# SQLite数据库逆向分析

**源码分享：**[https://github.com/**zmrbak**/SQLiteReverse](https://github.com/zmrbak/SQLiteReverse)

## 最简单的Main函数

## 从软件外部调用函数

## 从软件内部调用函数

没有源代码的情况下

《x86/x64软件逆向分析入门》

## 截获函数中的数据

000A1730 - 55 - push ebp

## 软件补丁的制作

## 为现有的软件开发一个控制界面

## Visual Studio 2019安装

## SQLite数据库源码介绍

## 使用命令行编译SQLite源文件

## 使用VS编写一个简单的数据库操作程序

## 获取软件访问SQLite数据库的句柄

## 从软件中读出SQLite数据库句柄

## 从软件中找到sqlite3\_exec函数

## 验证从软件中找到sqlite3\_exec函

## 使用IDA进一步分析软件中的函数

## 解决注入含窗口的易语言DLL的问题

本节课程DEMO由群友**“落雪无尘V”**提供，在此表示感谢！



启动线程 (&启动窗口, , )

载入 (窗口1, , 假)

.判断循环首 (真)

处理事件 ()

延时 (100)

.判断循环尾 ()

## 易语言中的参数传递和命名约定分析（1）

**push 4**

**push 3**

**push 2**

**push 1**

call ?f\_\_\_@@YAHHHHH@Z ; f\_\_\_

**add esp, 16** ; 00000010H

**push 4**

**push 3**

**push 2**

**push 1**

call ?f\_\_\_cdecl@@YAHHHHH@Z ; f\_\_\_cdecl

**add esp, 16**

**push 4**

**push 3**

**push 2**

**push 1**

call ?f\_\_stdcall@@YGHHHHH@Z

**push 4**

**push 3**

**mov edx, 2**

**mov ecx, 1**

call ?f\_\_fastcall@@YIHHHHH@Z

**push 4**

**push 3**

**mov edx, 2**

**mov ecx, 1**

call ?f\_\_vectorcall@@YQHHHHH@Z

**push 4**

**push 3**

**push 2**

**push 1**

call ?f\_\_naked@@YAHHHHH@Z ; f\_\_naked

**add esp, 16**

## 易语言中的参数传递和命名约定分析（2）

**总结：在易语言中，使用**stdcall来调用函数

**.text:00401043 push 4**

**.text:00401048 push 3**

**.text:0040104D push 2**

**.text:00401052 push 1**

**.text:00401057 call sub\_401091**

**.**

**push 4**

**push 3**

**push 2**

**push 1**

call ?f\_\_stdcall@@YGHHHHH@Z

\_TEXT SEGMENT

\_a$ = 8 ; size = 4

\_b$ = 12 ; size = 4

\_c$ = 16 ; size = 4

\_d$ = 20 ; size = 4

?f\_\_stdcall@@YGHHHHH@Z PROC ; f\_\_stdcall

; File C:\Users\libit\source\repos\SQLite\_17\SQLite\_17\SQLite\_17.cpp

; Line 19

**push ebp**

**mov ebp, esp**

; Line 21

**//我们要实现的代码**

xor eax, eax

; Line 22

**pop ebp**

**ret 16** ; 00000010H

?f\_\_stdcall@@YGHHHHH@Z ENDP ; f\_\_stdcall

\_TEXT ENDS

## 寻找SQLite数据库中回调函数的样板

**//调用sqlite3\_exec**

5303E510 |. 56  **push esi //** Error msg written here

5303E51B |. 50 **push eax //** 1st argument to callback

5303E51C |. 68 00E30353 push **WeChatWi.5303E300 //** Callback function

5303E521 |. FF75 0C  **push [arg.2] //** SQL to be evaluated

5303E524 |. FF75 08 **push [arg.1] //** An open database

5303E527 |. E8 9411FFFF call **WeChatWi.5302F6C0 //** sqlite3\_exec

5303E530 |. 83C4 14  **add esp,0x14**

回调函数：

call ?f\_\_\_cdecl@@YAHHHHH@Z

sqlite3\_exec

call ?f\_\_\_cdecl@@YAHHHHH@Z

int (\*callback)(void\*, int, char\*\*, char\*\*), /\* Callback function \*/

**//在sqlite3\_exec中调用回调函数**

5302F887 |> \53 ||push ebx

5302F888 |. FF7424 1C ||push dword ptr ss:[esp+0x1C] ; msctf.75F70900

5302F88C |. 57 ||push edi

5302F88D |. FF75 14 ||push [arg.4]

5302F890 |. FF55 10  **||call [arg.3] // Callback function**

5302F893 |. 83C4 10 ||add esp,0x10

## 如何将stdcall函数调用转换为cdecl函数调用

int \_\_stdcall e\_f(int a,int b,int c,int d)

{

index++;

printf("e\_f: %d,%d,%d,%d,%d\n", index, a, b, c, d);

return 0;

}

**int \_\_cdecl c\_f(int a, int b, int c, int d)**

{

**return e\_f(a,b,c,d);**

}

## 如何使用易语言构造cdecl类型函数（1）

## 如何使用易语言构造cdecl类型函数（2）

## 如何使用易语言构造cdecl类型函数（3）

## 如何使用易语言构造cdecl类型函数（4）

## 使用易语言调用软件中的sqlite3\_exec函数（1）

## 使用易语言调用软件中的sqlite3\_exec函数（2）

## 使用C语言调用软件中的sqlite3\_exec函数

## 补充：将易语言函数“改造”成cdecl类型函数

本节课程DEMO由群友“小邪”提供，在此表示感谢！



\_TEXT SEGMENT

\_a$ = 8 ; size = 4

\_b$ = 12 ; size = 4

\_c$ = 16 ; size = 4

\_d$ = 20 ; size = 4

?**f\_\_\_cdecl**@@YAHHHHH@Z PROC ; f\_\_\_cdecl

; File C:\Users\libit\source\repos\SQLite\_17\SQLite\_17\SQLite\_17.cpp

; Line 13

push ebp

mov ebp, esp

; Line 15

xor eax, eax

; Line 16

pop ebp

ret 0

?f\_\_\_cdecl@@YAHHHHH@Z ENDP ; f\_\_\_cdecl

\_TEXT ENDS

\_TEXT SEGMENT

\_a$ = 8 ; size = 4

\_b$ = 12 ; size = 4

\_c$ = 16 ; size = 4

\_d$ = 20 ; size = 4

**?f\_\_stdcall**@@YGHHHHH@Z PROC ; f\_\_stdcall

; File C:\Users\libit\source\repos\SQLite\_17\SQLite\_17\SQLite\_17.cpp

; Line 19

push ebp

mov ebp, esp

; Line 21

xor eax, eax

**//添加代码**

**pop ebp**

**ret 0**

; Line 22

pop ebp

ret 16 ; 00000010H

?f\_\_stdcall@@YGHHHHH@Z ENDP ; f\_\_stdcall

\_TEXT ENDS

; Line 47

push 4

push 3

push 2

push 1

call ?f\_\_\_cdecl@@YAHHHHH@Z ; f\_\_\_cdecl

**add esp, 16** ; 00000010H

;

push 4

push 3

push 2

push 1

call ?f\_\_stdcall@@YGHHHHH@Z ; f\_\_stdcall

改造方法：

函数开始：

' pushad

**置入代码 ({ 96 }) ' pushad**

函数结尾：

**置入代码 ({ 97 }) ' popad**

**置入代码 ({ 184, 0, 0, 0, 0 }) ' mov eax,0 返回值**

**置入代码 ({ 137, 236 }) ' mov esp, ebp**

**置入代码 ({ 93 }) ' pop ebp**

**置入代码 ({ 195 }) ' ret 0**

## SQLite数据库在线备份

## 用函数指针调用SQLite数据库函数

## 如何在内存中构造一个函数（1）

## 如何在内存中构造一个函数（2）

## 如何判断软件中的SQLite数据库版本

Version 3.17.0

fts5: 2017-02-13 16:02:40 ada05cfa86ad7f5645450ac7a2a21c9aa6e57d2c

<https://www.sqlite.org/src/info/ada05cfa86ad7f56>

<https://www.sqlite.org/src/zip/ada05cfa/SQLite-ada05cfa.zip>

## 如何组装老版本SQLite数据库源码

## 准备从正在运行的软件中备份SQLite数据库（1）

## 准备从正在运行的软件中备份SQLite数据库（2）

## 如何在软件中定位SQLite数据库中的函数（1）

//text:107A5180

DWORD address\_sqlite3\_open = wxBaseAddress + 0x7A5180;

//.text:1074EA30

DWORD address\_sqlite3\_backup\_init = wxBaseAddress + 0x74EA30;

//text:1074ED70

DWORD address\_sqlite3\_backup\_step = wxBaseAddress + 0x74ED70;

//.text:107A57F0

DWORD address\_sqlite3\_sleep = wxBaseAddress + 0x07A57F0;

//.text:1074F4C0

DWORD address\_sqlite3\_backup\_finish = wxBaseAddress + 0x074F4C0;

//.text:107A3000

DWORD address\_sqlite3\_close = wxBaseAddress + 0x7A3000;

//text:1074F5C0

DWORD address\_sqlite3\_backup\_remaining = wxBaseAddress + 0x74F5C0;

//.text:1074F5D0

DWORD address\_sqlite3\_backup\_pagecount = wxBaseAddress + 0x74F5D0;

//text:107A4250

DWORD address\_sqlite3\_errcode = wxBaseAddress + 0x7A4250;