# 网络安全入门实验01: 逆向分析三个文件



6.实验01总结

# 0.前言

大学三年终于上到了最最最想学的课。

经历过买了很多相关书籍、看了很多相关视频,但是发现其中大部分讲的太过浅显,感觉并没有什么实际作 用、不知道从哪里入手、讲的太难又看不懂的迷惘

曾经觉得"入门太简单,但要做的很好太难"

曾经怀疑过,为什么别人都可以自学,为什么自己这么差劲

在那一个个失落的夜晚之后,今天终于真的真的真的要入门了。

发现了原来在推开实现梦想的大门,踏上第一级阶梯之前,一切的失意都不值得一提。

甚至兴奋地颤抖。

所以在这里,将课堂的实验,以实验报告的形式发表博客,记录下自己学习的过程,也希望可以跟大家分享这 个过程

如有不足、如果错误,还希望大家指正

1.实验环境搭建

本周的第一个实验是逆向分析三个简单的文件,不需要搭配什么特别的环境 需要的软件: IDA PRO



p.s PRO专业版和普通的版本打开文件后的默认窗口是略有差别的,提供的功能也是不一样的

## 2.IDA PRO简单介绍



打开IDA PRO会显示快速启动的界面

其中有三个选项: "New"、"Go"、"Previous"

然后会默认打开一个"打开文件的窗口"

Select file to disassen	nble						l	x
	、▶ 本地	b磁盘 (E:) ▶ ida61 ▶			<b>▼</b> 4 <sub>7</sub>	搜索 ida61		م
组织 ▼ 新建文件夹	E						•	0
🚺 下載	1 名利	<u>م</u>	修改日期	类型	大小			<b>^</b>
📃 桌面		_RDW-ANTIPIRATE	2017\4\9 星期日	文件夹				
📃 最近访问的位置		cfg	2017\4\9 星期日	文件夹				=
		examples	2017\4\9 星期日	文件夹				-
■ 桌面		idc	2017\4\9 星期日	文件夹				
🛆 WPS网盘	Ξ 🔒	ids	2017\4\9 星期日	文件夹				
库		loaders	2017\4\9 星期日	文件夹				
🛃 视频		plugins	2017\4\9 星期日	文件夹				
■ 图片		procs	2017\4\9 星期日	文件夹				
		python	2017\4\9 星期日	文件夹				
→ 讯雷下载		python64	2017\4\9 星期日	文件夹				
		sig	2017\4\9 星期日	文件夹				
		til	2017\4\9 星期日	文件夹				
	Qt	assistant	2011\4\8 星期五	应用程序	1,225 KB			
	4	clp.dll	2011\4\8 星期五	应用程序扩展	1,061 KB			
■ 网络	+	dbghelp.dll	2011\4\8 星期五	应用程序扩展	1,185 KB			-
Ż	件名( <u>N</u> ):				-	All known file ex	tensions (*.	.3 🔻
						打开(O)	取消	

我们可以从这里选择要解析的文件,也可以直接拖动文件到IDA PRO里面进行解析

Drag a file here to disassemble it

在加载文件时有多个选项

Load a new file Load file C:\Users\Administrator\Desktop Portable executable for 80386 (PE) [pe6 MS-DOS executable (EXE) [dos64.164] Binary file	▶ ▶\实验样本(解压密码: 123)\Sample_1.exe as i4, 64]
Processor type	
Intel 80x86 processors: metapc Loading segment 0x00000000000000000000000000000000000	<ul> <li>✓ Set</li> <li>Analysis</li> <li>✓ Enabled</li> <li>✓ Indicator enabled</li> </ul>
<ul> <li>Create segments</li> <li>Load resources</li> </ul>	Kernel options <u>1</u>
✓ Rename DLL entries Manual load ✓ Fill segment gaps	Kernel options 2
✓ Make imports segment ○ Create FLAT group	Processor options
DLL directory C:\windows	ancel

主要是在窗口最上方的选项

"Load file xxx as"(程序的类型)

这里我们选择默认的第一个,其他选项也按照默认的参数来,点击OK即可

我们选择的文件是Sample\_1的可执行文件

OK后,开始Disassemble我们选择的文件,出现如下窗口

🖆 🖬 (+ + + + 🍓 🐴 🐜 斗 🔬 🗖 🕥	at at at 🖈 🖈 🖬 🕻	🗙 🕨 🔲 🖬 No debugger	• 🗊 🕈 🕅						
								3	-
📝 Functions window 🗗 🗙	IDA Vies-A	🛛 🖸 Hex View-A	Structures		Enuns 💽 🕇	Inports	Exports	×	
Unction name  Unction name  Unction name  Unction name U	180.004 (-189,-43) (2	<pre>Image: This file has h</pre>	een generated by The Interact ht (c) 2011 Hex-Rays, <support License info: A8-227-227 ESET Spol. S r.o. BSE2ASO953CB06CF18923F8EF50 907 erskdministrator\Desktop\33 ble executable for 80386 (PE) address 000f1000) address 000f1000) effor section: 0000000 (210 of essetton: 0000000 (210 of essetton: 0000000 (210 of essetton: 0000000 (250) for section: 0000000 (250) t facotable Readable ault Svindos Executable 32bit unicode subdir of ida for ini code : Read/Execute blic 'CODE' use22 00: WilnMain(x,x,x)</support 	ive Disassembler (IG GNex-rays.com) -B7 A祥本 (解压密码, 120 B.) B.) Fo on unicode	n) \Sample_1.exe				
Output window riushing purfers, please waltox									e ×
File 'Ci\Users\AdministratoryDesicop\Self## (銀田田母) Compiling file 'Eilde611de61. Descuting function 'main' Executing function 'Wolled' Executing function 'Wolled' Users and the self of the self of the self of the self on any start to explore the input file right now. Losdilbrary[Eilde61]pluginstyrhon.p64) error: 於不歸 Eilde61]pluginstyrthon.p64: cas't losd file Duing TLRT sendertic: Normation has been from April File In initial automalyzin has been firinghed.	123) \Sample_1.exe' h 指定的模块。 time	ss been successfully loaded int	to the database.						
IBC AU: idle Down Disk: 44GB									

"Function Name"窗口,即显示可执行文件所调用的函数

"IDA View-A"窗口,即可执行文件的反汇编窗口

"Output Window",即可执行文件的输出

这里我们主要关注的是反汇编窗口,

其余的有关以十六进制显示的窗口还没有学到,在此先不做研究

在反汇编窗口点击鼠标右键,可以看到有多个选项,我们可以通过点击"Text View"来更改视图(默认以Graph View显示)

	Group nodes	
<b>8</b> 24	Xrefs from	
:	Enter <u>c</u> omment	:
()	Enter <u>r</u> epeatable comment	;
f	Edit function	Alt+P
fo	Set function type	Y
-	<u>H</u> ide	-
	Text view	
×	<u>U</u> ndefine	U
	Synchronize with	+

其余相关介绍可以参考其他的参考资料,本文仅粗略介绍本次实验需要用到的部分

# 3.Sample\_01的逆向分析

\_stdcall wWinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPWSTR lpCmdLine, int nShowCmd) ; int wWinMain@16 proc near hInstance= dword ptr 8 hPrevInstance= dword ptr ACh lpString1= dword ptr 10h nShowCmd= dword ptr 14h push ebp mov ebp, esp eax, [ebp+lpString1] mov offset String2 ; "2012" push push eax ; 1pString1 call ds:1strcmpW ; uType push **O** offset Caption ; "MESSAGE" push test eax, eax short loc 401035 jnz 🛄 🖂 🖂 🖬 🖂 🖂 ; "Hello! 2012" push offset Text loc 401035: ; "Hello! Windows' ds:GetActiveWindow call ; hWnd push eax push offset aHelloWindows call ds:MessageBoxW call ds:GetActiveWindow xor eax, eax push eax ; hWnd

; int \_\_stdcall <mark>wWinMain</mark>(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPWSTR lpCmdLine, int nShowCmd)

call

XOP

рор

retn

ds:MessageBoxW

eax, eax

ebp 10h

\_wWinMain@16 endp

wWinMain@16 proc near

рор

retn

ebp

1 Øb

```
hInstance= dword ptr 8
hPrevInstance= dword ptr
                          OCh
lpString1= dword ptr 10h
nShowCmd= dword ptr 14h
push
        ebp
MOV
        ebp, esp
mov
        eax, [ebp+lpString1]
        offset String2 ; "2012"
push
push
        eax
                        ; 1pString1
call
        ds:1strcmpW
                        ; uType
push
        ß
        offset Caption
                        ; "MESSAGE"
push
test
        eax, eax
        short loc_401035
jnz
```

https://blog.csdn.net/Krumitz

https://blog.csdn.na

打开Sample\_01.exe后,跳过上面的文件信息的部分,我们就找到了这个可执行文件的Main函数的入口(黄色选中部分)

那么黑框里面的,就肯定是主函数的部分啦

前三行我们可以先不详细分析,因为本次实验的目的主要是通过逆向分析猜测可执行文件会执行一个怎么样的 操作,并且解释相关函数的作用

那么我们从第四行开始看

push offset String2 ; "2012" push eax ; 1pString1

我们可以看到,函数将"2012",和lpString1 push入栈

接着就来到了

call ds:lstrcmpW

call表示调用了一个函数,这个函数的名称为"lstrcmpW"

通过参考微软的函数手册,我们可以看到

# Syntax

C++	心复制
<pre>int lstrcmpW(    LPCWSTR lpString1,    LPCWSTR lpString2 );</pre>	

## Parameters

lpString1

Type: LPCTSTR

The first null-terminated string to be compared.

lpString2

Type: LPCTSTR

The second null-terminated string to be compared.

# **Return Value**

Type: int

test

If the string pointed to by *lpString1* is less than the string pointed to by *lpString2*, the return value is negative. If the string pointed to by *lpString1* is greater than the string pointed to by *lpString2*, the return value is positive. If the strings are equal, the return value is zero.

lstrcmpw函数比较了两个字符串lpString1和lpString2的值

如果lpString1小于lpString2,则函数返回一个负数

如果lpString大于lpString2,则函数返回一个正数

如果两个字符串相等,返回值为0

eax. eax

所以在这里我们就可以进行猜测了,这个程序首先将我们从命令行输入的值跟"2012"进行了比较,比较之 后做了什么呢?让我们继续往下看

	*	-	
🔛 🖂 🖻	3	u 🖬 🖬 🗉	
push	offset Text ; "Hello! 2012"		
call	ds:GetActiveWindow	loc_401035: ; "Hello! Wi	ndows"
push	eax ; hWnd	push offset aHelloWindows	
call	ds:MessageBoxW	call ds:GetActiveWindow	
xor	eax, eax	push eax ; hWnd	
рор	ebp	call ds:MessageBoxW	
retn	10h	xor eax, eax	
		- pop ebp	
		reto 18b	

接下来test了两个值,通过查阅资料,我们知道了在反汇编中

test和jnz在这里是连用的

首先 test eax eax的含义是

if(EAX == 0) ZF = 1 else ZF = 0

而JNZ short loc\_401035是

if (ZF == 0) GOTO loc\_401035

那么就是,首先test了一个值,看这个值是否为0

如果为0,则ZF = 1,如果不为0,ZF = 0

然后

如果ZF == 0

就跳转到loc\_401035

那loc 401035是啥

还好当你选中这个值的时候,软件会自动帮你选中这个窗口中所有相同的值

那我们就可以猜测

比较了lstrcmpW函数的值是否为0

如果函数返回0,则ZF为1,则不会跳转

如果函数返回不为0,则ZF为0,则会跳转到loc\_401035,执行他接下来的代码



接下来就很简单了,我们看到两边的代码(跳转 or 没有跳转)

分别有"Hello! 2012"和"Hello! Windows"

然后都call了"GetActiveWindow"和"MessageBoxW"函数

那么可以猜测

如果没有跳转,则程序获取一个窗口句柄,并show了一个MessageBox,显示Hello! 2012

### 如果跳转了,则程序获取一个窗口句柄,并show了一个MessageBox,显示Hello! Windows

最后程序return结束

我们可以来测试一下

打开命令行



### 如果不输入任何值,很明显, lstrcmpW返回肯定不为0,显示Hello! Windows



#### 如果输入不为2012,同理可得,也是现实Hello! Windows



#### 输入为2012呢?那当然会显示Hello!2012了

### Sample\_01总结(来自图灵社区相关文章)

#### http://www.ituring.com.cn/book/tupubarticle/9632

"和无参数的情况相比,这次显示出来的消息变成了 Hello! 2012。

可能有人要问:"那又如何?"我们发现了通过传递 2012 这个参数,程序的显示结果会发生变化,这很重要。因为我们在"完全没有源代码的情况下,搞清楚了程序的行为"。

#### 这就是逆向工程。

刚才这个参数是我们猜测出来的,其实只要阅读汇编语言代码,就可以发现其中使用了 lstrcmpW 对字符串 2012 和命令行参数进行了比较操作。"

"我们没必要看懂全部的汇编语言代码。和刚才使用二进制编辑器的时候一样,只要一眼望去能大概理解这段代 码做了什么事就可以了。"

"刚一听到"逆向工程""汇编"这些词的时候,大家总会以为它们很难,但实际上并非如此。使用 IDA,我们就可以 将可执行文件转换成像 C 语言一样容易理解(实际上还是有差距的)的汇编代码。尤其是它的 Graph view 十分 强大,可以让我们十分清晰地看出程序的分支逻辑。

只要一定程度上掌握这些工具的使用方法,大家就可以完成很多软件分析工作了。"

这三句话,也恰恰好的解释了上文所说的"本次实验的目的主要是通过逆向分析猜测可执行文件会执行一个怎么 样的操作,并且解释相关函数的作用"

## 4.Sample\_02的逆向分析

实验要求: Sample\_2分析范围: 0x00401000至0x0040105E

所以我们同样把Sample 02的可执行文件拖入IDA PRO,并右键切换到Text View视图

方便我们按范围分析

.text:00401000 ; int	stdcall <mark>WinMain</mark> (HINSTANCE	hInstance, HINSTANCE	hPrevInstance, LPSTR ]	<pre>ipCmdLine, int nShowCmd)</pre>
.text:00401000 _WinMa	in@16 proc near	; CODE XREF: _	tmainCRTStartup+14B_p	Ĵ
.text:00401000				
.text:00401000 Msg	= tagMSG ptr -820	ðh		
.text:00401000 String	2 = byte ptr -804h			
.text:00401000 String	1 = byte ptr -404h			
.text:00401000 var_4	= dword ptr -4			
.text:00401000 hInsta	nce = dword ptr 8			
.text:00401000 hPrevI	nstance = dword ptr OCh			
.text:00401000 lpCmdL	ine = dword ptr 10h			
.text:00401000 nCmdSh	ow = dword ptr 14h			
.text:00401000				
.text:00401000	push ebp			
.text:00401001	mov ebp, esp			
.text:00401003	sub esp, 820h	1		
.text:00401009	mov eax,s	security_cookie		
.text:0040100E	xor eax, ebp			
.text:00401010	mov [ebp+var_	_4], eax		
.text:00401013	push esi			
.text:00401014	mov esi,[ebp	)+hInstance]		
.text:00401017	push 400h	; nSize		
.text:0040101C	lea eax,[ebp	)+String2]		
.text:00401022	push eax	; lpFilename		
.text:00401023	push 0	; hModule		
.text:00401025	call ds:GetMod	luleFileNameA		
.text:0040102B	push 0	; fCreate		
.text:0040102D	push 7	; csidl		
.text:0040102F	lea ecx, [ebp	o+String1]		
.text:00401035	push ecx	; pszPath		
.text:00401036	push 0	; hwnd		
.text:00401038	call ds:SHGetS	specialFolderPathA		
.text:0040103E	push offset St	ring2 ; "\\0.exe"		
.text:00401043	lea edx, [ebp	)+String1]		

首先我们还是找到文件的主函数入口

从0x00401000至0x0040105E的范围内,一共call了四个函数

.text:00401025	<mark>call</mark>	ds:GetModuleFile	NameA
.text:0040102B	push	0	; fCreate
.text:0040102D	push	7	; csidl
.text:0040102F	lea	ecx, [ebp+String	1]
.text:00401035	push	ecx	; pszPath
.text:00401036	push	0	; hwnd
.text:00401038	call	ds:SHGetSpecialF	olderPathA
.text:0040103E	push	offset String2	; "\\0.exe"
.text:00401043	lea	edx, [ebp+String	1]
.text:00401049	push	edx	; lpString1
.text:0040104A	call	ds:1strcatA	
.text:00401050	lea	eax, [ebp+String]	2]
.text:00401056	push	eax	; lpString2
.text:00401057	lea	ecx, [ebp+String	1]
.text:0040105D	push	ecx	; lpString1
.text:0040105E	call	ds:1strcmpA	nevnumnz

分别查阅函数的作用

```
函数分析:
```

GetModuleFileNameA函数:

# Syntax

C++	Сору
DWORD GetModuleFileNameA( HMODULE hModule, LPSTR lpFilename, DWORD nSize ); https://blog.csdn.ne	

# Parameters

hModule

A handle to the loaded module whose path is being requested. If this parameter is NULL, GetModuleFileName retrieves the path of the executable file of the current process.

The GetModuleFileName function does not retrieve the path for modules that were loaded using the LOAD\_LIBRARY\_AS\_DATAFILE flag. For more information, see LoadLibraryEx.

lpFilename

A pointer to a buffer that receives the fully qualified path of the module. If the length of the path is less than the size that the *nSize* parameter specifies, the function succeeds and the path is returned as a null-terminated string.

If the length of the path exceeds the size that the *nSize* parameter specifies, the function succeeds and the string is truncated to *nSize* characters including the terminating null character.

Windows XP: The string is truncated to nSize characters and is not null-terminated.

The string returned will use the same format that was specified when the module was loaded. Therefore, the path can be a long or short file name, and can use the prefix "\?". For more information, see <u>Naming a File</u>.

nSize

The size of the lpFilename buffer, in TCHARs.

https://blog.csdn.net/Krumitz

# **Return Value**

If the function succeeds, the return value is the length of the string that is copied to the buffer, in characters, not including the terminating null character. If the buffer is too small to hold the module name, the string is truncated to *nSize* characters including the terminating null character, the function returns *nSize*, and the function sets the last error to ERROR\_INSUFFICIENT\_BUFFER.

Windows XP: If the buffer is too small to hold the module name, the function returns *nSize*. The last error code remains ERROR\_SUCCESS. If *nSize* is zero, the return value is zero and the last error code is ERROR\_SUCCESS.

If the function fails, the return value is 0 (zero). To get extended error information, call GetLastError.

hModule参数为NULL时,函数获取当前进程的可执行文件的路径,复制到lpFilename指向的缓冲区中,如果 lpFilename指向的缓冲区足够容纳该路径,则函数成功,path中存储了一个路径相关的字符串

### SHGetSpecialFolderPathA函数:

### Syntax

C++	🗅 Сору
BOOL SHGetSpecialFolderPathA( HWND hwnd, LPSTR pszPath, int csidl, BOOL fCreate );	

## Parameters

hwnd

#### Type: HWND

Reserved.

pszPath

#### Type: LPTSTR

A pointer to a null-terminated string that receives the drive and path of the specified folder. This buffer must be at least MAX\_PATH characters in size.

csidl

Type: int

A CSIDL that identifies the folder of interest. If a virtual folder is specified, this function will fail.

#### fCreate

Type: BOOL

Indicates whether the folder should be created if it does not already exist. If this value is nonzero, the folder is created. If this value is zero, the folder is not created.

## **Return Value**

Type: BOOL

TRUE if successful; otherwise, FALSE.

### 获取一个指定的系统路径

### IstrcatA函数:

## Syntax

```
C++ Copy
```

## Parameters

lpString1

Type: LPTSTR

The first null-terminated string. This buffer must be large enough to contain both strings.

lpString2

#### Type: LPTSTR

The null-terminated string to be appended to the string specified in the lpString1 parameter.

# Return Value 🗠

#### Type: LPTSTR

If the function succeeds, the return value is a pointer to the buffer.

If the function fails, the return value is NULL and *lpString1* may not be null-terminated.

https://blog.csdn.net/Krumitz

返回了一个指向长度足以存储lpString1和lpString2的缓冲区的指针

可以理解为返回了一个LPTSTR形的字符串

此字符串为lpString1和lpString2的连接字符串

IstrcmpA函数:

# Syntax

C++

```
int lstrcmpA(
   LPCSTR lpString1,
   LPCSTR lpString2
);
```

## Parameters

lpString1

Type: LPCTSTR

The first null-terminated string to be compared.

lpString2

Type: LPCTSTR

The second null-terminated string to be compared.

# **Return Value**

Type: int

If the string pointed to by *lpString1* is less than the string pointed to by *lpString2*, the return value is negative. If the string pointed to by *lpString1* is greater than the string pointed to by *lpString2*, the return value is positive. If the strings are equal, the return value is zero.

https://blog.csdn.net/Krumitz

如果lpString1小于lpString2,返回负数

如果大于,返回正数

如果相等,返回0

### 分析与验证

看到这里一度觉得十分的尴尬,从0x00401000至0x0040105E的范围,无非就是做了

获取文件名,获取一个路径,将路径和

.text:0040103E push offset String2 ; "\\0.exe"

"\\0.exe"拼接,然后做了一个比较

可以强行猜测一波,向这个路径复制了一个名为0的可执行文件

就没了啊,那我怎么继续逆向分析呢。。。

没办法,只好硬着头皮继续往下做

# Sample\_02拓展

发现做到这里再继续"正叙"写下去是根本一点也写不下去了

🗅 Сору

emmm可能这就是为什么只让我们分析到这里吧

因为后面的那些函数真是看不懂啊!

但是不能这么尴尬

所以我们偷偷懒,"倒叙"来看看程序是怎么走的

我们用Process Monitor来看看

 23:4...
 Sample\_2.exe
 1292
 CreateFile
 C:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\startup\0.exe

 23:4...
 Sample\_2.exe
 1292
 QueryBasicInforma...
 C:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\startup\0.exe

 23:4...
 Sample\_2.exe
 1292
 CloseFile
 C:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\startup\0.exe

看到Sample\_02向

C:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\startup

这个路径下Create了一个0.exe的File

而根据我们之前的分析和函数解释,大概可以猜到,这个0.exe跟Sample\_02.exe是一样的

我们用Stirling来验证一下

🚽 Stirling - 0.exe		A + (3.4)	
ファイル(F) 編集(E) 検索・移動(S) 設定(O) ウィンドウ(W) ヘルプ(H)			
	<u>****</u> ₀↓ °↑		
Sample_2.exe	- • X • 0.e	xe	
ADDRESS 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	0123456789ABCDEF	RESS 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E	OF 0123456789ABCDEF
00000000 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZ	30000 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00	00 MZ
00000010 B8 00 00 00 00 00 00 40 00 00 00 00 00 00	?@	10010 B8 00 00 00 00 00 00 00 40 00 00 00 00 00	00   ? @
<u> </u>			
00000030 00 00 00 00 00 00 00 00 00 00 0			00
99999949 9E 1F BA 9E 99 B4 99 CD 21 B8 91 4C CD 21 54 68	? .? ? ? L? Th	00040 DE 1F BH DE DO B4 D9 GD 21 B8 D1 4G GD 21 54	
00000050 09 73 20 70 72 0F 07 72 01 0U 20 03 01 0E 0E 0F	15 program canno 6666	30050 09 73 20 70 72 0F 07 72 0F 00 20 03 0F 0E 0E	20 t bo wup in DOS
00000000 74 20 02 05 20 72 75 0E 20 09 0E 20 44 4F 53 20		38878 AD AE AL AE 2E AD AD AD 24 21 07 0E 28 44 41 53	an modo
99999979 OV OF 04 05 2E 0V 0V 0H 24 00 00 00 00 00 00 00	o 印印=空T=空T=0 0000	30080 10 CC 3C FO 54 AD 52 RA 54 AD 52 RA 54 AD 52	80 2 明現等現等現 2
888888888 30 LD CE DO EA ON E2 DO 20 ND CC DO E2 DO	·· 门榜員榜員榜: 6666	38898 38 DB CE BA 56 AD 52 BA 38 DB CC BA 55 AD 52	BA ·希留福? 人名福?
88888898 38 08 62 08 30 80 52 08 38 08 66 08 33 80 52 08 888888888 38 38 08 62 08 57 60 52 08 30 08 66 08 53 80 52 08		B F8 BA 47 AD 52 BA 38 DB F9 BA 56 AD 52	BA · 将磁温? 停留温?
99999988 5D D5 C1 B4 59 4D 52 B4 54 4D 53 B4 1E 4D 52 B4	1. 格研現等種?	5 C1 BA 59 AD 52 BA 54 AD 53 BA 1E AD 52	BA 1 積篩環管理? 環?
9999999C9 3B DB ED BA 56 AD 52 BA 3B DB C8 BA 55 AD 52 BA	·姆管福?3 窓	B FD BA 56 AD 52 BA 3B DB C8 BA 55 AD 52	BA : '理管' ; 3 琴' ; 3 琴' ; 3 ; 3 ; 3 ; 3 ; 3 ; 3 ; 3 ; 3 ; 3 ;
99999909 3B DB CE BA 55 AD 52 BA 52 69 63 68 54 AD 52 BA	······································	B CF BA 55 AD 52 BA 52 69 63 68 54 AD 52	BA : 巯篓璕篟ichT璕?
888888E8 88 88 88 88 88 88 88 88 88 88 8	, 」, 「「「」」, 「」」, 「」, 「」, 「」, 「」, 「」, 「」,	ません 0 00 00 00 00 00 00 00 00 00 00 00 00	00
888888F8 88 88 88 88 88 88 88 88 88 88 58 45 88 88 4C 81 85 88	PEL	0 00 00 00 00 00 00 50 45 00 00 4C 01 05	00PEL
00000100 A3 7C CB 4F 00 00 00 00 00 00 00 00 E0 00 02 01	蕃	C CB 4F 00 00 00 00 00 00 00 00 E0 00 02	81 薕
00000110 0B 01 0A 00 00 0E 00 00 00 D4 00 00 00 00 00 00		1 0A 00 00 0E 00 00 04 00 00 00 00 00	00?
00000120 69 18 00 00 00 10 00 00 00 20 00 00 00 00 40 00	i	8 00 00 00 10 00 00 20 00 00 00 40	00 i
00000130 00 10 00 00 02 00 00 05 00 01 00 00 00 00 00		第定 0 00 00 00 02 00 00 05 00 01 00 00 00 00	00
00000140 05 00 01 00 00 00 00 00 00 20 01 00 00 04 00 00		0 01 00 00 00 00 00 00 20 01 00 00 4 00	00
88888158 49 92 81 88 82 88 48 81 88 88 10 88 10 88 10 88 88	I@	ער שכושנ 47 אין 2 01 00 02 00 40 81 00 00 10 00 00 10 00	00 I@
<u>86666169</u> 00 00 10 00 00 10 00 00 00 00 00 00 10 00 0		38169 00 00 10 00 00 10 00 00 00 00 00 00 10 00 0	00
86966176 80 80 80 80 80 80 80 80 80 80 80 80 80	D#	<b>39178 60 60 60 60 60 60 60 60 44 23 60 60 A0 60 60</b>	60D#

发现确实如此

嗯。。。虽然没搞懂IDA里面的东西,但是通过借助其他工具的帮助,我们还是大致知道了,这个Sample\_02会执行一个怎么样的操作

## 5.Sample\_03逆向分析

实验要求: Sample\_3分析范围: 0x00401000至0x00401062

_		
.text:00401000		
.text:00401000	push	ebp
.text:00401001	mov	ebp, esp
.text:00401003	MOV	eax, 2004h
.text:00401008	call	alloca_probe
text:0040100D	MOV	eax,security_cookie
.text:00401012	xor	eax, ebp
.text:00401014	mov	[ebp+var_4], eax
.text:00401017	push	1000h ; nSize
.text:0040101C	lea	eax, [ebp+ExistingFileName]
.text:00401022	push	eax ; 1pFilename
.text:00401023	push	0 ; hModule
.text:00401025	call	ds:GetModuleFileNameW
.text:0040102B	lea	ecx, [ebp+NewFileName]
.text:00401031	push	ecx ; pszPath
text:00401032	push	0 ; dwFlags
text:00401034	push	0; hToken
.text:00401036	push	7 ; csidl
text:00401038	push	0; hwnd
text:0040103A	call	ds:SHGetFolderPathW
text:00401040	push	offset String2 ; "\\wsample01b.exe"
.text:00401045	lea	edx, [ebp+NewFileName]
text:00401048	push	edx ; 1pString1
text:0040104C	call	ds:lstrcatW
.text:00401052	push	0 ; bFaillfExists
text:00401054	Tea	eax, [ebp+NewFileName]
.text:0040105A	push	eax ; ipnewFileName
. Text:00401058	Tea	ecx, [ebp+ExistingFileName]
, Text: 00401001	pusn	ecx ; ipExistingFileName
, Text: 00401002	Call	as:copyFilew
. Cext:00401008	MOV	ecx, [ebp+var_4]
LEXL:0040100B	xur	ecx, eup
LEXL:0040100D	xur	edx, edx
.LEAL.0040100F	LGII	e_securicy_cneck_coukie@4 ; _securicy_cneck_coukie(x)
. LEAL.004010/4 tout.00101076	NOV	esp, eup
.LEXL.004010/0	pop	https://blog.csdn.net/Krumitz
LEXL.004010//	reui	

咦让我们分析的部分没有找到函数入口啊

没事,我们往下拉一点可以看到

```
, _____
.text:00401062
                               call
                                        ds:CopyFileW
.text:00401068
                                        ecx, [ebp+var 4]
                               mov
.text:0040106B
                               xor
                                        ecx, ebp
.text:0040106D
                               xor
                                        eax, eax
.text:0040106F
                                        @_security_check_cookie@4 ; _security_check_cookie(x)
                               call
.text:00401074
                               mov
                                        esp, ebp
.text:00401076
                               рор
                                        ebp
.text:00401077
                               retn
.text:00401077 sub_401000
                               endp
.text:00401077
.text:00401077 ;
.text:00401078
                               align 10h
.text:00401080
.text:00401080 ;
                     ----- S U B R O U T I N E -----
.text:00401080
.text:00401080
.text:00401080 ; int stdc
.text:00401080 _wWinMain@16
                       stdcall wWinMaintHINSTANCE hInstance, HINSTANCE hPrevInstance, LPWSTR lpCmdLine, int nShowCmd)
                                                        ; CODE XREF: ____tmainCRTStartup+153_p
                               proc near
.text:00401080
.text:00401080 hInstance
                                = dword ptr
                                             4
.text:00401080 hPrevInstance
                               = dword ptr
                                             8
.text:00401080 lpCmdLine
                               = dword ptr
                                             OCh
.text:00401080 nShowCmd
                               = dword ptr
                                             10h
.text:00401080
.text:00401080
                               call
                                        <mark>sub_401000</mark>
.text:00401085
                               push
                                        0
                                                        ; uType
.text:00401087
                                        offset Caption
                                                        ; "MESSAGE"
                               push
.text:0040108C
                               push
                                        offset Text
                                                          "Copied!"
.text:00401091
                               call
                                        ds:GetActiveWindow
.text:00401097
                                                        ; hWnd
                               push
                                        eax
.text:00401098
                               call
                                        ds:MessageBoxW
.text:0040109E
                               xor
                                        eax, eax
.text:004010A0
                               retn
                                        10h
.text:004010A0 _wWinMain@16
                               endp
```

### 函数分析:

### GetModuleNameW函数:

### Syntax

C++	Сору
DWORD GetModuleFileNameW( HMODULE hModule, LPWSTR lpFilename, DWORD nSize );	

## Parameters

#### hModule

A handle to the loaded module whose path is being requested. If this parameter is NULL, GetModuleFileName retrieves the path of the executable file of the current process.

The GetModuleFileName function does not retrieve the path for modules that were loaded using the LOAD\_LIBRARY\_AS\_DATAFILE flag. For more information, see <u>LoadLibraryEx</u>.

#### lpFilename

A pointer to a buffer that receives the fully qualified path of the module. If the length of the path is less than the size that the *nSize* parameter specifies, the function succeeds and the path is returned as a null-terminated string.

If the length of the path exceeds the size that the *nSize* parameter specifies, the function succeeds and the string is truncated to *nSize* characters including the terminating null character.

Windows XP: The string is truncated to nSize characters and is not null-terminated.

The string returned will use the same format that was specified when the module was loaded. Therefore, the path can be a long or short file name, and can use the prefix "\?". For more information, see <u>Naming a File</u>.

nSize

The size of the lpFilename buffer, in TCHARs.

https://blog.csdn.net/Krumitz

## **Return Value**

If the function succeeds, the return value is the length of the string that is copied to the buffer, in characters, not including the terminating null character. If the buffer is too small to hold the module name, the string is truncated to *nSize* characters including the terminating null character, the function returns *nSize*, and the function sets the last error to ERROR\_INSUFFICIENT\_BUFFER.

Windows XP: If the buffer is too small to hold the module name, the function returns *nSize*. The last error code remains ERROR\_SUCCESS. If *nSize* is zero, the return value is zero and the last error code is ERROR\_SUCCESS.

If the function fails, the return value is 0 (zero). To get extended error information, call GetLastError.

https://blog.csdn.net/Krumitz

与Sample 02中的GetModuleFileNameA相似

# Syntax

C++

```
SHFOLDERAPI SHGetFolderPathW(
 HWND hwnd,
        csidl,
 int
  HANDLE hToken,
  DWORD dwFlags,
  LPWSTR pszPath
);
```

Copy

## **Parameters**

hwnd

Type: HWND

Reserved.

csidl

Type: int

A CSIDL value that identifies the folder whose path is to be retrieved. Only real folders are valid. If a virtual folder is specified, this function fails. You can force creation of a folder by combining the folder's CSIDL with CSIDL\_FLAG\_CREATE.

hToken

#### Type: HANDLE

An access token that can be used to represent a particular user.

Microsoft Windows 2000 and earlier: Always set this parameter to NULL.

Windows XP and later: This parameter is usually set to NULL, but you might need to assign a non-NULL value to hToken for those folders that can have multiple users but are treated as belonging to a single user. The most commonly used folder of this type is Documents.

The calling process is responsible for correct impersonation when hToken is non-NULL. The calling process must have appropriate security privileges for the particular user, including TOKEN\_QUERY and TOKEN\_IMPERSONATE, and the user's registry hive must be currently mounted. See Access Control for further discussion of access control issues.

Assigning the hToken parameter a value of -1 indicates the Default User. This enables clients of SHGetFolderPath to find folder locations (such as the Desktop folder) for the Default User. The Default User user profile is duplicated when any new user account is created, and includes special folders such as My Documents and Desktop. Any items added to the Default User folder also appear in any new user account.



#### Type: DWORD

Flags that specify the path to be returned. This value is used in cases where the folder associated with a <u>KNOWNFOLDERID</u> (or <u>CSIDL</u>) can be moved, renamed, redirected, or roamed across languages by a user or administrator.

The known folder system that underlies SHGetFolderPath allows users or administrators to redirect a known folder to a location that suits their needs. This is achieved by calling <u>IKnownFolderManager::Redirect</u>, which sets the "current" value of the folder associated with the SHGFP\_TYPE\_CURRENT flag.

The default value of the folder, which is the location of the folder if a user or administrator had not redirected it elsewhere, is retrieved by specifying the SHGFP\_TYPE\_DEFAULT flag. This value can be used to implement a "restore defaults" feature for a known folder.

For example, the default value (SHGFP\_TYPE\_DEFAULT) for FOLDERID Music (CSIDL MYMUSIC) is "C:\Usersuser name\Music". If the folder was redirected, the current value (SHGFP\_TYPE\_CURRENT) might be "D:\Music". If the folder has not been redirected, then SHGFP\_TYPE\_DEFAULT and SHGFP\_TYPE\_CURRENT retrieve the same path.

#### SHGFP\_TYPE\_CURRENT

Retrieve the folder's current path.

#### SHGFP\_TYPE\_DEFAULT

Retrieve the folder's default path.

pszPath

#### Type: LPTSTR

A pointer to a null-terminated string of length MAX\_PATH which will receive the path. If an error occurs or S\_FALSE is returned, this string will be empty. The returned path does not include a trailing backslash. For example, "C:\Users" is returned rather than "C:\Users".

https://blog.csdn.net/Krumitz

与Sample\_02中的SHGetSpecialFolderPathA功能类似

CopyFileW函数:

## Syntax

C++

```
BOOL CopyFileW(
LPCWSTR lpExistingFileName,
LPCWSTR lpNewFileName,
BOOL bFailIfExists
);
```

## Parameters

lpExistingFileName

The name of an existing file.

In the ANSI version of this function, the name is limited to MAX\_PATH characters. To extend this limit to 32,767 wide characters, call the Unicode version of the function and prepend "\?" to the path. For more information, see <u>Naming a File</u>.

Tip Starting with Windows 10, version 1607, for the unicode version of this function (CopyFileW), you can opt-in to remove the MAX\_PATH limitation without prepending "\\?\". See the "Maximum Path Length Limitation" section of Naming Files, Paths, and Namespaces for details.

If *lpExistingFileName* does not exist, CopyFile fails, and GetLastError returns ERROR\_FILE\_NOT\_FOUND.

https://blog.csdn.net/Krumitz

lpNewFileName

The name of the new file.

In the ANSI version of this function, the name is limited to MAX\_PATH characters. To extend this limit to 32,767 wide characters, call the Unicode version of the function and prepend "\?" to the path. For more information, see <u>Naming a File</u>.

Tip Starting with Windows 10, version 1607, for the unicode version of this function (CopyFileW), you can opt-in to remove the MAX\_PATH limitation without prepending "\\?\". See the "Maximum Path Length Limitation" section of Naming Files, Paths, and Namespaces for details.

bFailIfExists

If this parameter is TRUE and the new file specified by *lpNewFileName* already exists, the function fails. If this parameter is FALSE and the new file already exists, the function overwrites the existing file and succeeds.

## Return Value

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information, call GetLastError.

https://blog.csdn.net/Krumitz

复制一个文件,参数分别为现有文件名,目标文件名,和一个bool值

如果bool值为TRUE,且目标文件名已存在的话,函数失败

如果为FALSE,目标文件名已存在的话,函数将覆盖已有文件,并成功返回

Copy Copy

### 分析与验证

这就比Sample\_02容易猜测这个程序会执行一个什么操作了,毕竟有一个CopyFileW的存在嘛

所以在这里我们就已经大致可以猜测,这个程序

获取了一个系统的路径

text:00401040 push offset String2 ; "\\<mark>wsample01b</mark>.exe"

并朝这个路径下复制了一个名为wsample01b的可执行文件,而这个可执行文件跟Sample\_03应该是一样的

我们打开Process Monitor来查看一下

17:3	Sample_3.exe	7336 CreateFile	C:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\startup	SUCCESS
17:3	📰 Sample_3. exe	7336 🛃 QueryBasicInforma	C:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\startup	SUCCESS
17:3	📰 Sample_3. exe	7336 🛃 CloseFile	C:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\startup	SUCCESS
17:3	📰 Sample_3. exe	7336 🌋 RegQueryKey	HKLM	SUCCESS
17:3	📰 Sample_3. exe	7336 🌋 RegQueryKey	HKLM	SUCCESS
17:3	📰 Sample_3. exe	7336 🌋 RegOpenKey	HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Explorer\KnownFolderSett	NAME NOT F
	TT - 1		1000 57	

发现Sample\_3在C:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\startup这 个路径下CreateFile

打开这个文件夹,发现

系统 (C:)	▶ 用户	<ul> <li>Administrator</li> </ul>	•	AppData	۲	Roaming	<ul> <li>Microsoft</li> </ul>	۲	Windows ►	「开	始」菜单	۲	程序	F	startup
查看(V)	工具(T)	帮助(H)													
資库中 ▼	共享	▼ 新建文件系	ŧ												
<b>^</b>	名称	*				修改日期	期	类	型		大小				
	🔲 0	mple01b				2015\7 2019\3	7\26 星期	应应	Z用程序 7用程序		58 7	КВ			
		mpicoro				2010 (0	(0 <u>±</u> %)	122	2/13/132/13*		'	ND			

确实有wsample01b的可执行文件

M Stirling - wsample01b.exe	B strategy
ファイル(F) 編集(E) 検索・移動(S) 設定(O) ウィンドウ(W) ヘルプ(H)	
	4 <mark>™.+ °1                                   </mark>
Sample_3.exe	
ADDRESS 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	0123456789ABCDEF
00000000 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	M2
99999918 B8 00 00 00 00 00 00 40 00 00 00 00 00 00	?e. wy wsample01b.exe
99999929 99 99 99 99 99 99 99 99 99 99 9	ADDRESS 88 81 82 83 84 85 86 87 88 89 88 88 80 86 87 88 89 88 80 80 80 80 80 80 80 80 80 80 80 80
00000030 00 00 00 00 00 00 00 00 00 00 0	999999999 40 5A 90 90 93 90 90 94 90 90 97 FF F0 90 90 MZ
00000040 0E 1F BH 0E 00 B4 09 CU 21 B8 01 4C CU 21 54 68	
00000050 09 73 20 70 72 0F 07 72 01 00 20 03 01 0E 0E 0F 00000066 76 20 62 65 20 72 75 65 20 60 65 20 66 65 20	15 program cannu 66666926 00 00 00 00 00 00 00 00 00 00 00 00 00
999999999 44 44 29 02 03 29 72 73 06 29 07 06 29 44 4F 53 29 999999976 6D 6F 64 65 2F 8D 8D 80 24 88 88 88 88 88 88 88	mode \$ 90.000 90.000 90 90 90 90 90 90 90 90 90 90 90 90
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA	送用 注册 注册 969696948 ØE 1F BA ØE 06 B4 09 CD 21 B8 01 4C CD 21 54 68? .? ? ? L? Th
00000090 8B D2 2A CF E6 A4 B6 CF 8B D2 28 CF E6 A4 B6 CF	[Backbed] (1997) [1997] [190
000000A0 88 D2 1C CF F7 A4 B6 CF 88 D2 1D CF E6 A4 B6 CF	达· 20 72 75 6E 20 69 6E 20 44 4F 53 20 t be run in DOS
000000B0 ED DC 25 CF E3 A4 B6 CF E4 A4 B7 CF CA A4 B6 CF	
000000C0 8B D2 19 CF E5 A4 B6 CF 8B D2 2B CF E5 A4 B6 CF	
000000000 52 69 63 68 E4 A4 B6 CF 00 00 00 00 00 00 00 00	
000000E0 50 45 00 00 4C 01 05 00 EC 56 EB 51 00 00 00 00	PE.L 齊隥. 注いはありません FA HA DG CF FA AL B7 CF CA AL B
900000F9 00 00 00 E0 00 02 01 0B 01 0A 00 00 0A 00 00	
99999199 00 0E 00 00 00 00 00 88 13 00 00 10 00 00	E4 A4 B6 CF 69 69 69 69 69 69 69 B C CD # + + + + + + + + + + + + + + + + + +
	·····d······ 4C 01 05 00 EC 56 EB 51 00 00 00 PE 季階
	·····································
	60 90 40 90 90 90 90 90 92 90 90@
99999169 A4 22 89 89 64 88 89 89 89 88 88 88 88 88 88	2 . d
	00000130 00 60 00 00 04 00 00 5B F7 00 00 02 00 00 81 .`[
00000400 00 F0 00 00 FL 04 00 00 F0 00 00 40 00 00 00	90000140 00 00 10 00 00 00 00 00 00 00 00 00 00
	nups//biog.csdn.net/kiunik

用Stirling二进制编辑器,对比Sample3和wsample01b,发现他们是一样的

### 至此即可确定我们之前的分析是大致正确的

## Sample\_03拓展

在IDA中找了很久,都没有找到Sample\_03.exe将获取到的系统路径存放在了哪里

提出疑问,如果不通过进程监视器来看,我们应该怎么知道程序向哪条路径CopyFile了呢

### 我们选择使用OllyDbg调试,将Sample\_03拖入到OllyDbg中,如下显示

	ICBMH			
Octobilition         CS         SS         PUSH EEP           00401001         08EC         NOV EEP, ESP           00401001         58         04200000         HOV EEX, 2004           00401002         E8         0300000         CRLL 00401860	Allocates 8196, bytes on stack	ĺ	Registers (FPU)     EXX 002000     EXX 002000     EXX 002000     EXX 0020000     EXX 00000000     EXX 000000000	A
00401012         30C5         XOR EX, EBP           00401012         33C5         XOR EX, EBP           00401014         8946 FC         NU DWORD PTR SS.CLOCAL.13.EAP           00401017         68.0010040         PUSH 1000           00401017         68.0010040         PUSH 1000           00401017         5005 FCCFFFFLER ERX.LLOCAL.20491           00401022         50	Count = 4096. Buffer => OFFSET LOCH2049		EBX 00000000 ESP 0010FF94 EBP 0010FF98 ESI 00000001 EDI 00403378 Sample_3.00403378	
00401023 • 64 00 PUSH 0 00401025 • FFI5 0420400 CALL DUORD PTR 05:[(AKERNEL32. 00401026] • 8060 FCEFFF LEA ECX.[LOCAL.1025] 00401031 • 51 00401032 • 64 00 PUSH 0 00401032 • 64 00 PUSH 0	Settlodu le REPELS2. Settlodu le FiletlaneU		EIP 00401000 30mpte_3.00401000 C 8 ES 0020 22bit 0(FFFFFFF) P 0 CS 0023 22bit 0(FFFFFFF) R 0 SS 0023 22bit 0(FFFFFFF) Z 0 DS 0023 22bit 0(FFFFFFFF)	✤ 拖拽上传
00401035 80401030 80401030 FF15 <u>8420400</u> CRLL DW0RD PTR D51745HELL82.5 80401046 60401046 6041125 80401046 6041125 80401045 804010000000000000000000000000000000000	HBerFolde Sro = "vusanple01b.exe"		S 0 FS 0055 32511 (FEDU0001FFF) 0 0 LastEvr 00000000 ERROR_SUCCESS EFL 000003202 (N0.HB.NE.A.NS.P0.0E.0)	
200010080 - 523 00204000 CHL DUORD PTR DS:((&KERNEL32. 80401052) - 6715 00204000 CHL DUORD PTR DS:((&KERNEL32. 80401052) - 6005 FCEFFFFFL0 EAC,(LOCAL.1025) 804010531 - 500 FCEFFFFL0 EAC,(LOCAL.2049) 804010531 - 8000 FCEFFFFL0 ECK,(LOCAL.2049)	IstroatW) (Fall IOUALIAS FallIfeliate = FALSE HeuFileHame => OFFSET LOCAL.1025		ST0 enoty 0.0 ST1 enoty 0.0 ST2 enoty 0.0 ST2 enoty 0.0 ST4 enoty 0.0	
00401051 • 51 PUSH ECX 00401052 • FF15 0020400 CRLL DWORD PTR DS: (C4KERNEL32, 00401053 • 8840 FC HOU ECX, DWORD PTR SS: (LOCAL, 1) 004010561 • 33C0 XOR ECX, EEP 00401057 • 33C0 XOR EXX, EEX 00401057 • E2 27000000 CRL • 00401020	CopyFileV		515 embry 0.0 516 embry 0.0 517 embry 0.0 FST 0000 Cond 0 0 0 E Err 0 0 0 0 0 0 0 0 0 0 (61) FST 0000 Cond 0 0 0 E Err 0 0 0 0 0 0 0 0 0 (61) FCW 0277 Free MERES Task 1 1 1 1 1 1	
00401374         -         SBES         MOU ESP. EEB*           00401376         -         C3         RFIM           00401376         -         C3         RFIM           00401376         -         C3         RFIM           00401376         -         C3         RFIM           00401376         C4         1113         1113           00401377         C5         11173         1113           00401377         C4         11173         1113           00401377         C4         11173         1113           00401377         C4         11173         1113			Last ond UBBU UBBUBBBB Tomps processors because descense Tomps processors Tomps p	
00401007F 00401200 F8 E3 7EFFFFF CPLL 00401200 00401205 - 64 08 PUSH 0 00401205 - 64 08 PUSH 0FFET 00402194 00401205 - 68 34214000 PUSH 0FFET 00402144 00401205 - FF15 C232000 CPLL 00407 PTR 0514 (AUSER32.64	Savo Le_ 3. 8046118881 guessed Rcg1, Rcg2, Rcg8, Rcg41 [Joc = HB_0K.180, CFR0UTTON118B_APFL100AL Cupton = "MESSBAC Text = "Colod" Het Livekie (CERSE.2014011)		MCSR 00001F00 F2 0 D2 0 Err 0 0 0 0 0 Rnd NERR Hask 1 1 1 1 1	
00401997         -         DET G 0220400         DET LBC           00401997         -         DET G 0200         RET N 10           0040197         -         DET G 02         JRE SHORT 004010FD           0040197         -         75 02         JRE SHORT 004010FD           00401075         -         75 02         JRE SHORT 004010FD	ssageBoxV USERS2. HessageBoxV			
Breating A to become an arrow of a sec Stack (0019FE9)=Sample_3.00480804 EBr-0019FF80 Local call from 401080			× *	
Address Hex dunp	ASCI1 #500/33b	<ul> <li>CONSIDER CORPORADO VIE. RETURN From Sample 3. D018FEF8 C08401255 U89. RETURN From Sample 5.</li> </ul>	00401000 to Sample 3.00401005 00401000 to Sample 3.00401255	*
00403010 FE FF FF FF 10 01 00 00 00 00 00 00 00 00 00 00 00	5. <u>0</u>	0013FFEC 004000001%. Rrg1 = Sample_3. <stru 9013FF00 00000000Rrg2 = 0 0013FF04 00332306 etc., Rrg3 = 832306 0013FF00 0000000Rrg4 = 04 0013FF0C FFC1755 Mp<sup>4</sup>n</stru 	CT INRGE_DOS_HERDER>	
204433763         08         08         00         00         08         00		0018/F14 00000000 0018/F18 7EFDE000 0018/F10 00000044 0 0018/F20 0008239E 1m. 0018/F20 0083239E 1m. UNICODE "Winsta0\Defa	w1t"	
004430201         00		0013FF20 0000000 0013FF20 0000000 0013FF30 0000000 0013FF38 0000000 0013FF38 0000000 0013FF38 0000000		
00403123         00         <		0013FF40 0000000 0013FF40 0000000 0013FF40 00000001 0013FF40 00000001 0013FF60 0000000 0013FF60 0000000		
C014403         C03         C03 <thc03< th=""> <thc03< <="" td=""><td></td><td>013FF58 00000000 013FF50 0010F04 ' t 013FF50 0010F04 ' t 013FF60 0000000 013FF60 00000000 013FF60 00000000</td><td>QueryPerformanceCounter to Sample_3.0040186E</td><td></td></thc03<></thc03<>		013FF58 00000000 013FF50 0010F04 ' t 013FF50 0010F04 ' t 013FF60 0000000 013FF60 00000000 013FF60 00000000	QueryPerformanceCounter to Sample_3.0040186E	
Control Libre         Control		0018FF70 0018FF0C.1.1. 0018FF74 FA025F2 2X* 0018FF78 0018FF74 - 1. 0018FF78 0018FF74 - 1. 0018FF70 00401709 48.5% handler 0018FF50 EF90CP8 "*"n 0018FF50 EF90CP8	ecord	
Importance         Importa		0015FF00 0015FF01 *** 0015FF00 0015FF01 *** 0015FF00 0015FF01 *** 0015FF01 0015FF01 0** 0015FF01 0015FF01 0** 0015FF01 0015FF01 0** 0015FF01 0*** 0015FF01 0*** 0015FF01 0***	61336A 902	https://blog.csdn.net/Krumi

在00401000处设置断点,然后按下F8逐步执行指令

#### 运行到0040103A处,显示如下



此处即将执行SHGetFolderPathW函数

### 运行到下一行后,显示如下

00401038	· · ·	6Н. 00	PUSH	0	
0040103H	_	FF15 B420400	CHLL	DWORD PTR DS: [<&SHELL32.SHGetFolde	One of the second to Other second
00401040	•	0000 CCCCCCCC	PUSH	UFFSET 00402114	Src = "Nwsamplevib.exe"
00401045	11	E2 FUEFFFF	DIICU	EDA, LLUCHL. 1025J	Dest -> DEESET LOCAL 182E
00401046	11	FE15 0820400	COLL	DWORD PTR DS+F/&KERNEL32 [stroatW]	Dest -/ OFFBEI LOCHL.1025
00401052	11	69 00	PUSH	A A A A A A A A A A A A A A A A A A A	FEATLIFEVIATE = FOLSE
00401054	I.	8085 ECEFEEE	I FA	EĂX. ELOCAL. 10251	
00401050	١.	50	PUSH	EAX	NewFileName => OFFSET LOCAL.1025
0040105B	١.	808D FCDFFFFI	LÊÂ	ECX.[LOCAL.2049]	
00401061	۰.	51	PUSH	ECX	ExistingFileName => OFFSET LOCAL.2049
00401062	۱۰.	FF15 0020400	CALL	DWORD PTR DS:[<&KERNEL32.CopyFileW	↓KERNEL32.CopyFileW
00401068	۱· .	8B4D FC	MOV	ECX,DWORD PTR SS:[LOCAL.1]	
0040106B	۱· ۱	33CD	XOR	ECX, EBP	
00401060	<u>۰</u>	3300	XUR	EHX,EHX	
00401055	· ·	E8 2F000000	MOLL	004010H3	
00401074	11	ED DED	BOB	EOF, EDF	
00401077	L:	63	RETN	CDF	
00401078	-	čč l	INT3		
00401079		čč l	<b>ÎNT</b> 3		
0040107A		čč l	<b>INT3</b>		
0040107B		ČČ	<b>INT3</b>		
0040107C		CC	INT3		
0040107D		CC	INT3		
0040107E		CC	INT3		
0040107F		CC DEFERENCE	INT3	00404000	
00401080	r۹.	E8 7BFFFFFF	CHLL	00401000	Sample_3/04401080(guessed Hrg1,Hrg2,Hrg3,Hrg4)
00401085	· ·	6H 00	PUSH	00000000	TIME = HE_UKTHE_DEFBUTIONITHE_HPPENODHL
00401007	11	CO 44214000	DUCU	OFFSET 00402134	
00401000	11	EE15 C020400	COLL	DWDRD_PTR_DS+[/&USER32_GetOctivebli	TISER22 GetOct inellindou
00401097		50	PUSH	F9X	bluner
00401098	١.	FF15 BC20400	CALL	DWORD PTR DS: [<&USER32.MessageBoxW	-USER32. MessageBoxW
0040109E	•	3300	XOR	EAX.EAX	
004010A0	ι.	C2 1000	RETN	10	
004010A3	ſ٩	3B0D 0030400	CMP	ECX,DWORD PTR DS:[403000]	
00401009	••	75 02	JNE	SHORT 004010AD	
004010HB	<b>ب</b>	F3:C3	REP	RETN	
004010HU	~~	E9 E0020000	DHP	00401392 99491500	
Stack [00	1806	E81-0018FFFC	LINT	CODE "C:\Users\Odministrator\OppDat	NRoaming\Microsoft\Mindows\Start Menu\Programs\Startup"
Imm=Samp1	e 3.	00402114, UNI	CODE	"\wsampleR1b.exe"	

然后我们再确认一下Stack[0018DEE8]的内容

可以看到,虽然我们不知道SHGetFolderPathW函数设置了什么参数,但是我们知道了执行完这个函数之后,系统将获取到的路径push入栈,而我们的目的也达到了,寻找到了Sample\_03.exe向哪一个路径复制了程序

UNICODE "C:NUsersN8dministratorN8opDataN8opmingN1crosoftNUindowsNStart MenuNProgramsNStartur

最后我们用32位的IDA PRO查看我们分析范围0x00401000至0x00401062的这段子函数的代码,以确认之前逆向分析的结果(32位的可以按F5查看程序伪代码,64位的好像并没有这个功能)

```
int sub_401000()
{
    WCHAR Filename; // [sp+0h] [bp-2004h]@1
    WCHAR pszPath; // [sp+1000h] [bp-1004h]@1
    GetModuleFileNameW(0, &Filename, 0x1000u);
    SHGetFolderPathW(0, 7, 0, 0, &pszPath);
    lstrcatW(&pszPath, L"\\wsample01b.exe");
    CopyFileW(&Filename, &pszPath, 0);
    return 0;
}
```

可以发现,是大致正确的

6.实验01总结

三个实验做下来,其实并没有"写下这篇实验报告类型的博客"看起来这么简单与轻描淡写

还是花费了许多时间的

三个实验中Sample\_01和Sample\_03还是比较让人满意的

Sample\_02就真的,太打击人了

希望在日后的学习中可以继续好好努力,不耻下问

至少总有一天得搞懂Sample\_02的函数吧(小声bb)

虽然不是第一次在CSDN上发文了,(虽然以前也只发过一次)

但是这次有一点赶作业的嫌疑,并没有怎么认真排版,如果让大家阅读不便,敬请谅解!

附上三个Sample的IDA DATABASE文件和做作业时候的参考网址吧,原执行文件就不附上了

链接: https://pan.baidu.com/s/1v227QBImELFtbgMqmi8u5A 提取码: syll

http://www.ituring.com.cn/book/tupubarticle/9632