

Signature Matcher 使用说明

- Signature Matcher 是一个 DLL 动态库，需要注入进程使用。（怎么注入？自己想办法）
- Signature Matcher 有 x86 和 x64 两个版本
- Signature Matcher 需要 [Microsoft .NET Framework](#) 4.0及以上支持，需要 [VC++2013\(x86/x64\)](#)运行时库支持。
- 以下是 Signature Matcher 的启动界面：（x86/x64版本界面一致）



- 窗口上部控件分别为：【关闭 C】、【最小化 N】、【窗口移动】、【控制面板 D】。



- 【控制面板】接受鼠标悬停，以打开或关闭控制面板。
- 窗口不能改变大小

- 占据窗口大部分区域的文本控件是【信息输出控件】，双击将清除所有数据
- 鼠标单击【信息输出控件】会关闭已经打开的【控制面板】
- 【控制面板】有两个状态，分别如下显示：



以上为选择【输入特征码 B】选项时的状态



以上为选择【指定特征码文件 F】选项时的状态

- 单击【关闭 C】将关闭【控制面板】。另外，窗口也接受 ESC 按键关闭【控制面板】
- 单击【匹配 M】将开始匹配过程，同时关闭控制面板
- 【Trace 信息设置】用于匹配过程中的信息输出控制


☆



【详细信息 L】输出匹配

过程中的所有详细信息，包括匹配工作细节，不推荐选中，一般用于调试功能

☆



【仅错误 E】指定输出匹配过程中的错误，仅会输出导致匹配失败的错误原因



✧ **【No Trace Q】** 不输出匹配过程的任何信息

- 双击 **【文件 S】** 的文本框，或单击 **【打开 O】** 将打开文件选择窗口。文件只接受单选
- **【文件】** 的文本框也接受手工输入特征码文件路径
- 全窗口 **接受文件拖放**，文件被拖放后，自动开始分析
- **【指定特征码文件】** 时，接受.sig 特征码脚本文件或.atom 中间原子文件的解析
- ATOM 文件有 x86 与 x64 版本之分，具有独有的文件格式
- 文件解析其实可以无视文件名后缀，但推荐为.sig 与.atom
- SIG 文件应为 ASCII 纯文本文件，内容需要符合特征码脚本的书写格式
- ATOM 文件的优点在于隐藏了脚本书写细节，是对脚本的非文本化，可以算是简单加密吧
- 当 **【自动生成中间原子 A】** 被选中时，解析 SIG 文件后将自动生成同名的.atom 中间原子文件
- 当 **【生成 C++11 格式头文件 Z】** 被选中时，匹配结果将生成 C++11 格式，否则生成 C++98 格式

```
#pragma once

enum : size_t
{
    K_Apple_Client                = 0x00809740, //Client __stdcall .()
    K_Apple_PlayerInfo            = 0x00701CA0, //PlayerInfo __thiscall .[System]()
};

enum : unsigned __int32
{
    K_Apple_Player_Equipment      = 0x00000258, //Equipment = PlayerInfo + .
    K_Apple_Player_Inventory      = 0x000002DC, //Inventory = PlayerInfo + .
    K_Connector_TcpClient         = 0x000000AC, //TcpClient = Connector + .
};

enum : unsigned __int8
{
    K_ContainerOff                = 0x1C, //ContainerOff = Inventory + .
};
```

C++11 格式 enum 头文件

```

#pragma once

enum
{
    K_Apple_Client           = 0x00809740, //Client __stdcall .()
    K_Apple_PlayerInfo       = 0x00701CA0, //PlayerInfo __thiscall .[System]()

    K_Apple_Player_Equipment = 0x00000258, //Equipment = PlayerInfo + .
    K_Apple_Player_Inventory = 0x000002DC, //Inventory = PlayerInfo + .
    K_Connector_TcpClient    = 0x000000AC, //TcpClient = Connector + .

    K_ContainerOff           = 0x1C, //ContainerOff = Inventory + .
};
  
```

C++98 格式 enum 头文件



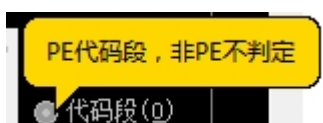
- 在【输入特征码】状态下，与 可缺省，当【起始地址】与【结束地址】缺省时，匹配范围默认为所在进程主模块(EXE)的 PE 映像范围
- 当【起始地址】与【结束地址】不空或不为 0 时，匹配范围被指定
- 【起始地址】与【结束地址】接受手工输入



- 【特征码输入控件】接受手工输入的特征码脚本语法，脚本语法将在之后介绍。**注意：**不仅仅是特征码语法，接受的是脚本语法



- 【全映像 I】选中时，作用如下：
 - ✧ 当【结束地址】获得焦点且为空或为 0 时，【起始地址】为 PE 有效起始时，将被自动填充为 PE 映像结束地址
 - ✧ 打开【模块】选择窗口时，模块被显示为 PE 全映像范围



- 【代码段 O】选中时，作用如下：

- ✧ 当【结束地址】获得焦点且为空或为 0 时，【起始地址】为 PE 有效起始时，将被自动填充为 PE 代码段结束地址
- ✧ 打开【模块】选择窗口时，模块被显示为 PE 代码段范围
- 单击【模块 A】将打开模块地址范围选择窗口，同时隐藏主窗口，直到选择一条范围或单击确定/关闭

起始地址	结束地址	模块路径
00007FF6E23A0000	00007FF6E23DE000	E:\work\SignatureMatcher\x64\Release\tes...
00007FFAE3610000	00007FFAE37BA000	C:\Windows\SYSTEM32\ntdll.dll
00007FFAE1340000	00007FFAE147A000	C:\Windows\system32\KERNEL32.DLL
00007FFAE0D80000	00007FFAE0E8F000	C:\Windows\system32\KERNELBASE.dll
00007FFAD4760000	00007FFAD47A4000	E:\work\SignatureMatcher\x64\Release\Sig...
00007FFACEB40000	00007FFACEBE6000	C:\Windows\SYSTEM32\MSVCP120.dll
00007FFACD6D0000	00007FFACD7BF000	C:\Windows\SYSTEM32\MSVCR120.dll
00007FFAE3490000	00007FFAE3601000	C:\Windows\system32\USER32.dll
00007FFAD6800000	00007FFAD6864000	C:\Windows\SYSTEM32\mscoree.dll
00007FFAE16C0000	00007FFAE1804000	C:\Windows\system32\GDI32.dll
00007FFAE1A60000	00007FFAE1A94000	C:\Windows\system32\IMM32.DLL
00007FFAE1920000	00007FFAE1A59000	C:\Windows\system32\MSCTF.dll
00007FFAE10B0000	00007FFAE1157000	C:\Windows\sy
00007FFAE1C20000	00007FFAE1CC5000	C:\Windows\sys

模块地址范围选择窗口

- ✧ 每次打开选择窗口都会刷新进程模块
- ✧ 窗口无法改变大小和移动，将屏幕居中显示，半透明
- ✧ [起始地址]、[结束地址]、[模块路径]接受指定简单顺序排序
- ✧ 单击一条范围，确定之（或双击一条范围）将关闭此窗口，并将选择的内存范围填入【起始地址】与【结束地址】
- ✧ 单击[关闭 X]将仅仅关闭窗口，重新显示主窗口

起始地址	结束地址	模块路径
00007FF6E23A0000	00007FF6E23C4A00	E:\work\SignatureMatcher\x64\Release\tes...
00007FFAE3610000	00007FFAE3739000	C:\Windows\SYSTEM32\ntdll.dll
00007FFAE1340000	00007FFAE1453A00	C:\Windows\system32\KERNEL32.DLL
00007FFAE0D80000	00007FFAE0E6D600	C:\Windows\system32\KERNELBASE.dll
00007FFAD4760000	00007FFAD4783E00	E:\work\SignatureMatcher\x64\Release\Sig...
00007FFACEB40000	00007FFACEB95000	C:\Windows\SYSTEM32\MSVCP120.dll
00007FFACD6D0000	00007FFACD777800	C:\Windows\SYSTEM32\MSVCR120.dll
00007FFAE3490000	00007FFAE3528600	C:\Windows\system32\USER32.dll
00007FFAD6800000	00007FFAD6853000	C:\Windows\SYSTEM32\mscoree.dll
00007FFAE16C0000	00007FFAE17DAC00	C:\Windows\system32\GDI32.dll
00007FFAE1A60000	00007FFAE1A84E00	C:\Windows\system32\IMM32.DLL
00007FFAE1920000	00007FFAE19FE000	C:\Windows\system32\MSCTF.dll
00007FFAE10B0000	00007FFAE1140000	C:\Windows\sy
00007FFAE1C20000	00007FFAE1CAE800	C:\Windows\sys

- ✧ 这是【代码段】

选中时的范围显示，注意结束地址的区别

- 单击【内存块 Z】将打开内存块地址范围选择窗口，同时隐藏主窗口，直到选择一条范围或单击确定/关闭

起始地址	结束地址	模块路径
0000000010000000	000000001005D000	C:\Windows\system32\WN_INPUT.IME
000000007F649000	000000007F64A000	
000000007FFE0000	000000007FFF0000	
00000060336F0000	0000006033700000	
0000006033700000	0000006033701000	
0000006033710000	000000603371F000	
0000006033720000	0000006033820000	
0000006033820000	0000006033824000	
0000006033830000	0000006033831000	
0000006033840000	0000006033842000	
0000006033850000	00000060338CE000	
00000060338D0000	00000060338EA000	
00000060338F0000	000000603390A000	
0000006033910000	0000006033911000	

内存块地址范围选择窗口

- ◇ 每次打开选择窗口都会刷新进程内存块
- ◇ 内存块对应模块时，模块路径栏会显示，否则为空
- ◇ **注意：**模块内存块的范围无视【全映像】与【代码段】是否选中，显示的是有效内存块的全范围
- ◇ 其它操作与以上同
- 全窗口接受文本拖放，文本被拖放后，视作特征码脚本文本，自动开始分析
- **注意：**由于托管 DLL 无法卸载完全的原因，Signature Matcher 卸载后无法再注入，只能注入一次
- 以下是解析结果示例

```
-----匹配工作开始-----
>> K_A[redacted]_Send = 0111A1F0 = 0111A62C 671ms
>> K_A[redacted]_Connector = 0C 532ms
>> K_A[redacted]_DefaultCryptor = 08 187ms
>> K_C[redacted]_h = 004737C0 63ms
>> K_S[redacted]_Mok = 007D285E 203ms
>> K_A[redacted]_ClientS = FFFF7EC
>> K_A[redacted]_Connector = FFFF7D4 172ms
>> K_A[redacted]_d = 00000150
>> K_C[redacted]_tcp[redacted] = 000000AC 187ms
>> K_A[redacted]_ClientS = 00000258
>> K_C[redacted]_ff = 1C 156ms
>> K_C[redacted]_MPS = 007FC250 328ms
成功写入H文件(0ms): [redacted].h
-----匹配工作结束----- 2546ms
```

- 以下是一条语法非法的特征码，以及一条没写精确的特征码的匹配输出

第一条。第 6 行 33 列的 0-02 应该写成 00-02

第二条。第 6 行的 B9\$L 应该写成\$R

```
-----匹配工作开始-----
第6行33列:hexhex配对失败
匹配失败

8B 96@L@R <D K_..._Equipment Equipment = PlayerInfo + ..>
8B 42@L$3R 04
8D BE@L$3R <D K_..._Equipment>
8B CF$R
C7 85 . . . FF FF . . . 0-02

原值007FC290: 00000007 @L引用值007FC27E: 01无法匹配
原值007FC440: 00000007 @L引用值007FC42E: 01无法匹配
匹配失败

<A K_...NPC>
8B 4D@L 08
C7 45 FC 00 00 00 00
85 C9$L$R
0F 84 . . 01-02 00 00
83 B9$L . . . 00 00 01

-----匹配工作结束----- 1937ms
```

特征码语法介绍

特征码语法是参考正则表达式的语法，重新设计以符合特征码查找要求

特征码语法由以下元素组成：

- **Blank 空白**
 - ❖ 空白符号由【空格】、【制表符】、【换行符】、【回车符】组成
 - ❖ 如无特别说明，【空白】允许出现在特征码的任意处
 - ❖ 如无特别说明，【空白】在特征码解析过程中被忽略

- **Quote 引用**

- ❖ 引用串以【"】号起始，【"】号结束
- ❖ 引用串为 Unicode 格式时，需要在起始【"】前加【L】或【l】
- ❖ 引用串内如果需要再包含【"】号，请使用【\"】。
- ❖ 引用串内将【空白】视作引用串的内容
- ❖ 引用串内的【换行】与【回车】请使用【\r】、【\n】。支持转义字符

```

1. B8 "ascii_quote"      #相当于定位 push XXXX,   XXXX 指向"ascii_quote"
2. B8 L"unicode_quote"  #相当于定位 push XXXX,   XXXX 指向 L"ascii_quote"
3. B8 l"unicode_quote"  #L 符允许小写
4. B8 "abc\"abc"        #引用串内包含"号
5. B8 "abc abc"         #引用串内包含空白
6. B8 "abc
   abc"                 #引用串内的使用换行与回车
7. B8 "abc\r\nabc"      #引用串内使用转义的换行与回车

```

➤ Record 特征点

- ❖ 特征点以【<】号起始，【>】号结束
- ❖ 特征点串第一个非空白符如果为【^】，则这是一个【偏移标记】
- ❖ 【偏移标记】存在时，特征点值提取结果将减去匹配块的起始地址
- ❖ 特征点串第一个非【偏移标记】、非空白符必须为下列字符的其中之一(无视大小写)
 - ✧ **A** 表示此点是一个地址，此点占用 0 byte
 - ✧ **F** 表示此点是一个偏移， 占用 4 byte
 - ✧ **Q** 表示此点是一个 QWORD，占用 8 byte
 - ✧ **D** 表示此点是一个 DWORD，占用 4 byte
 - ✧ **W** 表示此点是一个 WORD， 占用 2 byte
 - ✧ **B** 表示此点是一个 BYTE， 占用 1 byte
- ❖ 特征点串第二个非空白符开始视作【特征点名称】，由【空白】结束，名称可以为空
- ❖ 【特征点名称】允许的字符无特别限制，但实际应用中有有限制
- ❖ 【特征点名称】之后第一个非空白符至结束视作【特征点注释】
- ❖ 【特征点注释】中【换行符】【回车符】将被替换为【空格】
- ❖ 【特征点注释】中的前缀空白与后缀空白将被忽略

```

1. <A> 'MZ'              #相当于定位模块起始，无名字无注释
2. <A MZStart> 'MZ' <A MZEnd>
   #有名字无注释  MZStart == 0x00400000; MZEnd == 0x00400002;
3. <^A MZStart> 'MZ' <^A MZEnd>
   #有偏移标记  MZStart == 0;    MZEnd == 2;
4. E8 <F>                #偏移
5. B8 <D>                #DWORD

```

```

6. <A MZStart 这是 MZ 开始> #有名字有注释
7. <A MZ::Start>             #名字语法合法但 Signature Matcher 自动生成头文件不合法
8. <A MZS
   这里是
   MZ 开始>                  #允许换行符与回车符

```

➤ Range 范围指示

- ❖ 范围指示只能用于【点】【常量串】【组合】
- ❖ 范围指示不能重复使用
- ❖ 范围最小为 0，最大为 MAX，x86 下为 0x7FFFFFFF，x64 下为 0x7FFFFFFF_FFFFFFFF
- ❖ 范围指示必须为以下几种类型
 - ✧ * 表示匹配范围从 0 到 MAX == {0,MAX} == {,}
 - ✧ + 表示匹配范围从 1 到 MAX == {1,MAX} == {1,}
 - ✧ ? 表示匹配范围从 0 到 1 == {0,1} == {,1}
 - ✧ {N,M} 表示匹配范围从 N 到 M
 - ✧ {N} 表示匹配范围从 N 到 N
 - ✧ {,M} 表示匹配范围从 0 到 M
 - ✧ {N,} 表示匹配范围从 N 到 MAX
 - ✧ {} 表示匹配范围从 0 到 MAX == *
- ❖ N 与 M 间的分隔符允许为【,】【-】【~】【|】【:】，但一般为【,】
- ❖ 注意 N 与 M 的不得超过范围,否则识别失败
- ❖ 注意匹配模式是【非贪婪】的
- ❖ 除分隔符外，{}间不得有除【空白】外的非十六进制字符，否则识别失败
- ❖ N 与 M 将以十六进制识别，允许 N > M

```

1. .* .+ .? .{1,2} .{1} .{,2} .{2,} .{,}
2. .{2,1}                #允许 N>M

```

➤ String 字符串

- ❖ 字符串以【'】号起始，【'】号结束
- ❖ 字符串为Unicode格式时，需要在起始【'】前加【L】或【l】
- ❖ 字符串如果需要再包含【'】号，请使用【\'】
- ❖ 字符串内将【空白】视作字符串的内容
- ❖ 字符串内的【换行】与【回车】请使用【\r】、【\n】。支持转义字符

```

1. 'MZ'                  #匹配字符串"MZ"
2. L'abc'                #匹配字符串"abc"

```

3. l'abc'	#L 符允许小写
4. 'abc\'abc'	#字符串中转义'符
5. 'abc abc'	#字符串中不忽略空白符
6. 'abc\r\nabc'	#字符串中换行回车使用转义
7. 'abc abc'	#字符串中换行回车不使用转义

➤ MarkRef 附标记与附引用

- ❖ 【附标记】由两个十六进制、【@】符号，【L】或【R】符号组成
- ❖ 【附引用】由两个十六进制、【\$】符号，[1-F]十六进制数索引，【L】或【R】符号组成
- ❖ 【附标记】与【附引用】只标识一个十六进制数值，不能应用【范围】
- ❖ 【附标记】与【附引用】可以同时存在，此时，无视书写顺序，一律认为【附标记】在【附引用】之前
- ❖ 【L】或【R】可以同时存在，此时，无视书写顺序，一律认为【L】在【R】之前
- ❖ 【附引用】的【\$】后十六进制数可以缺省，缺省值为1
- ❖ 【R】标记模糊处理0、1、2位。【L】标记模糊处理3、4、5位

如50@R，匹配50-57。C8@L，匹配C8-F8

模糊匹配的相关说明参考【常量串】

1. 50@R	#匹配 push r32 或 push r64，并标记 r32/r64
2. 58\$R	#匹配 pop r32 或 pop r64，同时校验 r32/r64 是否与之前的标记相同，索引缺省
3. 8B C3\$R@L	#标记与引用同时存在，实际被识别为 C3@L\$R
4. 8B C3@R@L	#L 与 R 同时存在，实际被识别为 C3@L@R

➤ Ucbit 常量串

- ❖ 常量串由两个十六进制字符组成，允许中间有【空白】存在
- ❖ 常量串的表达也可以由【:】后紧接一个字符表示此字符
- ❖ 注意当使用【:】表示字符时，紧接的【空白】不被忽略
- ❖ 常量串可以通过以下操作符连接
 - ✧ - 连接操作符。00-FF 表示匹配从 00 到 FF 的任意值
 - ✧ | 或者操作符。00|FF 表示匹配 00 或者 FF。
 - ✧ & 模糊操作符。注意，这个操作符是按位模糊处理，一般用于寄存器匹配

处理的关键：maskA 的位与 maskB 的位相异时，即此位 0/1 都能匹配

73&70 即 0111 0011 & 0111 0000，匹配 70、71、72、73

7C&71 即 0111 1100 & 0111 0001，匹配 70、71、74、75、78、79、7C、7D

其效果等同于 7D&70

半字节匹配例如 7F&70，全字节匹配例如 FF&00

模糊操作符可接【R】【L】标记

【R】标记模糊处理 0、1、2 位。【L】标记模糊处理 3、4、5 位

❖ 常量串允许通过操作符无限次连接，但请注意值冲突

```
1. 50                #匹配 push eax 或 push rax
2. 5 0              #允许空白
3. :A :             #匹配 A 字符，空格符，相当于 41 20
4. 00-41            #匹配 00 至 41，注意 00 不能写成 0
5. 00|41            #匹配 00 与 41
6. 00-41|45-48|88|90 #匹配 00 至 41 与 45 至 48 与 88 与 90，操作符可无限连接
7. 51&R            #匹配 push r32 或 push r64
8. 50&57            #与上同，写法不同而已
9. 51|50&57        #与上同，这是一个技巧，指示当前源匹配为 51。现在有新语法推荐使用 51&R
```

➤ Collection 组合

- ❖ 组合由【[】号起始，【]】号结束
- ❖ 组合的第一个非空白字符如果为【^】，匹配值组将被翻转
- ❖ 组合内容由【常量串】组成
- ❖ 除翻转操作外，组合一般可以由操作符连接的【常量串】就可以完成
- ❖ 组合内部的模糊操作符不可接【R】【L】标记
- ❖ 组合的其它条件参考【常量串】说明。

```
1. [505153]        #相当于 50|51|53
2. [^00CC]          #匹配除 00 与 CC 外的所有值
3. [50&57]          #与 50&57 相同
```

➤ Dot 点

- ❖ 点相当于 00-FF，但任意匹配点匹配处理上有优化，效率较高

```
1. ....            #相当于.{4} 00-FF{4}
```

➤ Note 注释

- ❖ 注释以【#】号开始，直到行尾
- ❖ 注释在解析过程中被忽略

```
1. <F show_msg>      #bool __stdcall show_msg(char* msg);
2. <F show_msg bool __stdcall show_msg(char* msg)>
   #请注意与特征点注释的区别，特征点注释在解析过程中不被忽略，跟随解析结果
```

➤ Notice 额外说明

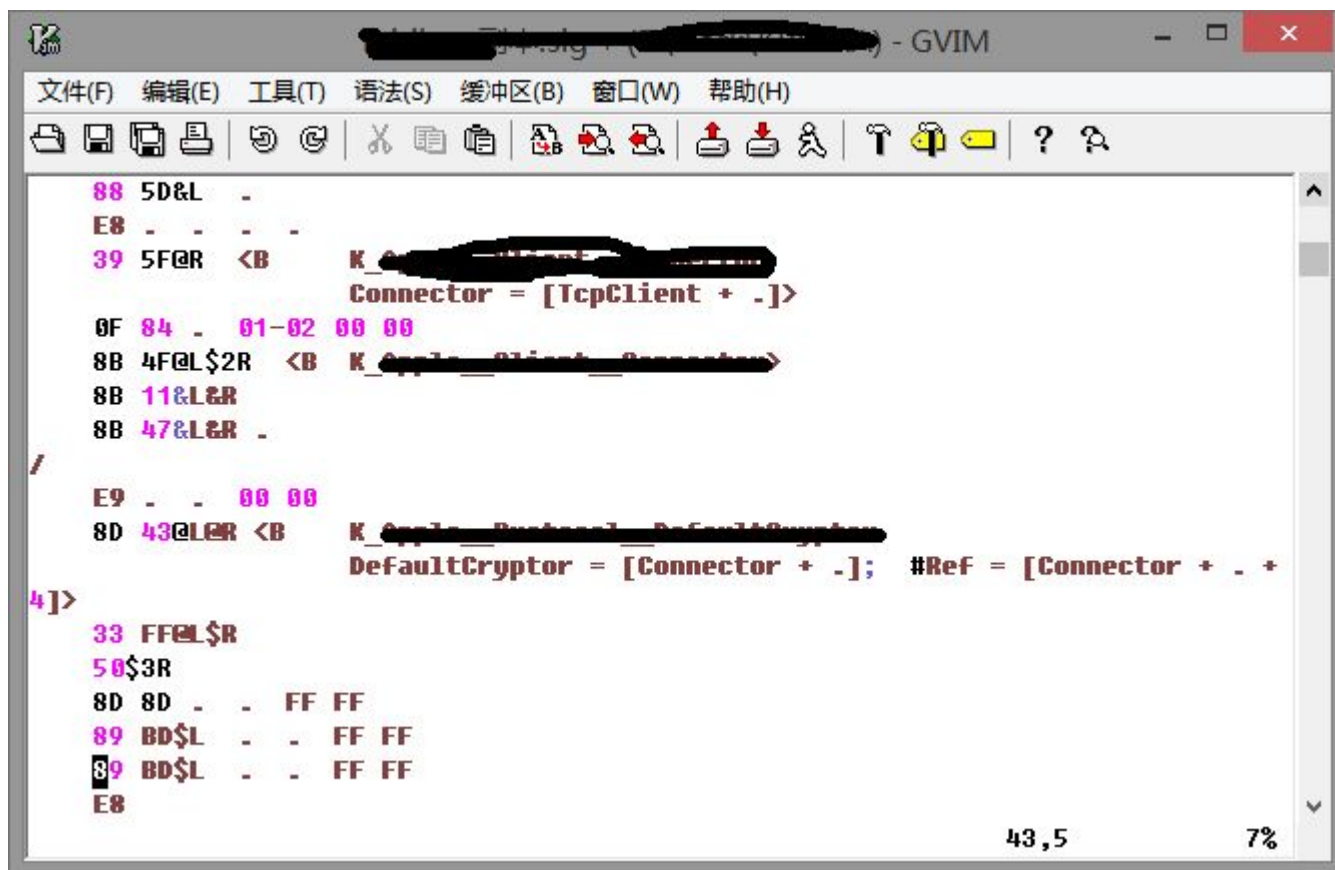
- ❖ 如果特征码串中无特征点，自动在串头部添加<A>
- ❖ 【特征点名称】允许重名，将判定重名特征点结果是否一致
- ❖ 特征码串非法时，会指示非法位置
- ❖ 书写特征码时，应充分利用模糊处理、不定长匹配，以使特征码长效健壮
- ❖ 书写特征码的技巧，需要长期的经验积累，这里无法一一叙述

特征码脚本格式说明

- 特征码脚本只有两种类型：【特征码串】、【范围指示】以及【/】分隔标记
- 每个【特征码串】与【范围指示】使用【/】标记起始与结束
- 用于分隔的起始标记可省略，在脚本结尾，结束标记也可以省略
- 【范围指示】必须使用【@】或【\$】作为起始非空白字符
- 【范围指示】可以指示多块
- 【范围指示】优先识别为模块，当模块不存在时，识别为【范围组】
- 【范围组】必须成对出现，且不能有交集
- 【范围指示】为标准 PE 文件时，【@】表示全映像，【\$】表示代码段
- 【@】与【\$】应用于【范围组】意义相同

```
GVIM
文件(F) 编辑(E) 工具(T) 语法(S) 缓冲区(B) 窗口(W) 帮助(H)
[Icons]
/
88 5D&L .
E8 . . .
39 5F@R <B [Redacted]
[Redacted] [Redacted]
0F 84 . 01-02 00 00
8B 4F@L$2R <B K [Redacted]
8B 11&L&R
8B 47&L&R .
/
/
E9 . . 00 00
8D 43@L&R <B K [Redacted]
DefaultCryptor = [Connector + .]; #Ref = [Connector + . +
4]>
33 FFE@L$R
50$3R
8D 8D . . FF FF
89 BD$L . . FF FF
89 BD$L . . FF FF
E8
/
34,14 6%
```

起始与结束标记完整的示例



```
88 5D&L .
E8 . . . .
39 5F@R <B K_...
Connector = [TcpClient + .]>
0F 84 . 01-02 00 00
8B 4F@L$2R <B K_...
8B 11&L&R
8B 47&L&R .
/
E9 . . 00 00
8D 43@L&R <B K_...
DefaultCryptor = [Connector + .]; #Ref = [Connector + . +
4]>
33 FF@L$R
50$3R
8D 8D . . FF FF
89 BD$L . . FF FF
89 BD$L . . FF FF
E8
```

43,5 7%

省略起始标记，省略脚本结尾的结束标记，同样合法的示例

1. @xxx.dll #指示一个模块，也可写成@xxx
2. @xxx.dll@yyy.dll@zzz.dll #指示多个模块
3. @400000@401000 #指示一个范围
4. @400000@401000@401000@402000 #指示多个范围