****

**分析报告**

|  |  |
| --- | --- |
| **样本名** | **010Editor** |
| **班级** | **35期** |
| **作者** | **Lracker** |
| **时间** | **11月13日** |
| **平台** | **Windows 7 32位** |

15PB信息安全研究院(协议分析报告)

# 目录

[目录 2](#_Toc483659381)

[1．样本概况 3](#_Toc483659382)

[1.1 应用程序信息 3](#_Toc483659383)

[1.2 分析环境及工具 3](#_Toc483659384)

[1.3 分析目标 3](#_Toc483659385)

[2．具体分析过程 3](#_Toc483659386)

[2.1 爆破 3](#_Toc483659387)

[2.1.1 通过关键字符串找出关键函数 3](#_Toc483659388)

[2.2 逆向出验证算法 3](#_Toc483659389)

[2.3 写出注册机 3](#_Toc483659388)

[2.4 网络验证 3](#_Toc483659388)

[3．总结 4](#_Toc483659390)

[参考文献 5](#_Toc483659391)

[附 录 6](#_Toc483659392)

[致 谢 7](#_Toc483659393)

# 1．样本概况

## 1.1 应用程序信息

应用程序名称：010Editor

MD5值：DB355048EF963913173F64AE0872AEE5

SHA1值：92CABC7485896E40FA3081FDF51425CB6ADBFE81

简单功能介绍：010 Editor 是一款全新概念的[十六进制编辑器](https://baike.baidu.com/item/%E5%8D%81%E5%85%AD%E8%BF%9B%E5%88%B6%E7%BC%96%E8%BE%91%E5%99%A8" \t "https://baike.baidu.com/item/010%20Editor%20V2.0/_blank)，能解析和编辑一切可视的二进制文件方面功能强大。

## 1.2 分析环境及工具

系统环境:win7 32位

工具:OLLDBG

## 1.3 分析目标

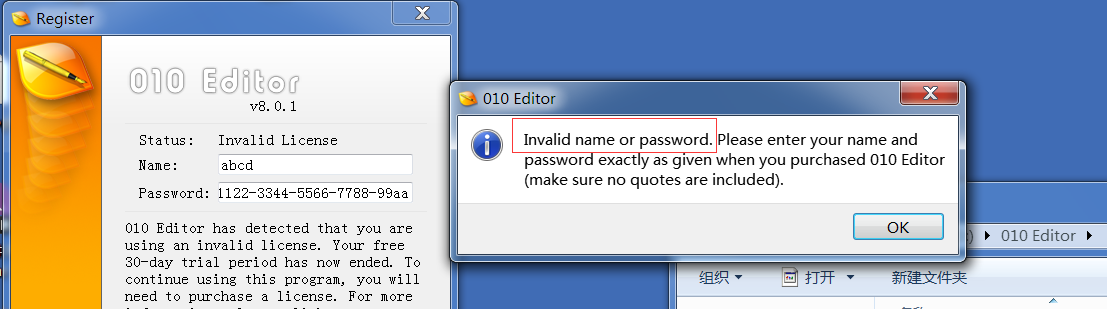
找出加解密算法，写出注册机

# 2．具体分析过程

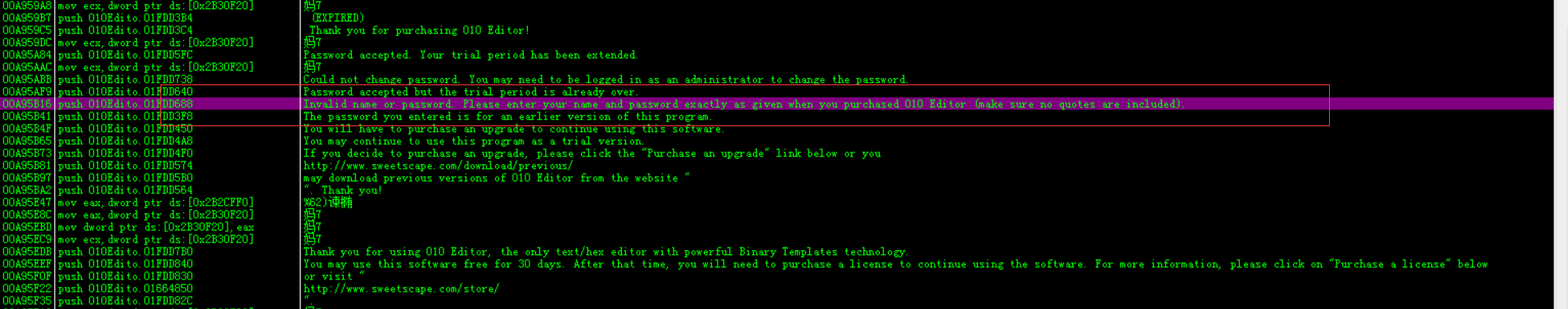
## 2.1 爆破

## 2.1.1 通过关键字符串找出关键函数

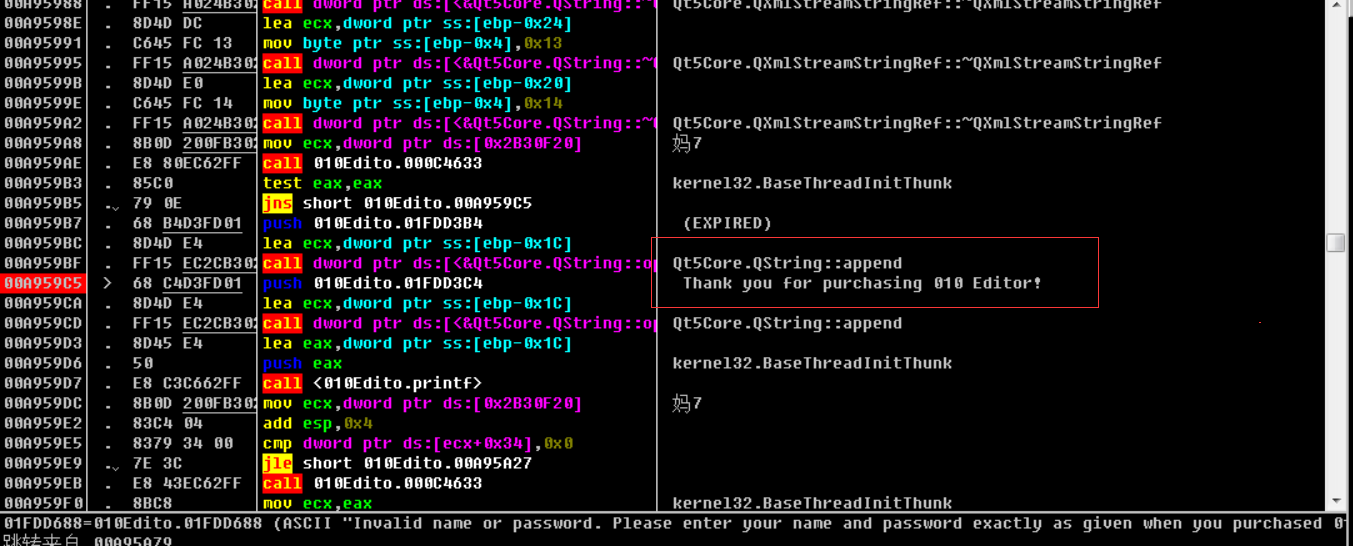
当我们尝试输入用户名和密码时，软件便会弹出一个窗口，告诉我们出错了。



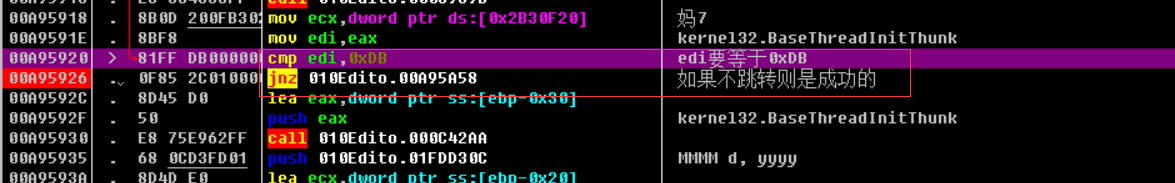
“Invalid name or password………………”根据这个字符串，我们可以在OD中通过只能搜索，搜索出该字符串。



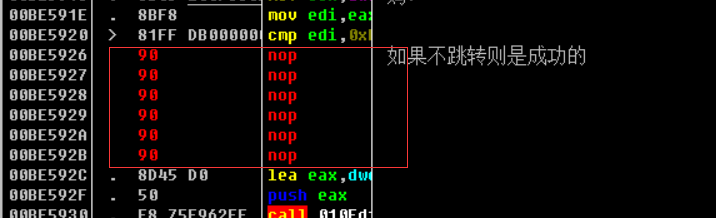
进而我们可以找到用户名和密码的验证函数了。找到以后继续往上翻翻，我们可以发现，上面有一个成功的标识。



所以我们要找到是哪里进行了判断，使得跳转跳到了错误提示。如果我们可以通过修改跳转来实现爆破。



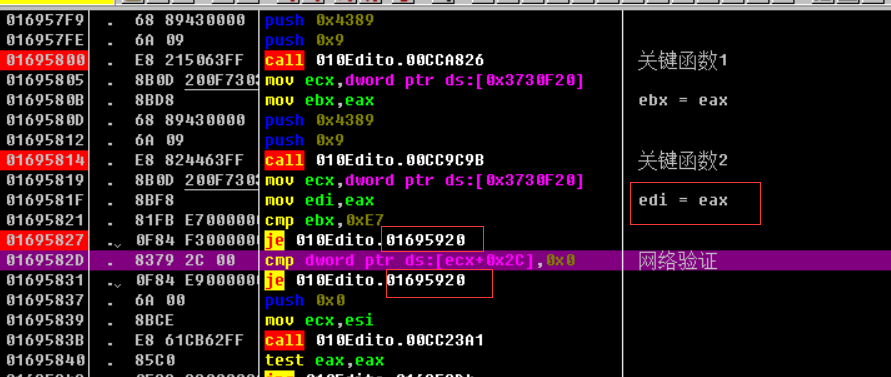
可以发现，这个跳转导致了我们提示错误信息。所以我们只要nop掉这里即可实现爆破。



## 2.2 逆向出验证算法(如有代码验证贴出关键代码)

下面的密码用k[]数组表示

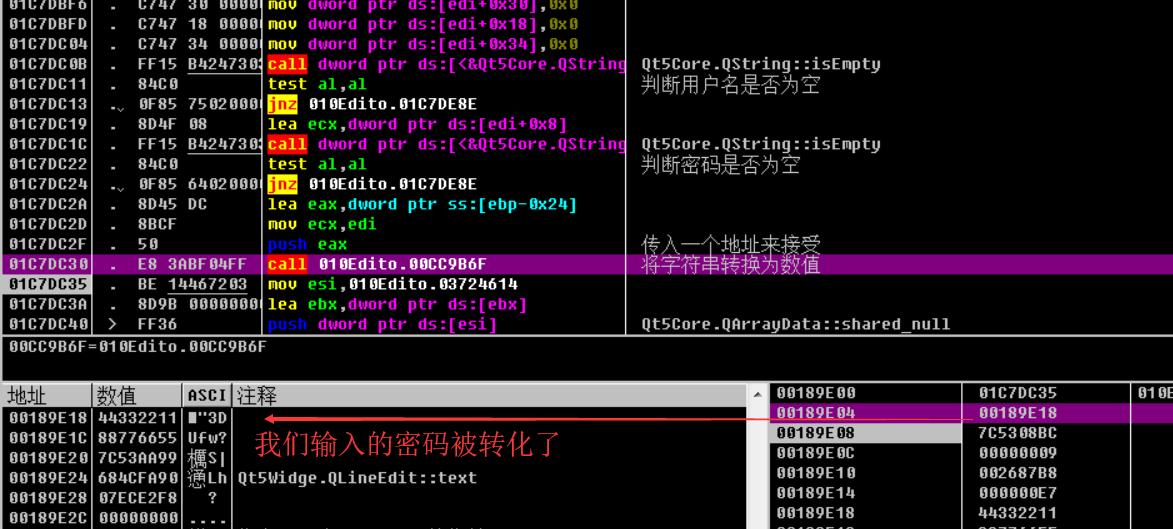
在上面的分析中，我们已经找到了是哪里进行判断跳转的。并且得知，EDI要等于0xDB才是正确的答案。所以我们可以继续往上翻翻，到底是哪里使得EDI的值改变的。



可以看到，这个edi的值是由关键函数2的返回值eax决定的，而这里有两个判断跳转，判断了ebx是否等于0xE7，和[ecx+0x2C]是否为0，这俩也很重要，但是首要任务是判断出edi怎样才能变成0xDB。



我们进去关键函数2里看一下，发现，这里面也有一个关键函数1，并且当这个函数的返回值为2D的时候，我们的程序才能走向成功的路线。所以我们需要分析关键函数一，并且让他的返回值等于0x2D即可。

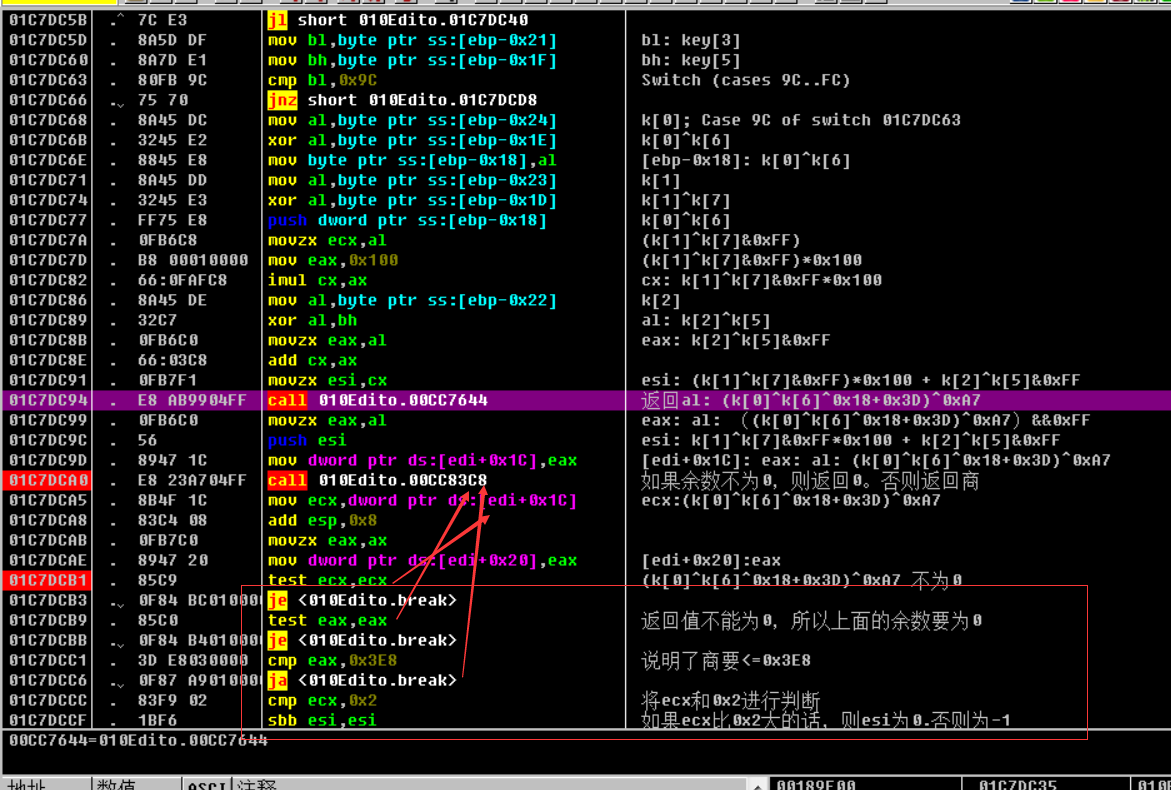


跟进去进行分析。我们可以发现，首先判断了一下用户名和密码是否为空，以及用户名如果是“999”就会失败了。然后将我们输入的密码字符串转换为数值了。

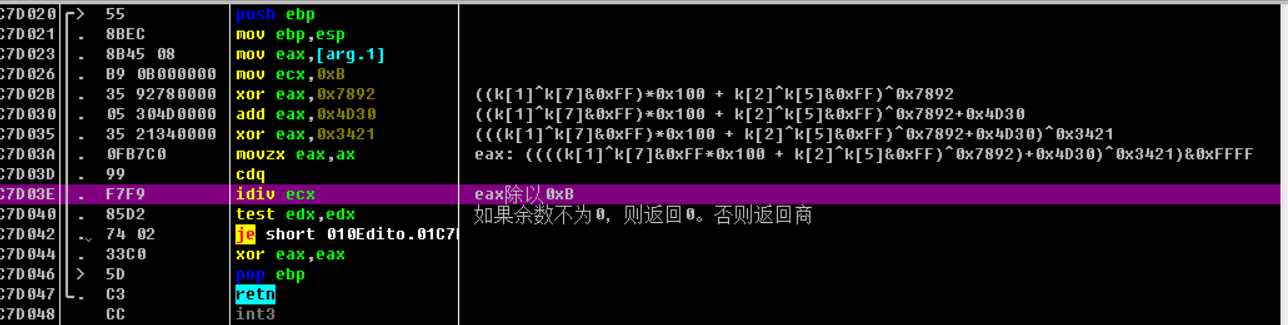
往下分析，我们看到这里判断了k[3]是否为0x9C，所以我们重新输入密码，并且将第三位输入为0x9C



继续往下看，我们可以看到有三个关键的条件跳转。首先我们来分析一下第一个，第一个是判断ecx是否为0的，根据上面的分析这个ecx里存放了(k[0]^k[6]^0x18+0x3D)^0xA7。第二个判断是来自于上面CALL 0x00CC83C8的返回值



所以我们继续分析一下CALL 0x00CC83C8这个函数里面。



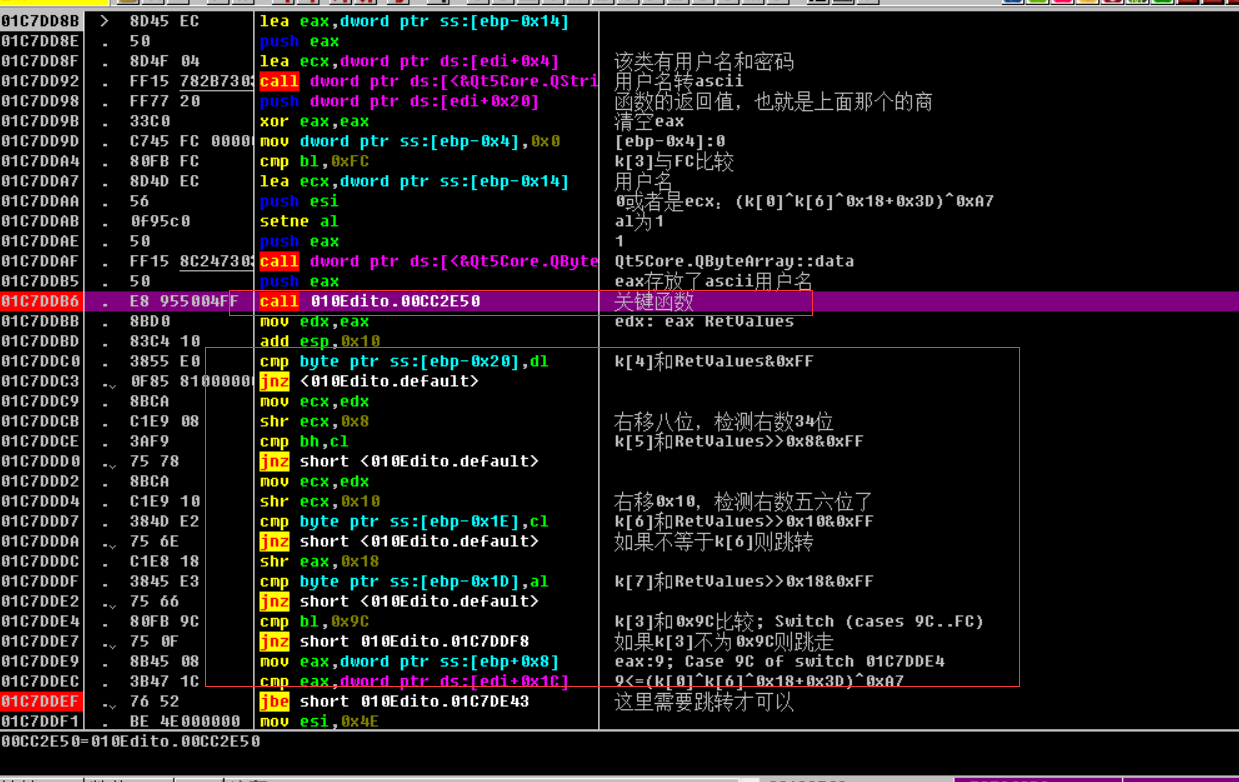
可以看到，这里面做了一个除法，假如余数不为零，则会将eax清零，这样会导致后面的第二个判断test eax,eax 失败。所以我们可以获得的条件就是说，((((k[1]^k[7]&0xFF\*0x100 + k[2]^k[5]&0xFF)^0x7892)+0x4D30)^0x3421)&0xFFFF%0xB要等于0。

又由于后面跳转的第三个判断是判断eax的，也就是判断除法的商是否小于等于0x3E8。

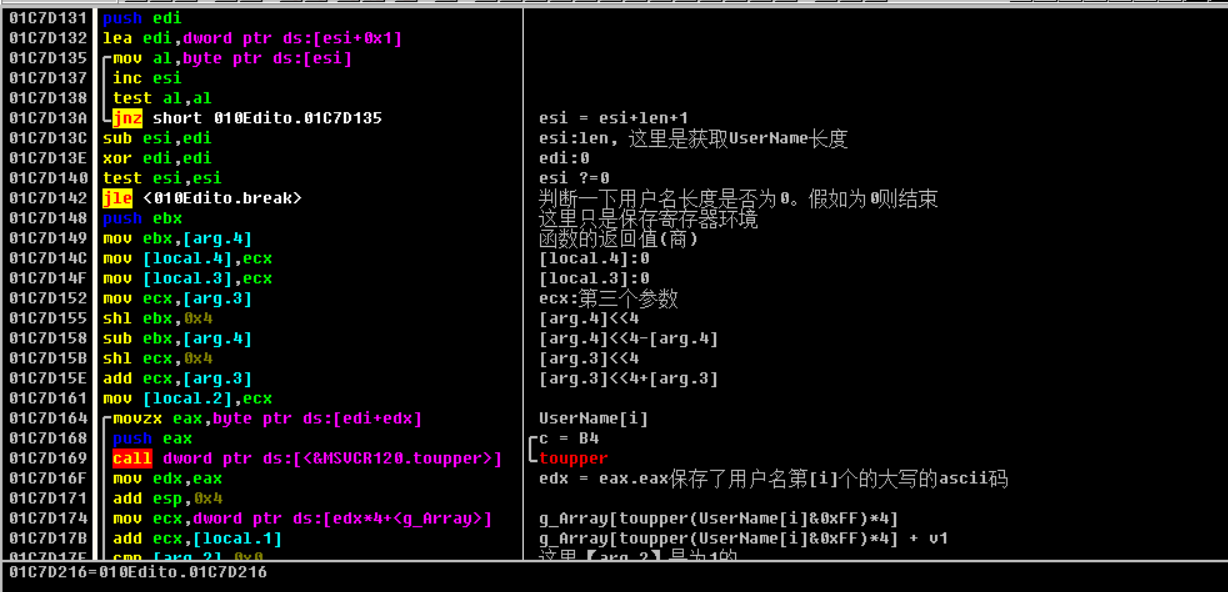
所以根据我们现在的分析，可以得出三个条件。

1. (k[0]^k[6]^0x18+0x3D)^0xA7!=0
2. (((((k[1] ^ k[7] & 0xFF) \* 0x100 + k[2] ^ k[5] & 0xFF) & 0xFFFF ^ 0x7892) + 0x4D30) ^ 0x3421) & 0xFFFF % 0xB == 0
3. (((((k[1] ^ k[7] & 0xFF) \* 0x100 + k[2] ^ k[5] & 0xFF) & 0xFFFF ^ 0x7892) + 0x4D30) ^ 0x3421) & 0xFFFF / 0xB <= 0x3E8

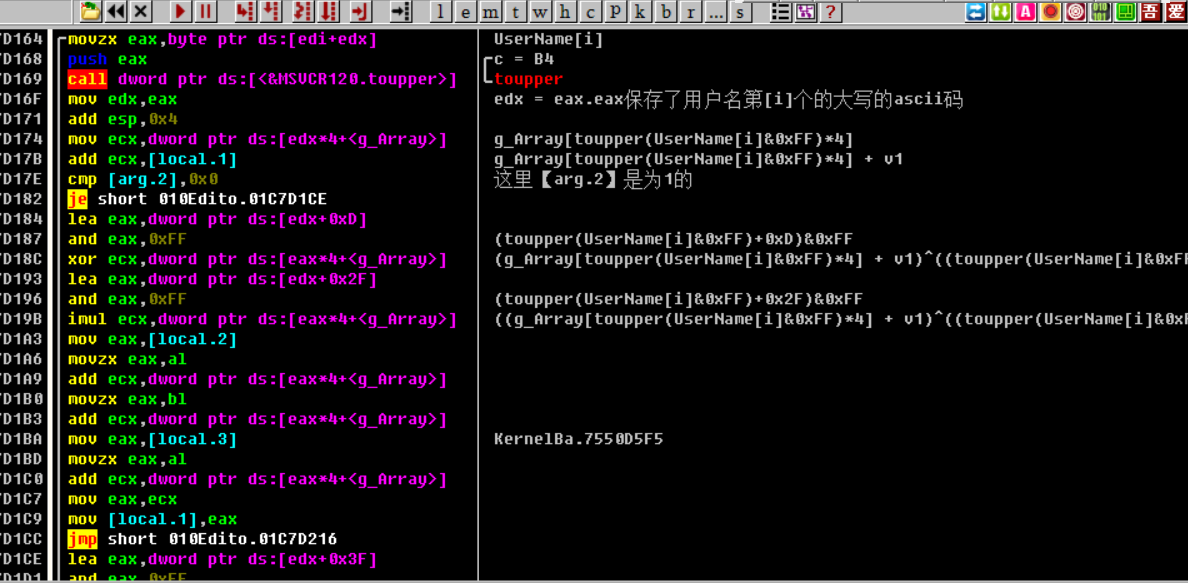
我们继续往下分析。



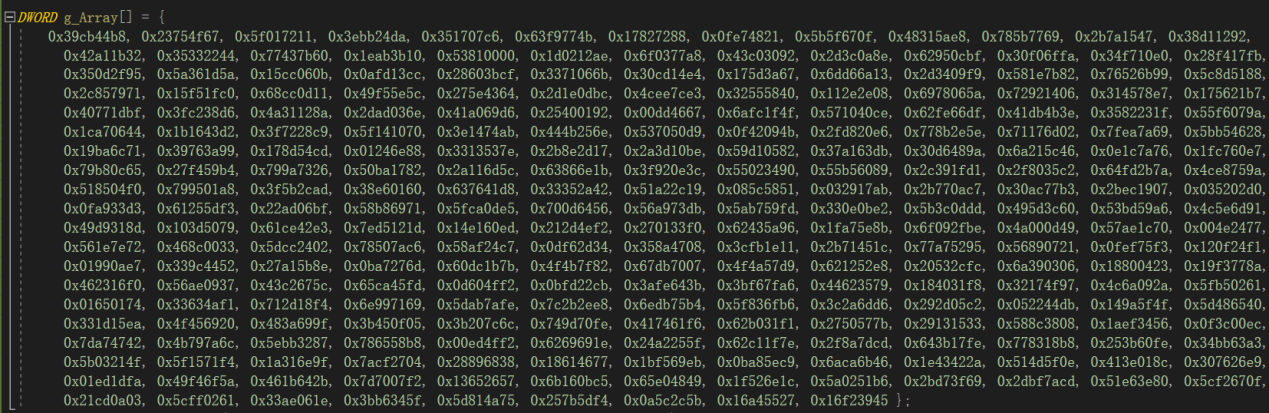
可以看到后面又有几个关键的跳转，分别判断了k[4]k[5]k[6]k[7]和上面一个函数的返回值的关系。因此我们得要去分析这个函数的返回值了。



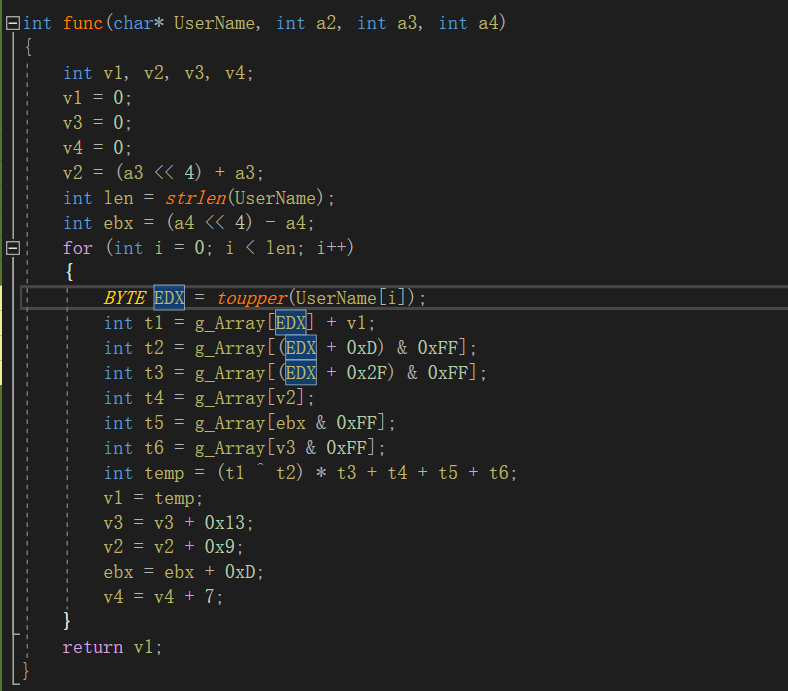
这看起来很复杂，需要一步一步的分析，前面先是获取了用户名的长度，并且判断长度是否为0。假如为0就跳走。接下来是一个循环。



里面还多次使用了一个全局数组。我们可以把这个数组复制出来



这个函数就是通过传入的用户名配合这个全局数组进行加密，函数的返回结果可以得到固定的值，从而获取k[4]k[5]k[6]k[7]这四个值。



**2.3 写出注册机**

接下来我们可以尝试着写注册机了。

#include <windows.h>

#include <stdio.h>

#include <time.h>

*DWORD* g\_Array[] = {

0x39cb44b8, 0x23754f67, 0x5f017211, 0x3ebb24da, 0x351707c6, 0x63f9774b, 0x17827288, 0x0fe74821, 0x5b5f670f, 0x48315ae8, 0x785b7769, 0x2b7a1547, 0x38d11292, 0x42a11b32, 0x35332244, 0x77437b60, 0x1eab3b10, 0x53810000, 0x1d0212ae, 0x6f0377a8, 0x43c03092, 0x2d3c0a8e, 0x62950cbf, 0x30f06ffa, 0x34f710e0, 0x28f417fb, 0x350d2f95, 0x5a361d5a, 0x15cc060b, 0x0afd13cc, 0x28603bcf, 0x3371066b, 0x30cd14e4, 0x175d3a67, 0x6dd66a13, 0x2d3409f9, 0x581e7b82, 0x76526b99, 0x5c8d5188, 0x2c857971, 0x15f51fc0, 0x68cc0d11, 0x49f55e5c, 0x275e4364, 0x2d1e0dbc, 0x4cee7ce3, 0x32555840, 0x112e2e08, 0x6978065a, 0x72921406, 0x314578e7, 0x175621b7, 0x40771dbf, 0x3fc238d6, 0x4a31128a, 0x2dad036e, 0x41a069d6, 0x25400192, 0x00dd4667, 0x6afc1f4f, 0x571040ce, 0x62fe66df, 0x41db4b3e, 0x3582231f, 0x55f6079a, 0x1ca70644, 0x1b1643d2, 0x3f7228c9, 0x5f141070, 0x3e1474ab, 0x444b256e, 0x537050d9, 0x0f42094b, 0x2fd820e6, 0x778b2e5e, 0x71176d02, 0x7fea7a69, 0x5bb54628, 0x19ba6c71, 0x39763a99, 0x178d54cd, 0x01246e88, 0x3313537e, 0x2b8e2d17, 0x2a3d10be, 0x59d10582, 0x37a163db, 0x30d6489a, 0x6a215c46, 0x0e1c7a76, 0x1fc760e7, 0x79b80c65, 0x27f459b4, 0x799a7326, 0x50ba1782, 0x2a116d5c, 0x63866e1b, 0x3f920e3c, 0x55023490, 0x55b56089, 0x2c391fd1, 0x2f8035c2, 0x64fd2b7a, 0x4ce8759a, 0x518504f0, 0x799501a8, 0x3f5b2cad, 0x38e60160, 0x637641d8, 0x33352a42, 0x51a22c19, 0x085c5851, 0x032917ab, 0x2b770ac7, 0x30ac77b3, 0x2bec1907, 0x035202d0, 0x0fa933d3, 0x61255df3, 0x22ad06bf, 0x58b86971, 0x5fca0de5, 0x700d6456, 0x56a973db, 0x5ab759fd, 0x330e0be2, 0x5b3c0ddd, 0x495d3c60, 0x53bd59a6, 0x4c5e6d91, 0x49d9318d, 0x103d5079, 0x61ce42e3, 0x7ed5121d, 0x14e160ed, 0x212d4ef2, 0x270133f0, 0x62435a96, 0x1fa75e8b, 0x6f092fbe, 0x4a000d49, 0x57ae1c70, 0x004e2477, 0x561e7e72, 0x468c0033, 0x5dcc2402, 0x78507ac6, 0x58af24c7, 0x0df62d34, 0x358a4708, 0x3cfb1e11, 0x2b71451c, 0x77a75295, 0x56890721, 0x0fef75f3, 0x120f24f1, 0x01990ae7, 0x339c4452, 0x27a15b8e, 0x0ba7276d, 0x60dc1b7b, 0x4f4b7f82, 0x67db7007, 0x4f4a57d9, 0x621252e8, 0x20532cfc, 0x6a390306, 0x18800423, 0x19f3778a, 0x462316f0, 0x56ae0937, 0x43c2675c, 0x65ca45fd, 0x0d604ff2, 0x0bfd22cb, 0x3afe643b, 0x3bf67fa6, 0x44623579, 0x184031f8, 0x32174f97, 0x4c6a092a, 0x5fb50261, 0x01650174, 0x33634af1, 0x712d18f4, 0x6e997169, 0x5dab7afe, 0x7c2b2ee8, 0x6edb75b4, 0x5f836fb6, 0x3c2a6dd6, 0x292d05c2, 0x052244db, 0x149a5f4f, 0x5d486540, 0x331d15ea, 0x4f456920, 0x483a699f, 0x3b450f05, 0x3b207c6c, 0x749d70fe, 0x417461f6, 0x62b031f1, 0x2750577b, 0x29131533, 0x588c3808, 0x1aef3456, 0x0f3c00ec, 0x7da74742, 0x4b797a6c, 0x5ebb3287, 0x786558b8, 0x00ed4ff2, 0x6269691e, 0x24a2255f, 0x62c11f7e, 0x2f8a7dcd, 0x643b17fe, 0x778318b8, 0x253b60fe, 0x34bb63a3, 0x5b03214f, 0x5f1571f4, 0x1a316e9f, 0x7acf2704, 0x28896838, 0x18614677, 0x1bf569eb, 0x0ba85ec9, 0x6aca6b46, 0x1e43422a, 0x514d5f0e, 0x413e018c, 0x307626e9, 0x01ed1dfa, 0x49f46f5a, 0x461b642b, 0x7d7007f2, 0x13652657, 0x6b160bc5, 0x65e04849, 0x1f526e1c, 0x5a0251b6, 0x2bd73f69, 0x2dbf7acd, 0x51e63e80, 0x5cf2670f, 0x21cd0a03, 0x5cff0261, 0x33ae061e, 0x3bb6345f, 0x5d814a75, 0x257b5df4, 0x0a5c2c5b, 0x16a45527, 0x16f23945 };

int func(char\* UserName, int a2, int a3, int a4)

{

int v1, v2, v3, v4;

v1 = 0;

v3 = 0;

v4 = 0;

v2 = (a3 << 4) + a3;

int len = *strlen*(UserName);

int ebx = (a4 << 4) - a4;

for (int i = 0; i < len; i++)

{

*BYTE* EDX = *toupper*(UserName[i]);

int t1 = g\_Array[EDX] + v1;

int t2 = g\_Array[(EDX + 0xD) & 0xFF];

int t3 = g\_Array[(EDX + 0x2F) & 0xFF];

int t4 = g\_Array[v2];

int t5 = g\_Array[ebx & 0xFF];

int t6 = g\_Array[v3 & 0xFF];

int temp = (t1 ^ t2) \* t3 + t4 + t5 + t6;

v1 = temp;

v3 = v3 + 0x13;

v2 = v2 + 0x9;

ebx = ebx + 0xD;

v4 = v4 + 7;

}

return v1;

}

int main()

{

*printf*("请输入用户名\n");

char UserName[] = { "abcd" };

*srand*(*time*(*NULL*));

*BYTE* k[8] = {};

k[3] = 0x9C;

int nKey = func(UserName, 1, 0, 0x3E8);

k[4] = nKey & 0xFF;

k[5] = nKey >> 0x8 & 0xFF;

k[6] = nKey >> 0x10 & 0xFF;

k[7] = nKey >> 0x18 & 0xFF;

*DWORD* dwResultVelue = 0;

while (*TRUE*)

{

*BYTE* k0 = *rand*() % 0xFF;

if (((k0 ^ k[6] ^ 0x18) + 0x3D) ^ 0xA7 >= 9)

{

k[0] = k0;

break;

}

}

while (*TRUE*)

{

*BYTE* k1 = *rand*() % 0xFF;

*BYTE* k2 = *rand*() % 0xFF;

*DWORD* EAX = (((((k1 ^ k[7] & 0xFF) \* 0x100 + k2 ^ k[5] & 0xFF) & 0xFFFF ^ 0x7892) + 0x4D30) ^ 0x3421) & 0xFFFF;

if (EAX % 0xB == 0 && EAX / 0xB == 0x3E8)

{

k[1] = k1;

k[2] = k2;

break;

}

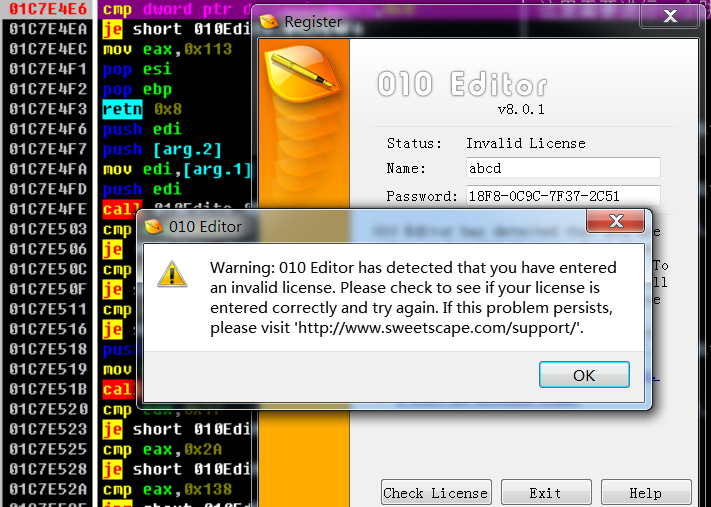
}

*printf*("%02X%02X-%02X%02X-%02X%02X-%02X%02X", k[0], k[1], k[2], k[3], k[4], k[5], k[6], k[7]);

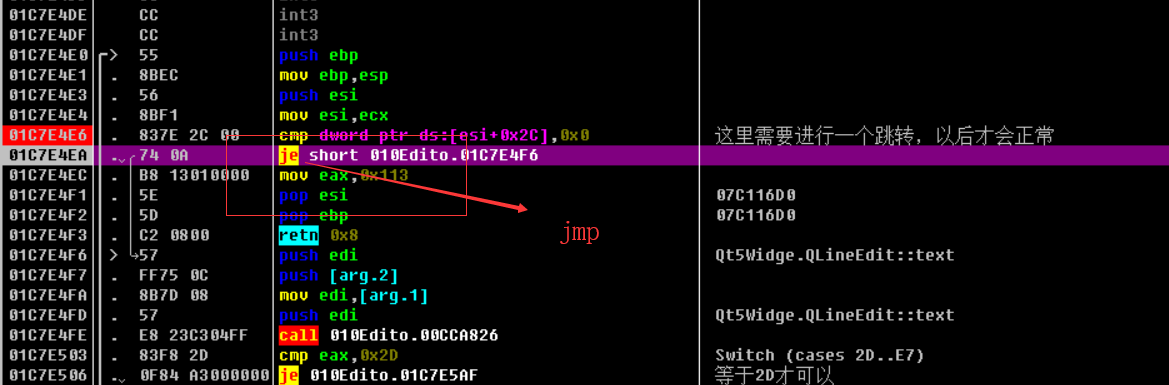
}

**2.4 网络验证**

在计算出正确的密码后，发现还是无法打开。并且弹出以下错误。

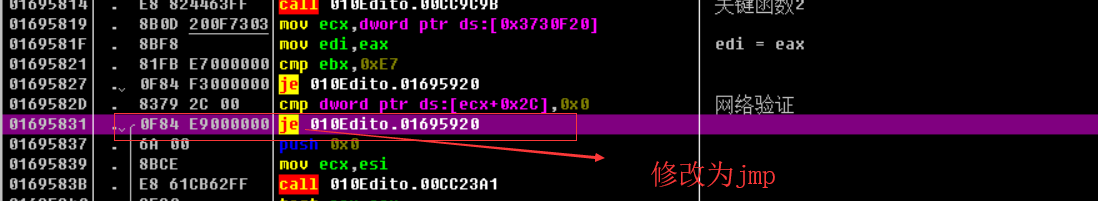


这个错误来自于网络验证，也就是说，我们其实并没有在服务端那边进行一个注册，我们只是做了本地的逆向破解。所以我们需要找出网络验证的部分，并且破解他。



首先第一个，这里判断了以下[esi+0x2C]的位置是否为0，如果不为0的话就会失败了，所以我们直接将je改为jmp。

第二个就是这里，也同样修改为jmp。



然后就成功打开我们的010Editor了

# 3．总结

在逆向的过程中，一开始还是有点困难的，毕竟密密麻麻的汇编代码，但这的确是要耐心，慢慢分析，一步一步的就能够分析出来了，到了后面就轻松多了。目标明确而不像一开始那样迷茫地寻找加解密函数。

参考文献

参考文献要另起一页，放在正文之后，且只列出编写人直接阅读过或在正文中被引用过的文献资料。

作者名只写到第三位，其余的写“等”，英文作者超过三人下“etal”(斜体)。

几种主要参考文献著录表的格式为：

出版物：作者，文题，刊号，年，卷号(期号)；起～止页码。

专(译)著：作者，书名(译者)，出版地；出版者，出版年，起～止页码。

论文集：作者，文题，编者，文集名，出版地；出版者，出版年，起～止页码。

专利：申请表，专利名，国名，专利文献种类，专利号，授权日期。

技术标准：发布单位、技术标准代号，技术标准名称，出版地；出版者，出版日期。

例如：

参考文献

(黑体三号，居中，上下空一行)

[1] 丰生强. Android软件安全与逆向分析. 人民邮电出版社. 2013.

[2] 杨丰盛. Android应用开发揭秘. 机械工业出版社. 2010.

[3] Bruce Eckel. Thinking in Java. Prentice Hall. 2006.

以上序号用中括号，与文字之间空两格，如果需要两行，第二行要位于序号的后面，与第一行文字对齐，中文用宋体小四，英文用Times New Roman字体

附 录

(黑体三号，居中，“附录”二字中间空五格，上下空一行)

附录要另起一页。附录得根据有无说明书（设计）情况而定，内容一般包括正文内不便列出的冗长公式推导、符号说明（含缩写）、样本等。

致 谢

(黑体三号，居中，“致谢”二字中间空五格,上下空一行)

正文用宋体小四，内容限1页，一律向15PB信息安全研究院谢意。