int dwlenght = szBuff.length();
 char nIndex = 0;

 //遍历正确字符串
 for (int i=0;i<dwlen;i++)
 {
  //遍历加密字符串,找到索引
  nIndex = szBuff.find(szString[i]);
  //计算输入字符:字符=索引+差值
  //判断字符类型是否是数字字母
  if (check(nIndex, 0x30))
  {
   myKey[i] = nIndex + 0x30;
  }
  else if (check(nIndex, 0x1D))
  {
   myKey[i] = nIndex + 0x1D;
  }
  else if (check(nIndex, 0x57))
  {
   myKey[i] = nIndex + 0x57;
  }
 }
 cout << myKey << endl;
 system("pause");
}
```

## 五、分析完成


E:\Users\source\repos\CTF题目破解\Deb
jOrXI4bTeustBiIGHeCF7ODDM
请按任意键继续. . .


Good Luck!
passwod:
jOrXI4bTeustBiIGHeCF7ODDM
验证
恭喜!
pass!
确定
https://blog.csdn.net/w_g3366