单人完成

**问题1：**

打开动态调试运行，在内存中可以找到：

表格

描述已自动生成

起始地址为0x0040004E，长度43字节

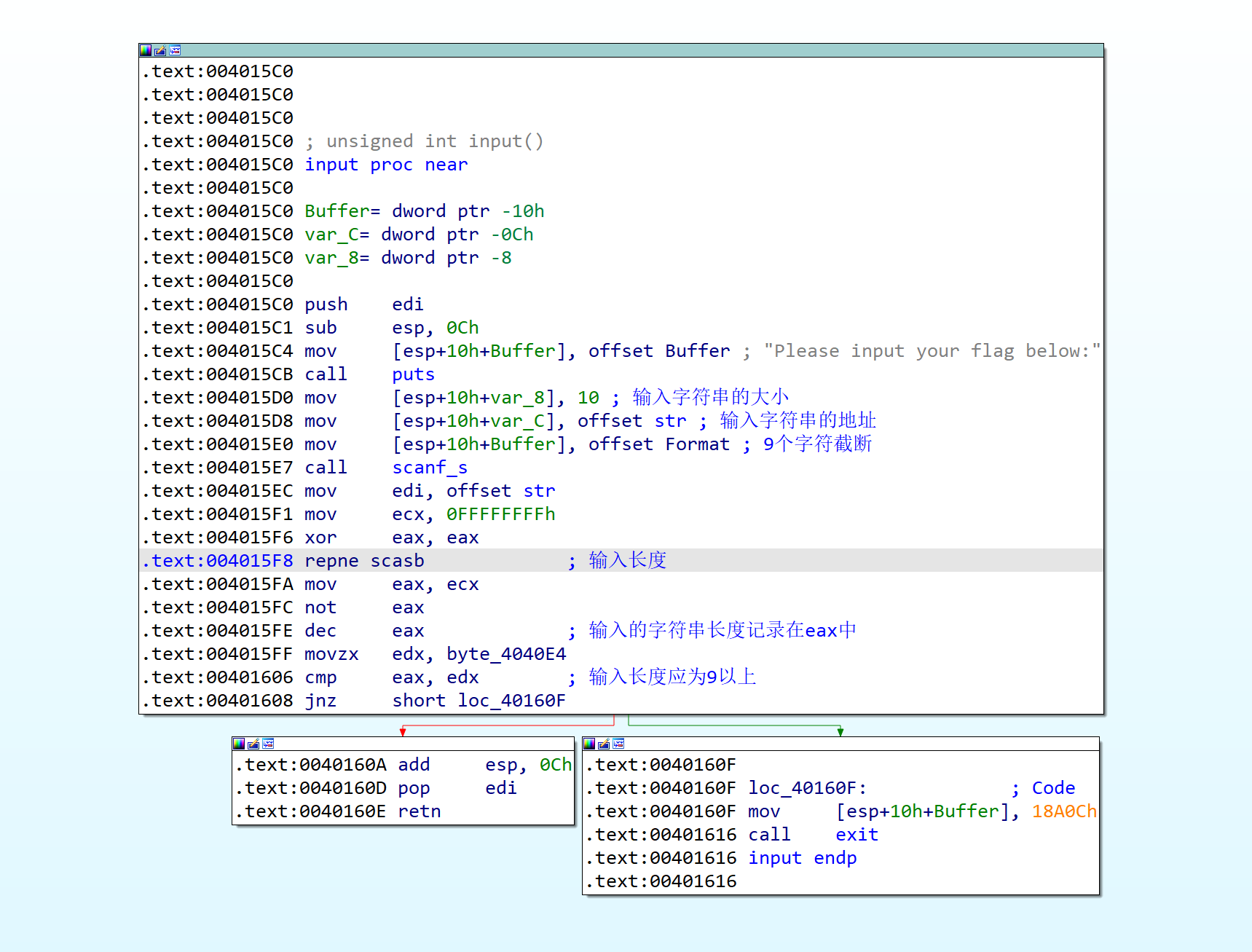
**问题2：**

程序main函数主体结构：



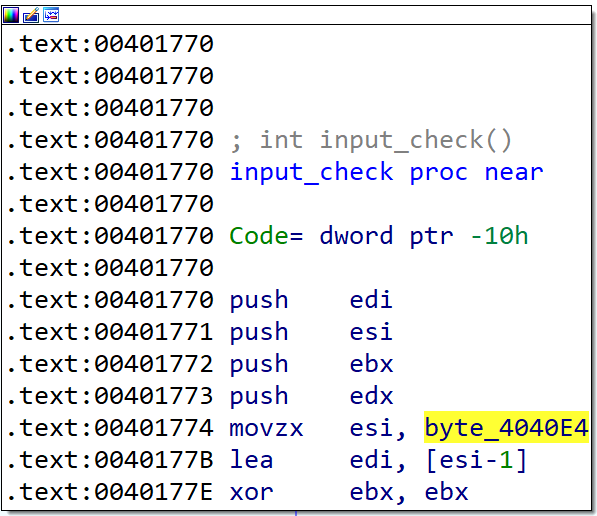
可得重点部分为如图几个有注释的函数。下分析几个函数：

Input部分：



如图易得input部分就调用了scanf\_s函数，对输入的字符串进行截断，如果输入不足9位就直接退出函数。

Input\_check部分：



对变量赋初始值，其中esi赋为9，edi赋值为8，ebx清零，初始化

图形用户界面, 文本, 应用程序

描述已自动生成

进入循环，当esi和ebx相同就说明已经全部比完，跳出循环

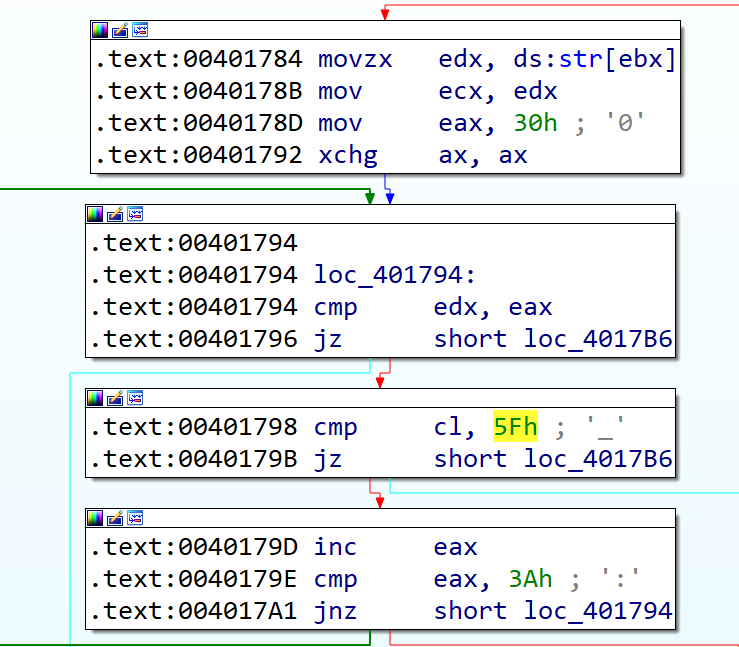
不然就进入以下模块：

将eax赋为30H，既字符‘0’的ascii值，取出输入字符串str[ebx]上的值进入edx作比较

图形用户界面, 文本, 应用程序

描述已自动生成

下面几段循环判断当前字符是否是数字或者字符‘\_’如果是的话，就跳到 loc\_4017B6,结束循环，当进入最下面cmp eax和3A相同时，说明当前字符既不是数字，也不是’\_’，则跳到下方代码段判断字符是否是大写字母。



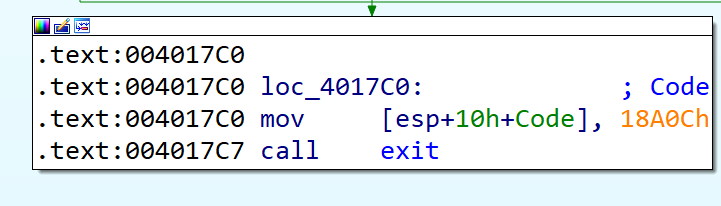
下面代码和上面逻辑相似，循环判断当前代码是否为大写字母，只是最后当判断不是大写字母时，认为输入错误，直接退出程序

图形用户界面, 文本, 应用程序, 表格

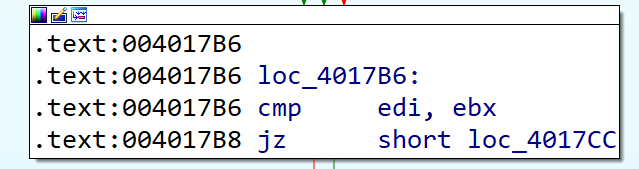
描述已自动生成

表格

中度可信度描述已自动生成



如果当前轮次判断字符合法，则会进入



当整个字符串还没比完，则会跳入下图代码块，进入下一轮比较

图片包含 日程表

描述已自动生成

如果已经比完，就会跳入到：

文本, 表格

描述已自动生成

其中我们会发现该处调用了sub\_401700函数，下面我们分析这个子函数：

首先该函数进行初始化，将输入字符串的长度计入ecx中，由于esi为0，故肯定不跳转

表格

描述已自动生成

下面进行的函数将a12345…字符串的长度记录在了edi中，对edx进行清零，对循环进行初始化。

图形用户界面, 文本, 应用程序

描述已自动生成

下面代码所实现的功能就是寻找输入字符串的字符在a1234中的位置，并将其记录在如注释所修改的整形数组中（每个int32位，所以位置为esi\*4）

图形用户界面

中度可信度描述已自动生成

当所有的位置都找到后就能结束当前函数，并且同时返回和结束input\_check函数。

Finish部分：

首先展示finish部分的伪C代码：

图形用户界面, 文本, 应用程序

描述已自动生成

回到汇编代码：

文本, 表格

描述已自动生成

第一个block就是初始化部分，由ida可知该部分都是各值进行初始化、为函数开辟空间。

其中esi在进入test之前就被赋值成9，故test的结果在正常情况下必不为0，则jz不起效。

进入下一个block：

图形用户界面, 文本, 应用程序, 表格

描述已自动生成

前两句的作用如注释

ebx自身异或清0，为进入循环初始化

剩下两句就是eax赋值给栈空间对应位置，和没有什么效果的ax和ax交换内容

结合下面的block：

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

而401650行的注释为：

当ecx不为0且edi内的数据不等于eax（既0）时一直比，每次ecx-1，结合上面将ecx设为FFFFFFFF和下面将ecx取反、减一，可知该用法的作用就相当于strlen函数

可得到在00401654行之后的ecx就相当于a1234…字符串的长度。

从00401655开始就是仿射密码的部分：

.text:00401655：首先用eax记录一个整形变量数组的开头位置，在32位程序中整形数据为32位，故而每次往后地址都要加32/8=4，故而在系数里会有ebx\*4

且通过前面分析我们可知该整形数组记录的就是输入各位对应在a1234…字符串中对应位置。

.text:0040165C, .text:0040165F:分别就是位置乘3和加9

.text:00401663：edx清0

.text:00401665: eax 既仿射后的位置除以ecx既字符串长度，将余数放入edx中，作为新的位置

.text:00401667, .text:0040166D:将新位置对应的值赋值回str既输入的字符串当中，完成当前位的仿射变换。

剩余几行既比较完成整个字符串的仿射变化没，没有就继续循环。

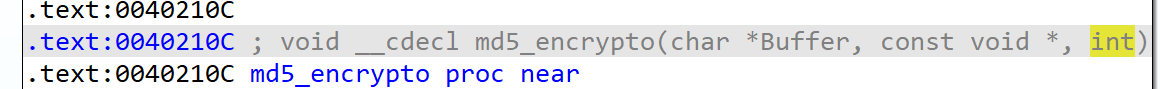
进入下一个代码块：

文本

描述已自动生成

这个代码块的任务时给md5加密函数进行初始化以及调用md5加密函数进行加密，将加密结果和密钥进行比较

由于后面比较部分明显，故而分析前面的md5函数部分:



可看到md5加密函数由三个参数由传参顺序可知int为5，第二个为变换后的输入字符串，第一个既函数的输出

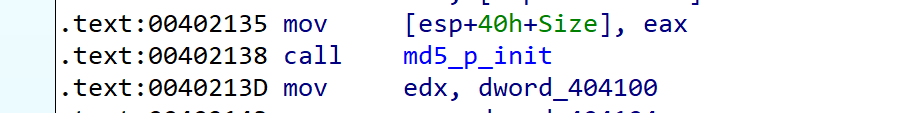
首先开出一个长度40H既64，每个位置为8位的空间

日程表, 瀑布图

中度可信度描述已自动生成

用于中间结果存放和运算

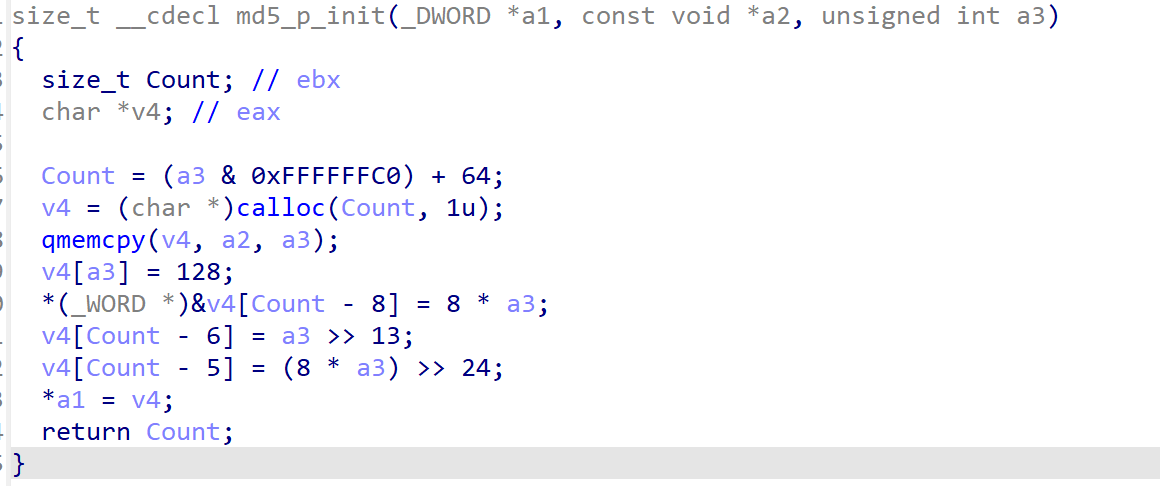
然后函数调用了md5明文初始化的函数：



明文初始化函数内关键部分为：

文本

描述已自动生成



运用伪C代码可知该函数的作用就是利用前5位的信息对明文进行初始化和设置

其中V4[a3]=128或者80H的作用既在明文后填充一个1和若干个0

最后64为的前16位为8\*a3=40就表明了真实明文的长度既从输入字符串中取出的5个字符也即40bit。

回到md5加密模块部分：

文本

描述已自动生成

这几句完成了md5初始链接向量的设置，同时也是md5加密最明显的特征。

下一部分和其对应的伪C代码为：

图形用户界面, 文本, 应用程序, 表格

描述已自动生成

图形用户界面, 文本, 应用程序

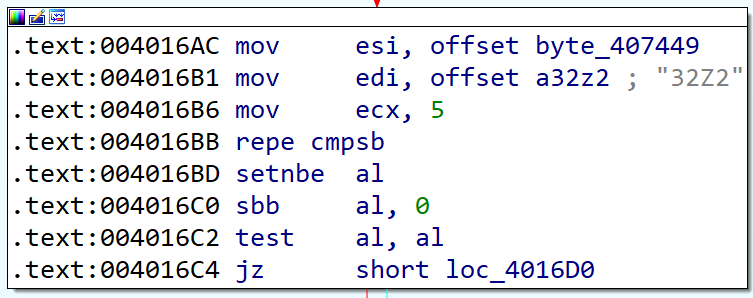
描述已自动生成

即MD5的对明文和链接向量分组加密部分，每次从明文处取512bit，内部调用的函数就是md5的核心函数不作过多解释

最后一部分代码完成大小端转换的功能。

返回md5加密的值。

回到final函数，当前五位的hash加密正确后还需后几位的校验：



由于str输入字符串的地址为00407444，而00407449既str后5个BYTE的位置，可知00407449存的就是输入字符串的后四位，该函数即将仿射变换的后4位的值和“32Z2”比较。

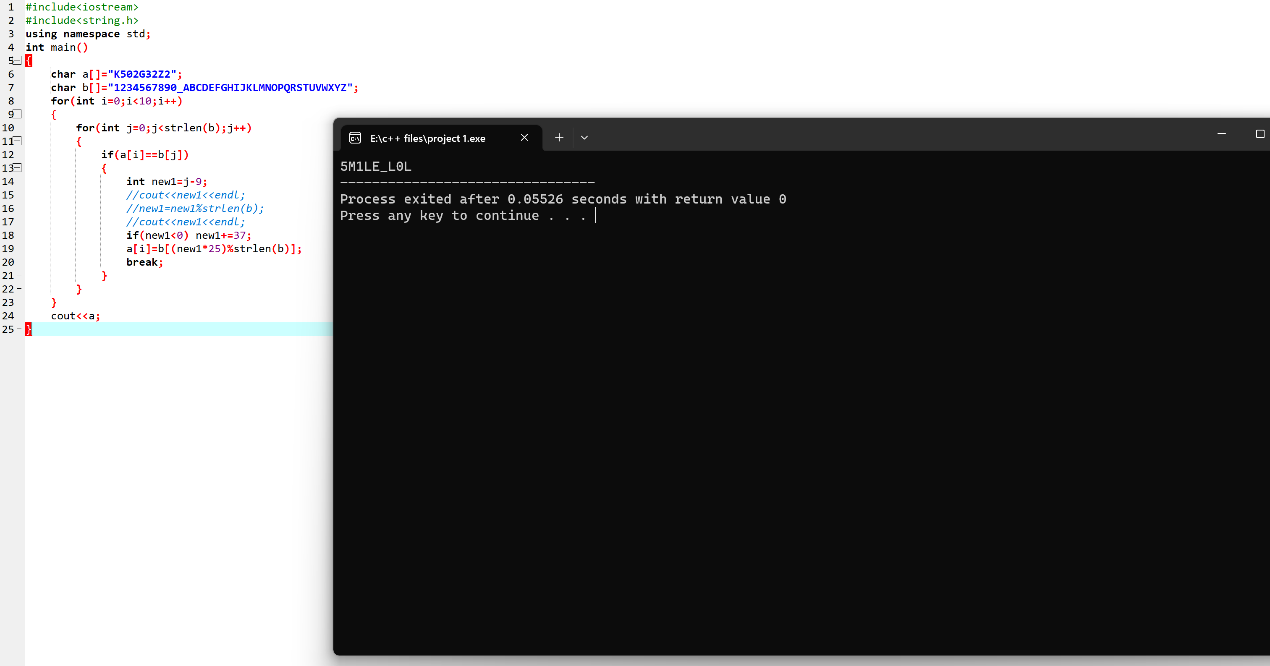
最后一部分既输出验证成功，不做过多解释。



最后寻找flag：

图形用户界面, 应用程序

描述已自动生成



由于有截断存在，故而5M1LE\_L0L后加或不加任意长度的内容都可以

图形用户界面

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

图形用户界面, 应用程序

描述已自动生成