

外挂开发

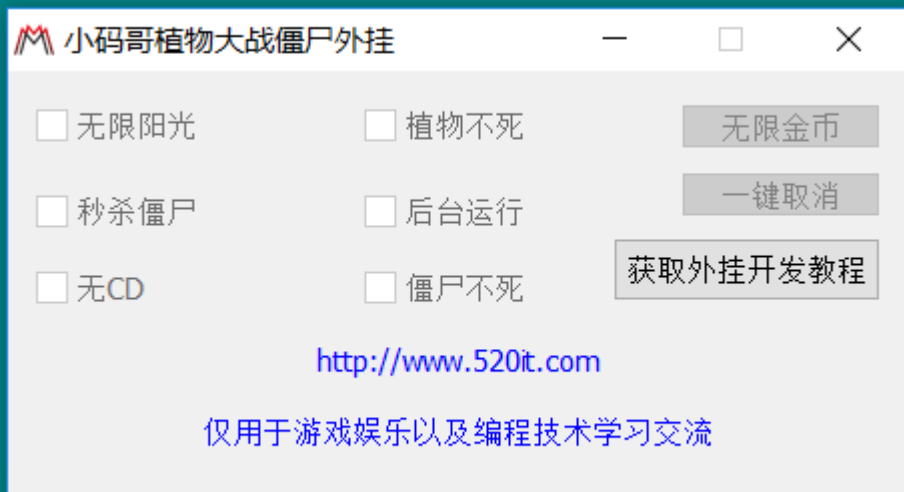
@M了个J

<https://weibo.com/exceptions>

<https://github.com/CoderMJLee>



实力IT教育 www.520it.com



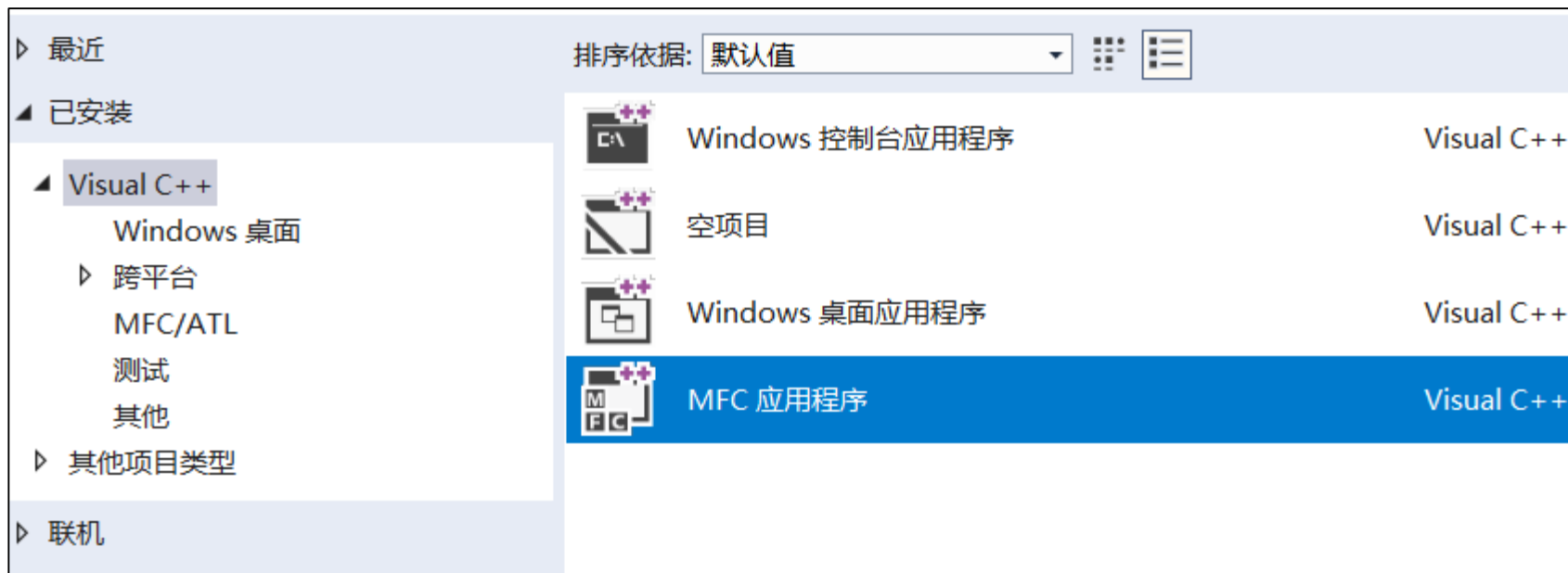
■ 外挂界面

■ 事件处理

■ 跨进程访问

- Windows平台的桌面开发
 - C++：MFC、Qt
 - C#：WinForm、WPF
- 我们课程选用最古老的MFC，不用引入其他外部的框架
- 要先安装好MFC组件

创建MFC项目



MFC 应用程序

应用程序类型选项

应用程序类型

文档模板属性

用户界面功能

高级功能

生成的类

应用程序类型(T):

基于对话框

应用程序类型选项:

- ☐ 选项卡式文档(B)
- ☐ 文档/视图结构支持(V)
- ☒ 安全开发生命周期(SDL)检查(C)

基于对话框的选项(I):

<无>

复合文档支持:

<无>

文档支持选项:

- ☐ 活动文档服务器(A)
- ☐ 活动文档容器(D)
- ☐ 支持复合文件(U)

项目样式:

MFC standard

视觉样式和颜色(Y):

Windows Native/Default

☐ 启用视觉样式切换(C)

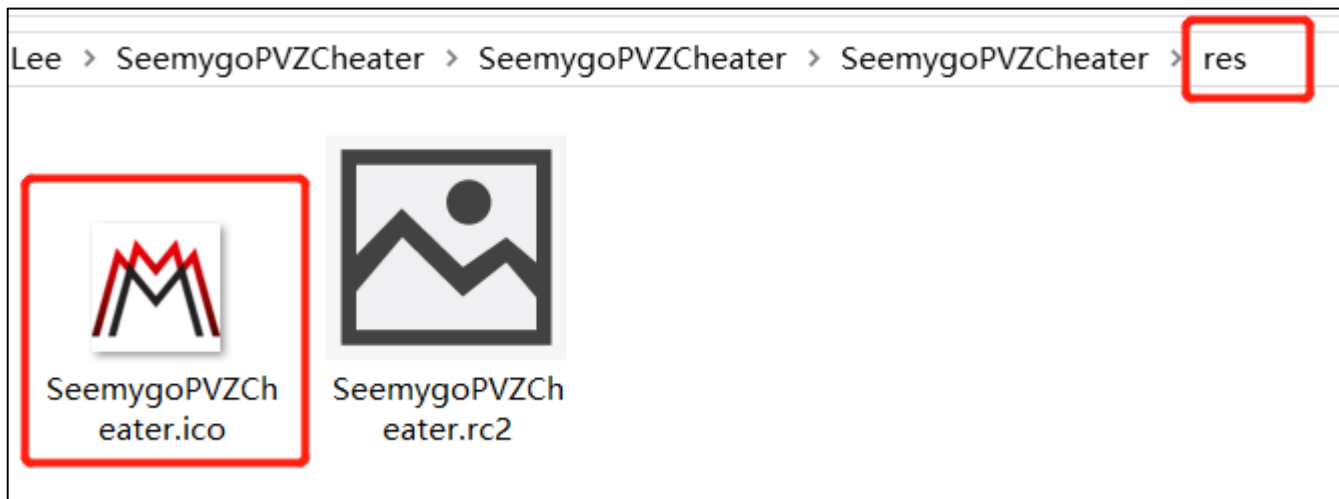
资源语言(L):

English (United States)

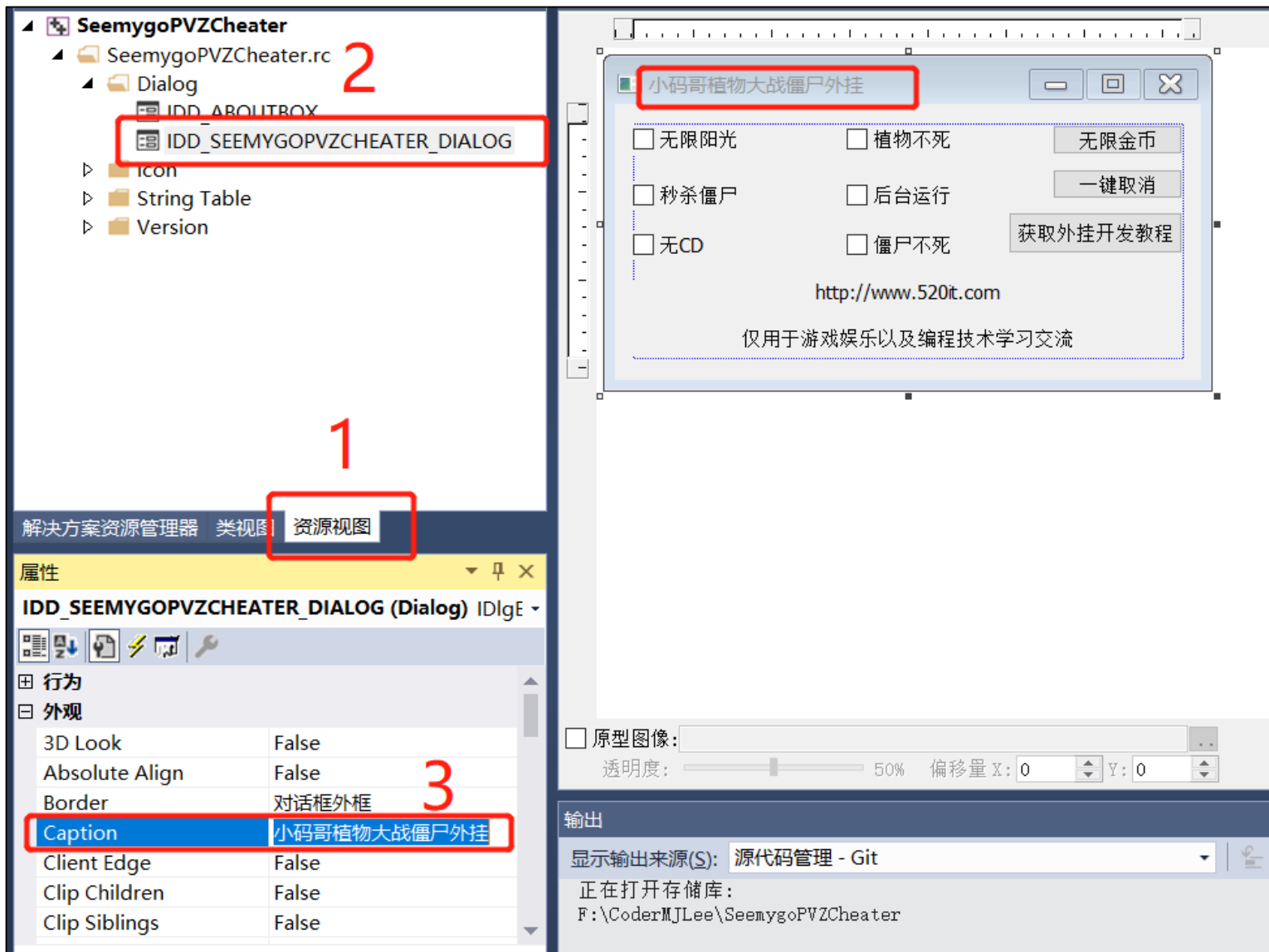
使用 MFC:

在共享 DLL 中使用 MFC

- 替换掉res目录下的ico文件即可



修改窗口标题



打开工具箱

视图(V)	项目(P)	生成(B)	调试(D)	团队(M)
<>	代码(C)			
↻	打开(O)			
	打开方式(N)...			
🔗	解决方案资源管理器(P)	Ctrl+Alt+L		
👥	团队资源管理器(M)			
📁	服务器资源管理器(V)	Ctrl+Alt+S		
🔖	书签窗口(Q)	Ctrl+K, Ctrl+W		
🔗	调用层次结构(H)	Ctrl+Alt+K		
🔗	类视图(A)	Ctrl+Shift+C		
🔗	代码定义窗口(D)			
🔗	对象浏览器(J)	Ctrl+Alt+J		
🔗	错误列表(I)			
🔗	输出(O)	Ctrl+Alt+O		
🔗	任务列表(K)			
🔗	工具箱(X)	Ctrl+Alt+X		

- TRACE函数：类似于C语言的printf，只能在DEBUG调试模式下看到打印信息（F5启动）

```
TRACE("age is %d\n", 20);
```

- AfxMessageBox函数

```
CString str;  
str.Format(CString("age is %d"), 20);  
AfxMessageBox(str);
```

- MessageBox函数：只在CWnd的子类中使用，功能比AfxMessageBox多

```
CString str;  
str.Format(CString("age is %d"), 20);  
MessageBox(str);  
MessageBox(str, CString("错误"), MB_YESNO | MB_ICONERROR);
```

自定义log宏，简化打印

```
#define log(fmt, ...) \  
CString str; \  
str.Format(CString(fmt), __VA_ARGS__); \  
AfxMessageBox(str);
```

```
log("age is %d", 20);
```

打开URL

```
ShellExecute(  
    NULL,  
    CString("open"),  
    CString("https://ke.qq.com/course/336509"),  
    NULL, NULL, SW_SHOWNORMAL);
```

事件注册 - 手动

```
Resource.h  [x]
MFCApplication1 (全局范围)
7  #define IDS_ABOUTBOX 101
8  #define IDD_MFCAPPLICATION1_DIALOG 102
9  #define IDR_MAINFRAME 128
10 #define IDC_COURSE 1000
```

```
MFCApplication1Dlg.cpp  [x]
MFCApplication1 (全局范围)
163 void CMFCApplication1Dlg::OnBnClickedCourse()
164 {
165     ...
166 }
```

```
MFCApplication1Dlg.h  [x]
MFCApplication1 (全局范围)
25 protected:
26     HICON m_hIcon;
27
28     // 生成的消息映射函数
29     virtual BOOL OnInitDialog();
30     afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
31     afx_msg void OnPaint();
32     afx_msg HCURSOR OnQueryDragIcon();
33     DECLARE_MESSAGE_MAP()
34 public:
35
36     afx_msg void OnBnClickedCourse();
37 };
38
```

```
MFCApplication1Dlg.cpp  [x] WinUser
MFCApplication1 (全局范围)
69 BEGIN_MESSAGE_MAP(CMFCApplication1Dlg, CDialogEx)
70     ON_WM_SYSCOMMAND()
71     ON_WM_PAINT()
72     ON_WM_QUERYDRAGICON()
73     ON_BN_CLICKED(IDC_COURSE, &CMFCApplication1Dlg::OnBnClickedCourse)
74 END_MESSAGE_MAP()
```

事件注册 – 自动 (也可以直接双击控件)



命令名:
IDC_KILL

消息类型(Y):
BN_CLICKED
BCN_DROPDOWN
BCN_HOTITEMCHANGE

类列表(L):
CAboutDlg
CMFCApplication1App
CMFCApplication1Dlg

函数处理程序名称(N):
OnBnClickedKill

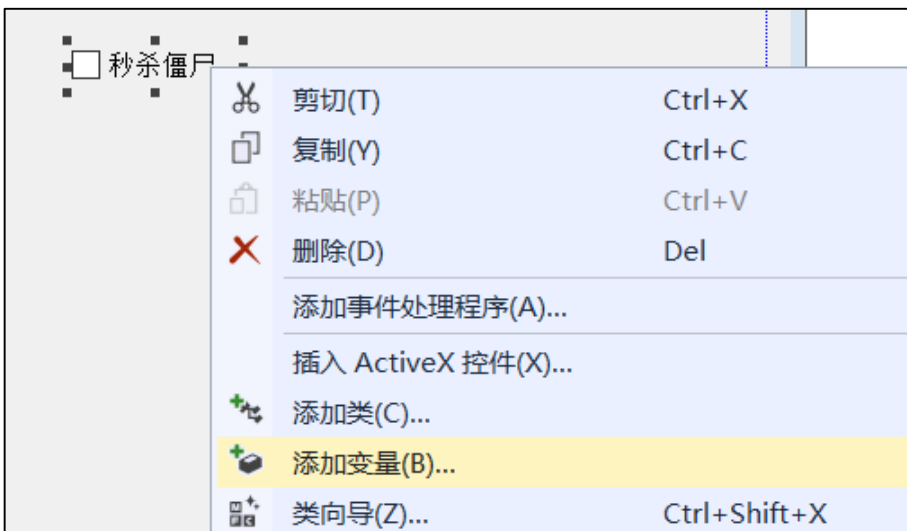
处理程序说明:
指示用户单击了按钮

绑定成员变量 - 手动

```
MFCApplication1Dlg.h  [X]
MFCApplication1
40 private:
41     CButton m_sunshine;
42     CButton m_kill;
43 };
```

```
MFCApplication1Dlg.cpp  [X]
MFCApplication1 (全局范围)
64 void CMFCApplication1Dlg::DoDataExchange(CDataExchange* pDX)
65 {
66     CDialogEx::DoDataExchange(pDX);
67     DDX_Control(pDX, IDC_SUNSHINE, m_sunshine);
68     DDX_Control(pDX, IDC_KILL, m_kill);
69 }
```

绑定成员变量 - 自动



控件	控件 ID(I):	控件类型(Y):
其他	IDC_KILL	CHECK
	类别(T):	名称(N):
	控件	m_kill
	访问(A):	变量类型(V):
	private	CButton
	注释(M):	秒杀僵尸

单选框的状态读取和修改 – 方法1

// 读取

```
BOOL state = IsDlgButtonChecked(IDC_SUNSHINE);
```

// 修改

```
CheckDlgButton(IDC_SUNSHINE, true);
```


单选框的状态读取和修改 – 方法2

```
CButton *button = (CButton *)GetDlgItem(IDC_SUNSHINE);
```

```
// 读取
```

```
BOOL state = button->GetCheck();
```

```
// 修改
```

```
button->SetCheck(true);
```

单选框的状态读取和修改 – 方法3

// 读取

```
BOOL state = this->m_kill.GetCheck();
```

// 修改

```
this->m_kill.SetCheck(true);
```

■ Windows平台软件破解必备知识

- 文件格式：PE文件
- 汇编语言：x86、x64汇编
- 工具：Ollydbg
- Windows API

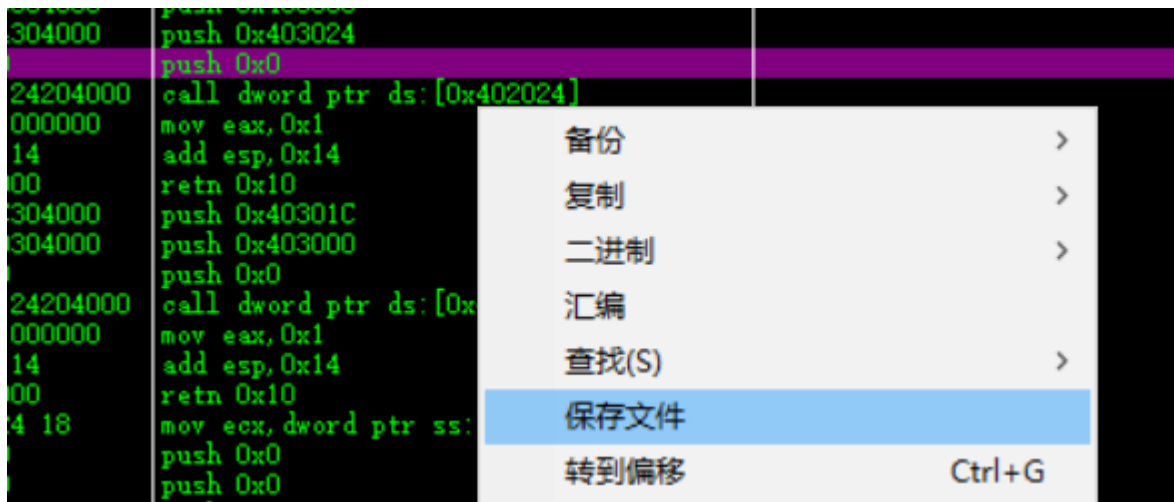
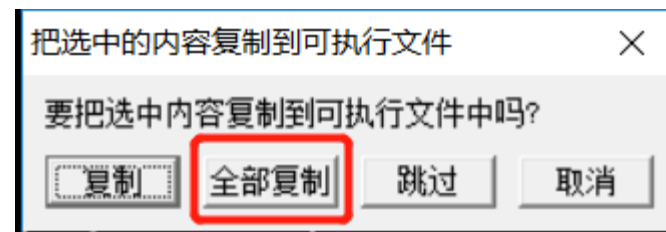
■ OD常用快捷键

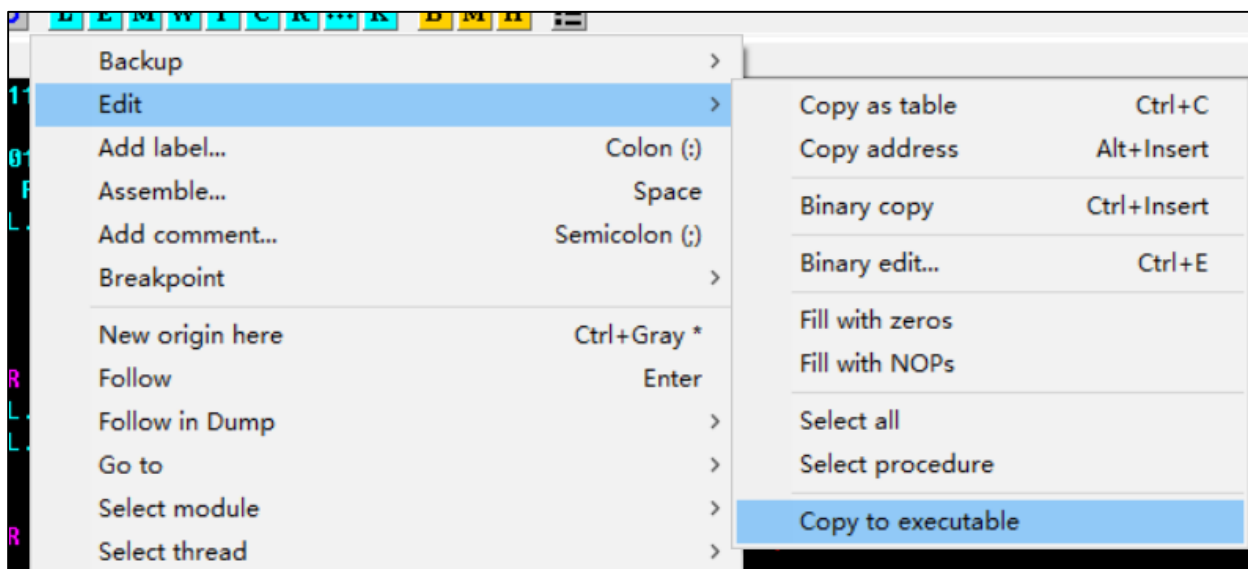
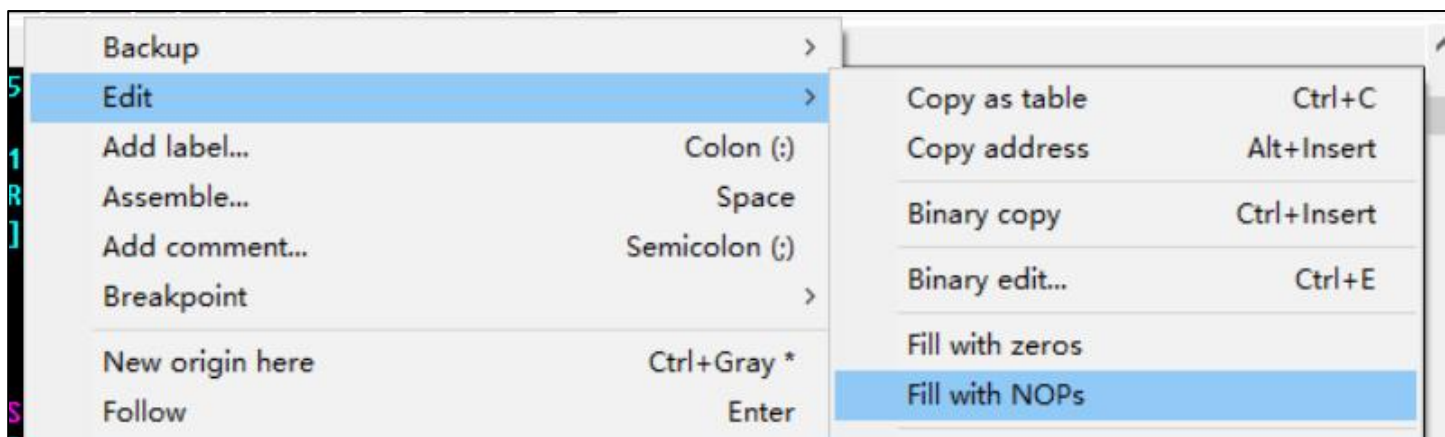
- F2：切换断点
- F9：运行程序
- F7：Step Into
- F8：Step Over
- Ctrl + G：搜索代码

用NOP填充



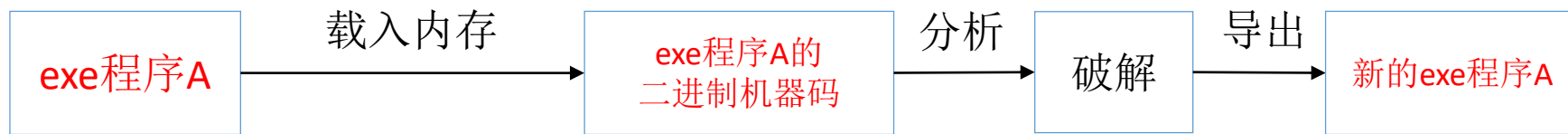
导出破解后的exe文件



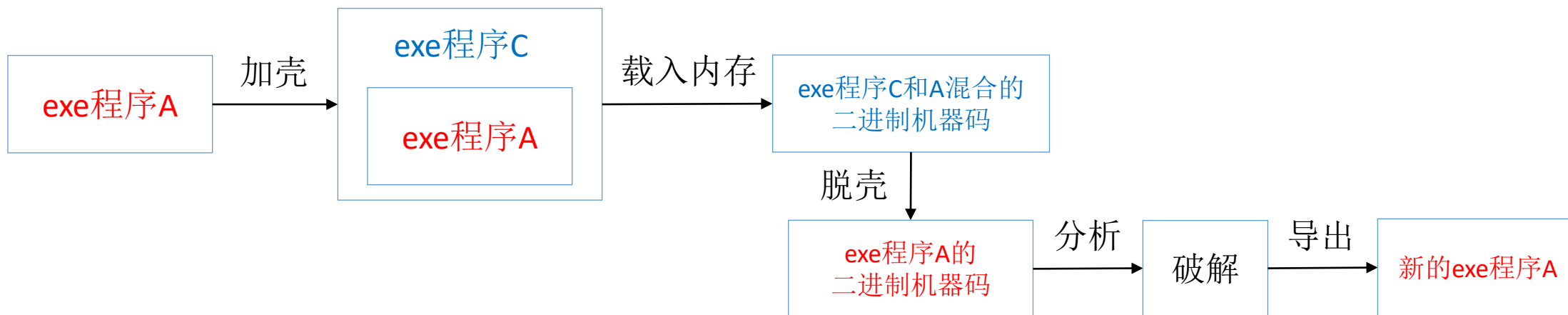


加壳与脱壳

■ 一般的软件破解思路



■ 加壳和脱壳



外挂的本质

■ 常见的外挂功能有2种做法

□ 修改内存中的数据



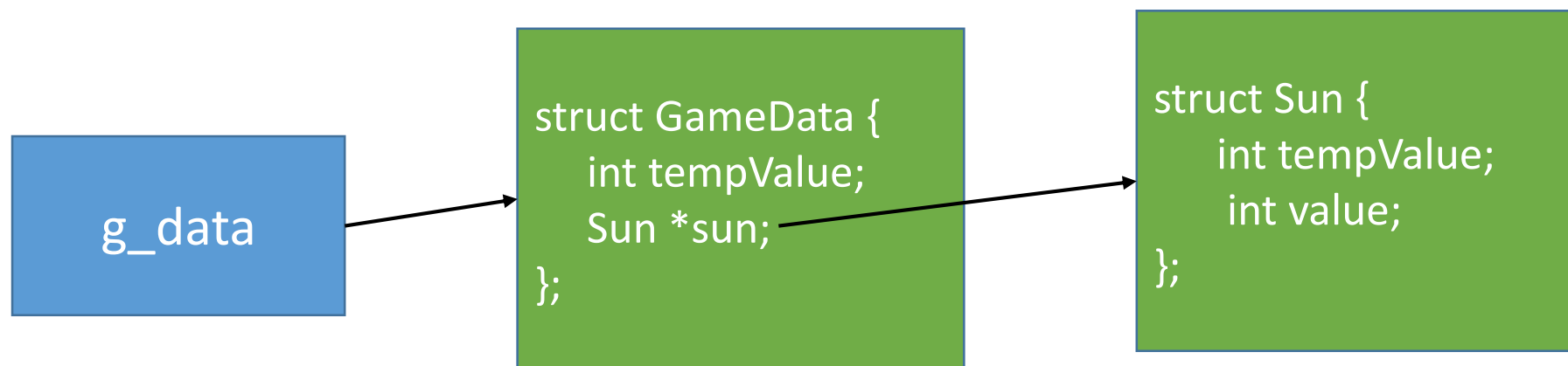
□ 修改内存中的代码

0052FCE7	. 51	PUSH ECX
0052FCE8	. 8B4D 04	MOV ECX,DWORD PTR SS:[EBP+4]
0052FCEB	. E8 20CEEDFF	CALL 0040CB10
0052FCF0	. 8346 40 FC	ADD DWORD PTR DS:[ESI+40],-4
0052FCF4	. 8B4E 40	MOV ECX,DWORD PTR DS:[ESI+40]
0052FCF7	. C786 B4000000	MOV DWORD PTR DS:[ESI+0B4],32

外挂

0052FCE7	. 51	PUSH ECX
0052FCE8	. 8B4D 04	MOV ECX,DWORD PTR SS:[EBP+4]
0052FCEB	. E8 20CEEDFF	CALL 0040CB10
0052FCF0	. 90	NOP
0052FCF1	. 90	NOP
0052FCF2	. 90	NOP
0052FCF3	. 90	NOP
0052FCF4	. 8B4E 40	MOV ECX,DWORD PTR DS:[ESI+40]
0052FCF7	. C786 B4000000	MOV DWORD PTR DS:[ESI+0B4],32

■ 其实数据和代码并没有本质区别, 在内存中都是0和1



使用Spy++查找窗口

