

# # 任意RAT改加载器过云沙盘

## # 前言

沉寂许久，时隔良久，赴当年之约，感触良多。百川赴海返潮易，一叶报秋归树难。今天分享一篇任意RAT Bypass杀软的思路，**提供思路仅供参考，不提供任何源码**。文章中提到的地方若有错误的地方请指正。

直入主题，目前RedTeam的主流RAT无非就Cobalt Strike。其他的RAT大多基于C/C#开发（这里不做讨论），小众点的RAT，BUG，兼容性等一系列体验感不是很好。而Cobalt Strike的特征已经被各大厂商记录到特征库中，所以用Cobalt Strike做Bypass的效果并不是很友好。

此次测试使用的杀毒软件是国内企业杀软，以及Tinder（自行翻译，懂得都懂，毕竟ByPass见光死），RAT使用的是gh0st的变种。（**使用gh0st原因简单阐明：特征多，容易被杀**）

## # 正文

### 0X00

这里笔者拿国内的老版本RAT进行测试。仅提供研究思路，不提供任何源码。

ShellBase、ShellCode、RomteLoad工程源码使用VS2010编译，念旧。在打开的时候默认VS2019，见谅！Gh0st源码使用VC++6.0编译。

之前研究目前主流的RAT以及国内一些老版本的RAT，这里笔者就拿国内的老版本RAT进行测试。因为较CS，国内老版RAT特征多能够更好的体现笔者的思路。

### 0X01

国内的RAT是基于gh0st以及gh0st的变种。知名的如大灰狼、灰鸽子等。其中，主流的写法是功能插件化（功能插件化，这一点跟CS差不多，插件化的优越性大家都知道的，（简单概括就是：方便、快捷），然后用外壳内存加载功能插件，主机上线后通过IOCP协议传输来调用功能，形成Payload – Server ==> Client为一体的工作环境。

服务控制端过程大概是：

功能集成Server.dll => 16进制加密Server.dll => 外壳Install解密内存加载Server.dll => 生成二进制可执行文件

### 0X02

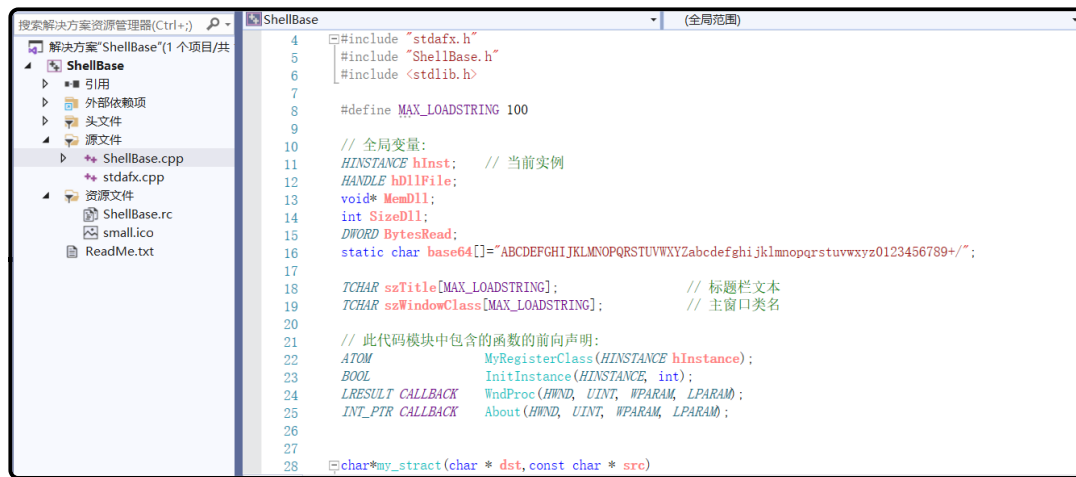
经分析发现现在的杀毒对于这种写法基本看的很死 比如某企业杀软会把一个exe文件区块化去查杀 导致实际操作去Bypass时，做好的Server端一次Bypass基本2-5天就被再次查杀。

原因：Bypass基本思路都是 杀小红伞基本去处理输入表 杀QVM一般处理代码段。所以，从根本上来看是没改变这种写法的

笔者的思路是通过下载把Server.dll直接下载到内存里，然后再执行。

过程大概是：Server.dll => 16进制加密Server.dll => 把Server.dll放到下载地址然后base64加密 => 加载器解密地址下载到内存执行

**这种思路对Cobalt Strike也适用**

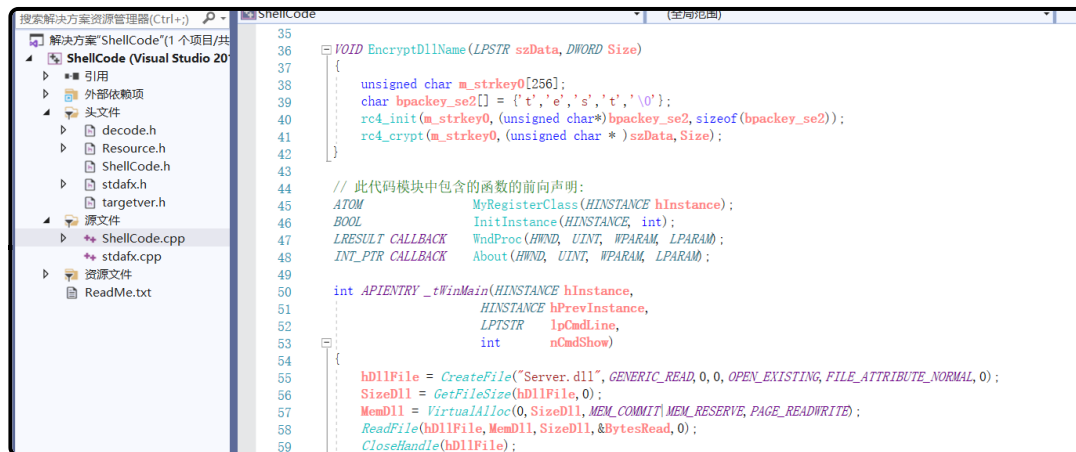


```

4  #include "stdafx.h"
5  #include "ShellBase.h"
6  #include <stdlib.h>
7
8  #define MAX_LOADSTRING 100
9
10 // 全局变量:
11 HINSTANCE hInst;    // 当前实例
12 HANDLE hDllFile;
13 void* MemDll;
14 int SizeDll;
15 DWORD BytesRead;
16 static char base64[]="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/";
17
18 TCHAR szTitle[MAX_LOADSTRING];    // 标题栏文本
19 TCHAR szWindowClass[MAX_LOADSTRING];    // 主窗口类名
20
21 // 此代码模块中包含的函数的前向声明:
22 ATOM MyRegisterClass(HINSTANCE hInstance);
23 BOOL InitInstance(HINSTANCE, int);
24 LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
25 INT_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);
26
27
28 char*my_struct(char * dst,const char * src)

```

ShellBase是写的对地址进行base64加密的工具



```

35
36 VOID EncryptDllName(LPSTR szData, DWORD Size)
37 {
38     unsigned char m_strkey0[256];
39     char bpackkey_se2[] = {'t','e','s','t','\0'};
40     rc4_init(m_strkey0, (unsigned char*)bpackkey_se2, sizeof(bpackkey_se2));
41     rc4_crypt(m_strkey0, (unsigned char *)szData, Size);
42 }
43
44 // 此代码模块中包含的函数的前向声明:
45 ATOM MyRegisterClass(HINSTANCE hInstance);
46 BOOL InitInstance(HINSTANCE, int);
47 LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
48 INT_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);
49
50 int APIENTRY _tWinMain(HINSTANCE hInstance,
51                       HINSTANCE hPrevInstance,
52                       LPTSTR lpCmdLine,
53                       int nCmdShow)
54 {
55     hDllFile = CreateFile("Server.dll", GENERIC_READ, 0, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);
56     SizeDll = GetFileSize(hDllFile, 0);
57     MemDll = VirtualAlloc(0, SizeDll, MEM_COMMIT|MEM_RESERVE, PAGE_READWRITE);
58     ReadFile(hDllFile, MemDll, SizeDll, &BytesRead, 0);
59     CloseHandle(hDllFile);

```