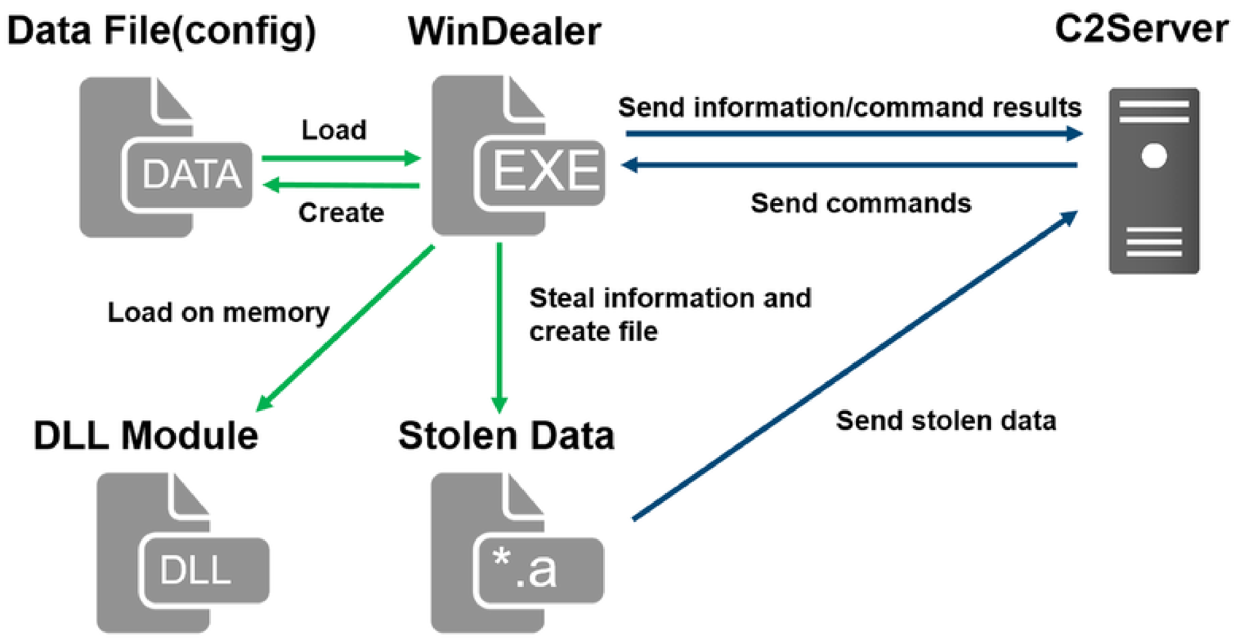
Malware WinDealer used by LuoYu Attack Group

blogs.jpcert.or.jp/en/2021/10/windealer.html



増渕 維摩(Yuma Masubuchi)

October 26, 2021

Email

During JSAC2021 on 28 January 2021, there was a presentation about an attack group LuoYu, which targets Korean and Japanese organisations since 2014 [1][2]. Recently, JPCERT/CC came across malware WinDealer used by this group. This article introduces some findings of our analysis.

Malware WinDealer overview

WinDealer steals information of an infected PC and sends it to a C2 server as described in Figure 1.

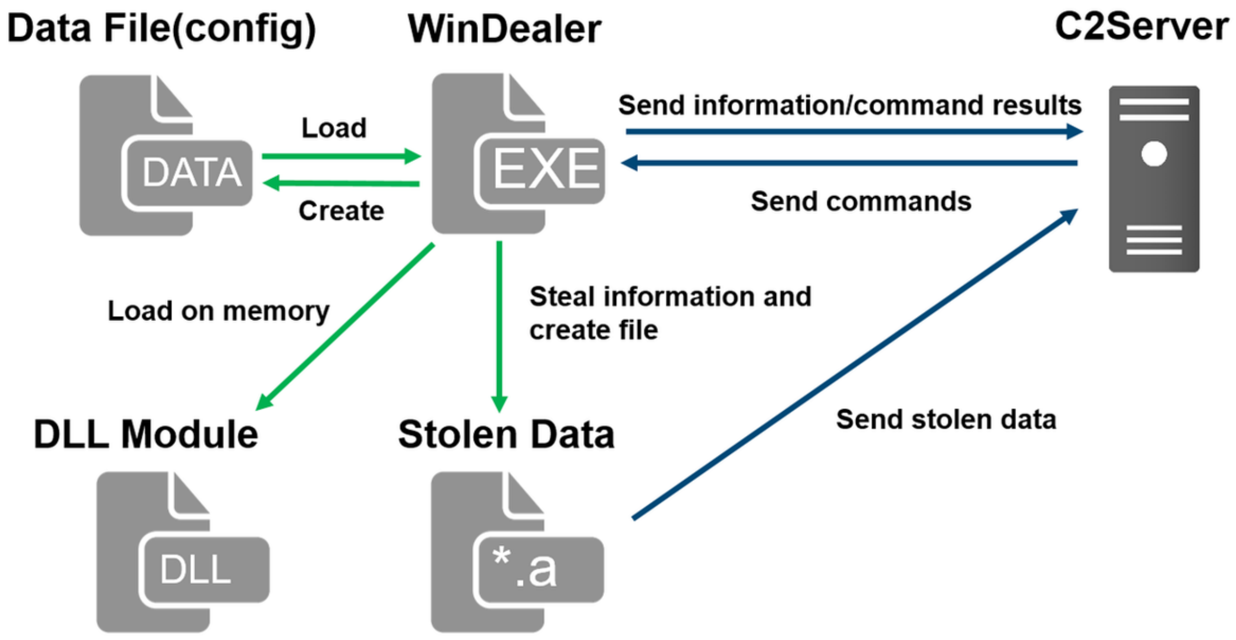


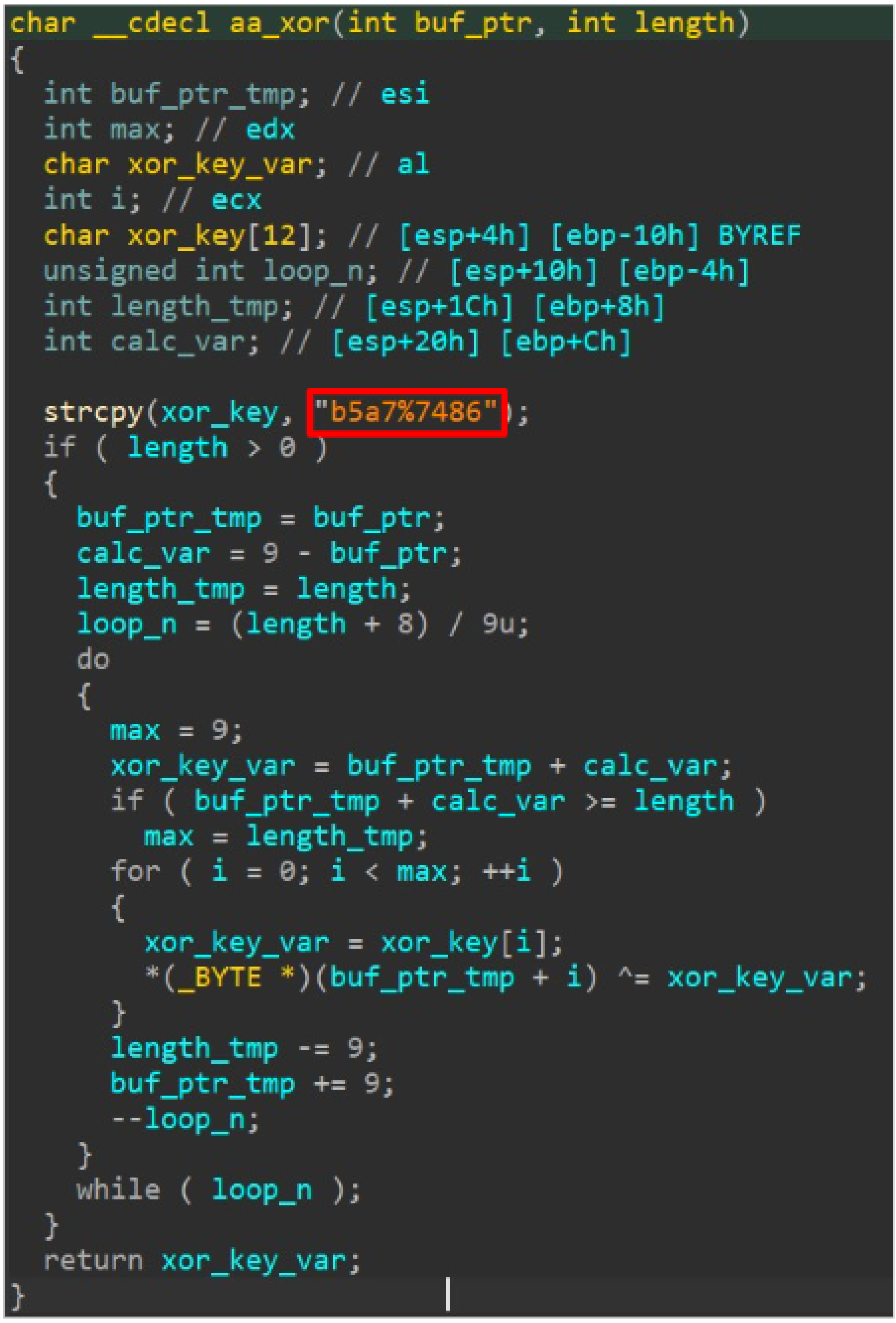
Figure 1： Malware WinDealer behaviour overview

Once launched, the malware reads configuration from a file under C:\ProgramData and loads a DLL module on its memory. It steals information about the victim PC, network configuration and SNS applications etc. and saves them in a file with an ".a" extension under %TEMP%, which is then sent to a C2 server.
The following points will be described in the next sections.

- Read configuration
- Communicate with C2 servers
- Process and send stolen data
- Functions of modules loaded on memory

Read configuration

The malware stores its configuration in several folders under C:\ProgramData and reads it when executed. The contents are encoded based on XOR with its key value "b6a7%7486". Please refer to Appendix A for the configuration file path and its contents. Figure 2 shows a function to decode configuration.



```
char __cdecl aa_xor(int buf_ptr, int length)
{
  int buf_ptr_tmp; // esi
  int max; // edx
  char xor_key_var; // al
  int i; // ecx
  char xor_key[12]; // [esp+4h] [ebp-10h] BYREF
  unsigned int loop_n; // [esp+10h] [ebp-4h]
  int length_tmp; // [esp+1Ch] [ebp+8h]
  int calc_var; // [esp+20h] [ebp+Ch]

  strcpy(xor_key, "b5a7%7486");
  if ( length > 0 )
  {
    buf_ptr_tmp = buf_ptr;
    calc_var = 9 - buf_ptr;
    length_tmp = length;
    loop_n = (length + 8) / 9u;
    do
    {
      max = 9;
      xor_key_var = buf_ptr_tmp + calc_var;
      if ( buf_ptr_tmp + calc_var >= length )
        max = length_tmp;
      for ( i = 0; i < max; ++i )
      {
        xor_key_var = xor_key[i];
        *(_BYTE *)(buf_ptr_tmp + i) ^= xor_key_var;
      }
      length_tmp -= 9;
      buf_ptr_tmp += 9;
      --loop_n;
    }
    while ( loop_n );
  }
  return xor_key_var;
}
```

Figure 2 : Function to decode a file storing configuration

Communicate with C2 servers

If the following configuration files exist in the designated folder, WinDealer loads the C2 server information from them and starts communicating.

- C:\ProgramData\ad5f82e8
- C:\ProgramData\1c76cbfe
- C:\ProgramData\9c3b6294

If no such file exists, WinDealer communicates to a random IP address in one of the following ranges (port 6999/UDP or 55556/TCP). It switches to an IP address in the other range at a certain interval.

- 113.62.0.0 - 113.63.255.255
- 111.120.0.0 - 111.123.255.255

Figure 3 shows the malware's communication flow with its C2 server. First, it encrypts an AES key with RSA algorithm and sends to a C2 server. Information stolen from a victim PC is encrypted with this AES key and sent to a C2 server at a certain interval. After that, C2 server sends a command to the victim PC. The malware executes it and sends the result to the C2 server after encryption. Besides the data exchange, the malware also communicates with domains such as www[.]microsoftcom (non-existent at the moment) and icanhazip[.]com.

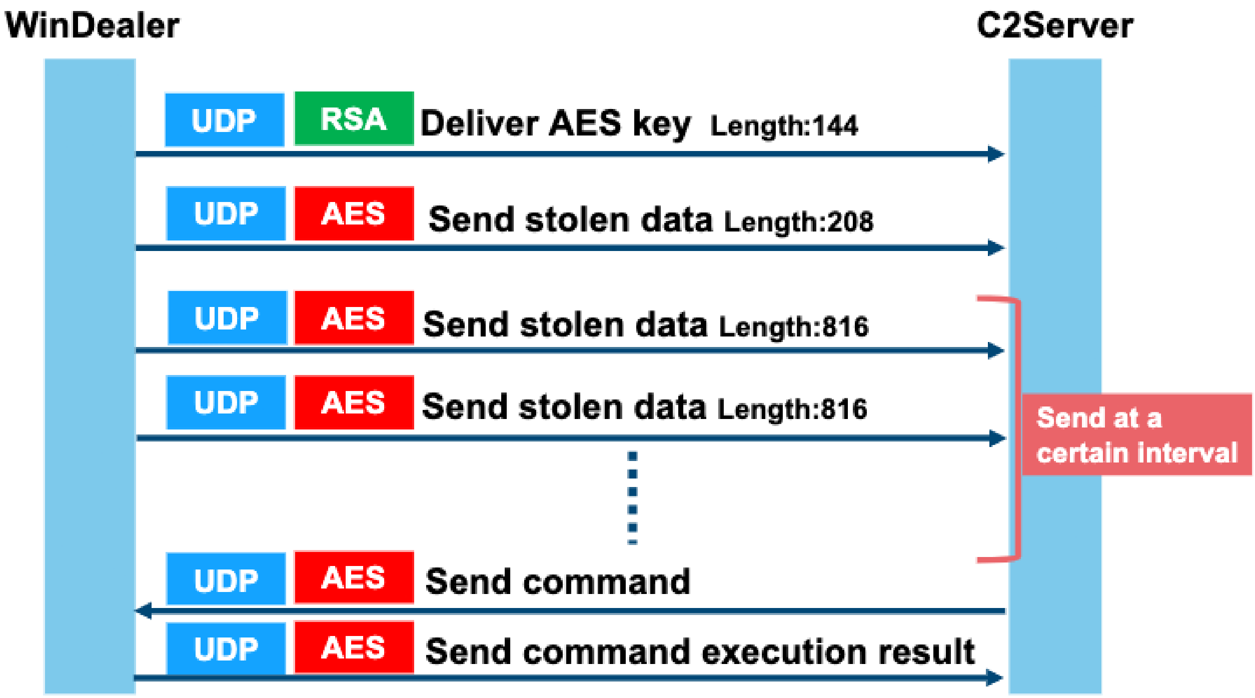
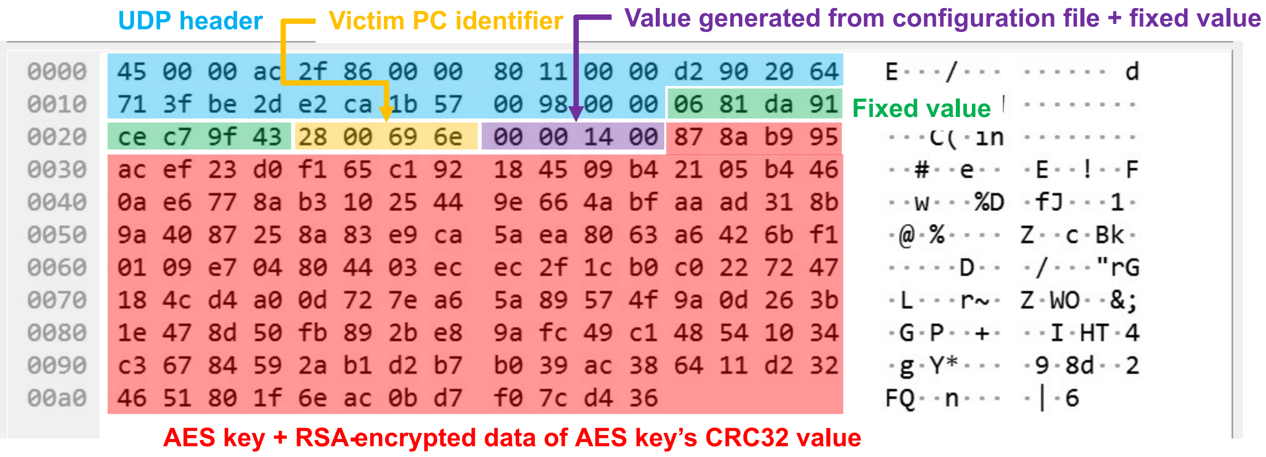

WinDealer
C2Server
UDP
RSA
Deliver AES key Length:144
UDP
AES
Send stolen data Length:208
UDP
AES
Send stolen data Length:816
UDP
AES
Send stolen data Length:816
Send at a certain interval
UDP
AES
Send command
UDP
AES
Send command execution result

Figure 3: Communication flow with a C2 server

Figure 4 describes the communication contents when delivering an AES key. AES key and its CRC32 checksum value are encrypted with RSA1024bit public key. The public key is hardcoded in the sample, which is also used for other samples as well.


UDP header
Victim PC identifier
Value generated from configuration file + fixed value
Fixed value
AES key + RSA-encrypted data of AES key's CRC32 value

```
0000  45 00 00 ac 2f 86 00 00  80 11 00 00 d2 90 20 64   E···/··· ······ d
0010  71 3f be 2d e2 ca 1b 57  00 98 00 00 06 81 da 91            ········
0020  ce c7 9f 43 28 00 69 6e  00 00 14 00 87 8a b9 95   ···C(·in ········
0030  ac ef 23 d0 f1 65 c1 92  18 45 09 b4 21 05 b4 46   ··#··e·· ·E··!··F
0040  0a e6 77 8a b3 10 25 44  9e 66 4a bf aa ad 31 8b   ··w···%D ·fJ···1·
0050  9a 40 87 25 8a 83 e9 ca  5a ea 80 63 a6 42 6b f1   ·@·%···· Z··c·Bk·
0060  01 09 e7 04 80 44 03 ec  ec 2f 1c b0 c0 22 72 47   ·····D·· ·/···"rG
0070  18 4c d4 a0 0d 72 7e a6  5a 89 57 4f 9a 0d 26 3b   ·L···r~· Z·WO··&;
0080  1e 47 8d 50 fb 89 2b e8  9a fc 49 c1 48 54 10 34   ·G·P··+· ··I·HT·4
0090  c3 67 84 59 2a b1 d2 b7  b0 39 ac 38 64 11 d2 32   ·g·Y*··· ·9·8d··2
00a0  46 51 80 1f 6e ac 0b d7  f0 7c d4 36               FQ··n··· ·|·6
```

Figure 4： Example of contents sent with AES key

From the second round of communication and onwards, data is encrypted in AES128bit ECB mode based on the AES key which was dynamically generated during the initial communication. Please refer to Appendix B for the details of data format.

Process and send stolen data

WinDealer processes a series of stolen data as ".a" file in a folder under %TEMP%, encrypts it with AES and send it to a C2 server. The flow of event is illustrated in Figure 5. The modules steal and process the data, while WinDealer itself monitors the files under %TEMP%, encrypts the file and sends it to a C2 server.

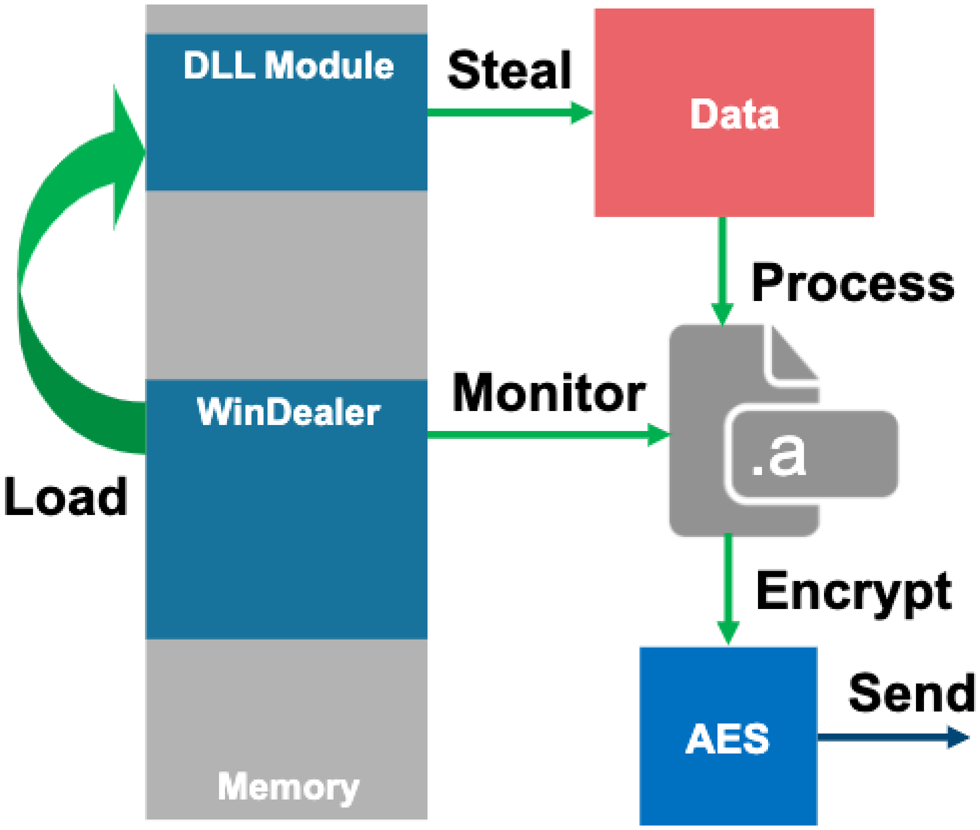
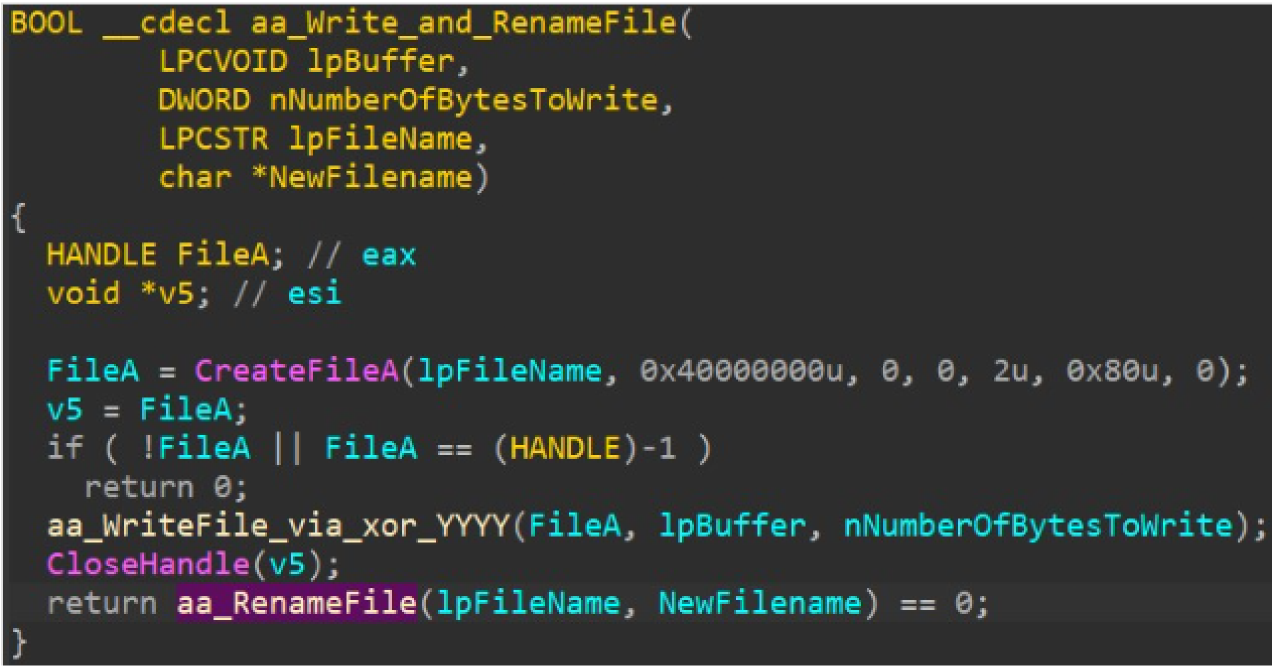


Figure 5 : Flow of events by WinDealer and modules

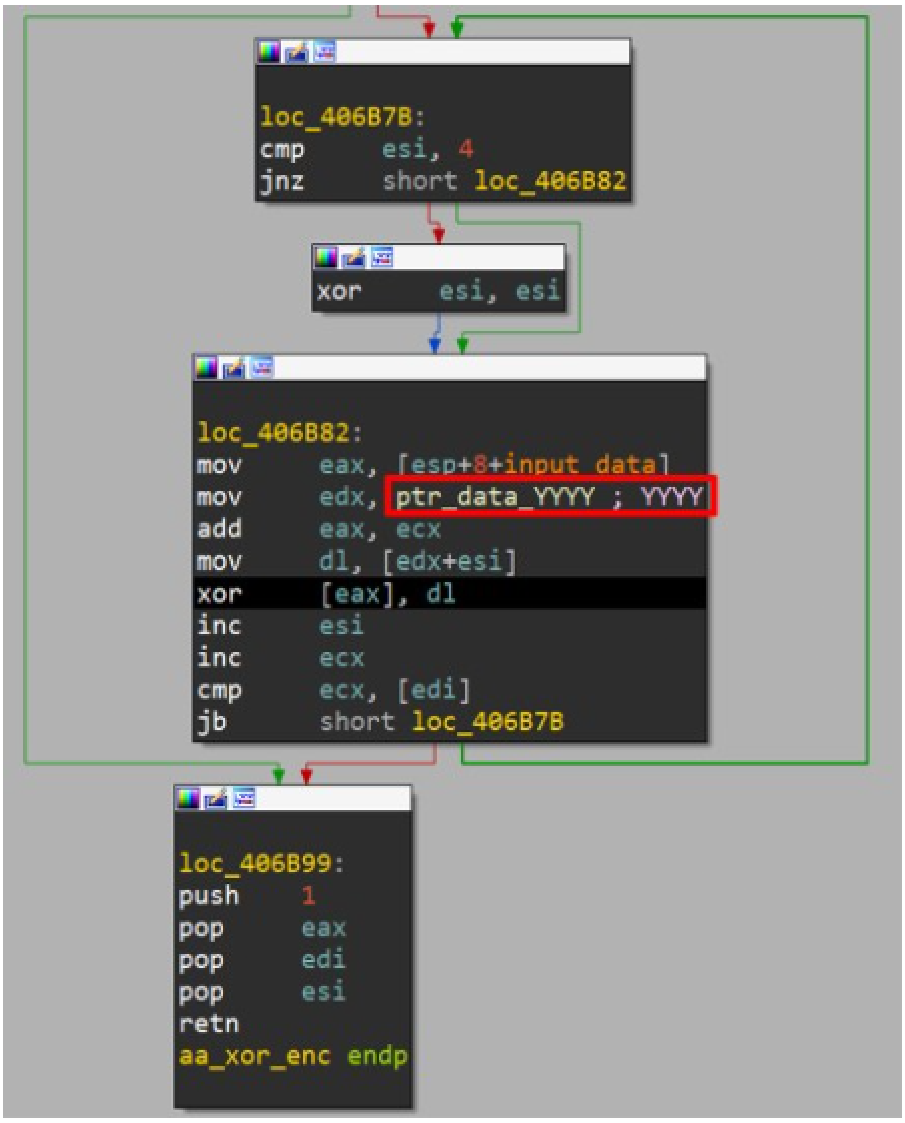
A part of the code for generating “.a” file by a module is as follows (Figure 6):



```
BOOL __cdecl aa_Write_and_RenameFile(
        LPCVOID lpBuffer,
        DWORD nNumberOfBytesToWrite,
        LPCSTR lpFileName,
        char *NewFilename)
{
  HANDLE FileA; // eax
  void *v5; // esi

  FileA = CreateFileA(lpFileName, 0x40000000u, 0, 0, 2u, 0x80u, 0);
  v5 = FileA;
  if ( !FileA || FileA == (HANDLE)-1 )
    return 0;
  aa_WriteFile_via_xor_YYYY(FileA, lpBuffer, nNumberOfBytesToWrite);
  CloseHandle(v5);
  return aa_RenameFile(lpFileName, NewFilename) == 0;
}
```

Figure 6 : DLL module’s code to generate “.a” file

The stolen data is first stored in a file with an “.t” extension, which is then renamed to “.a”. The series of data is stored in different directories based on the data category, and they are taken out when “.a” file is created. Please refer to Appendix E for the details of each directory.
Before writing and reading the files, the data is encoded/decoded by XOR-based function with its key value “YYYY” as in Figure 7.

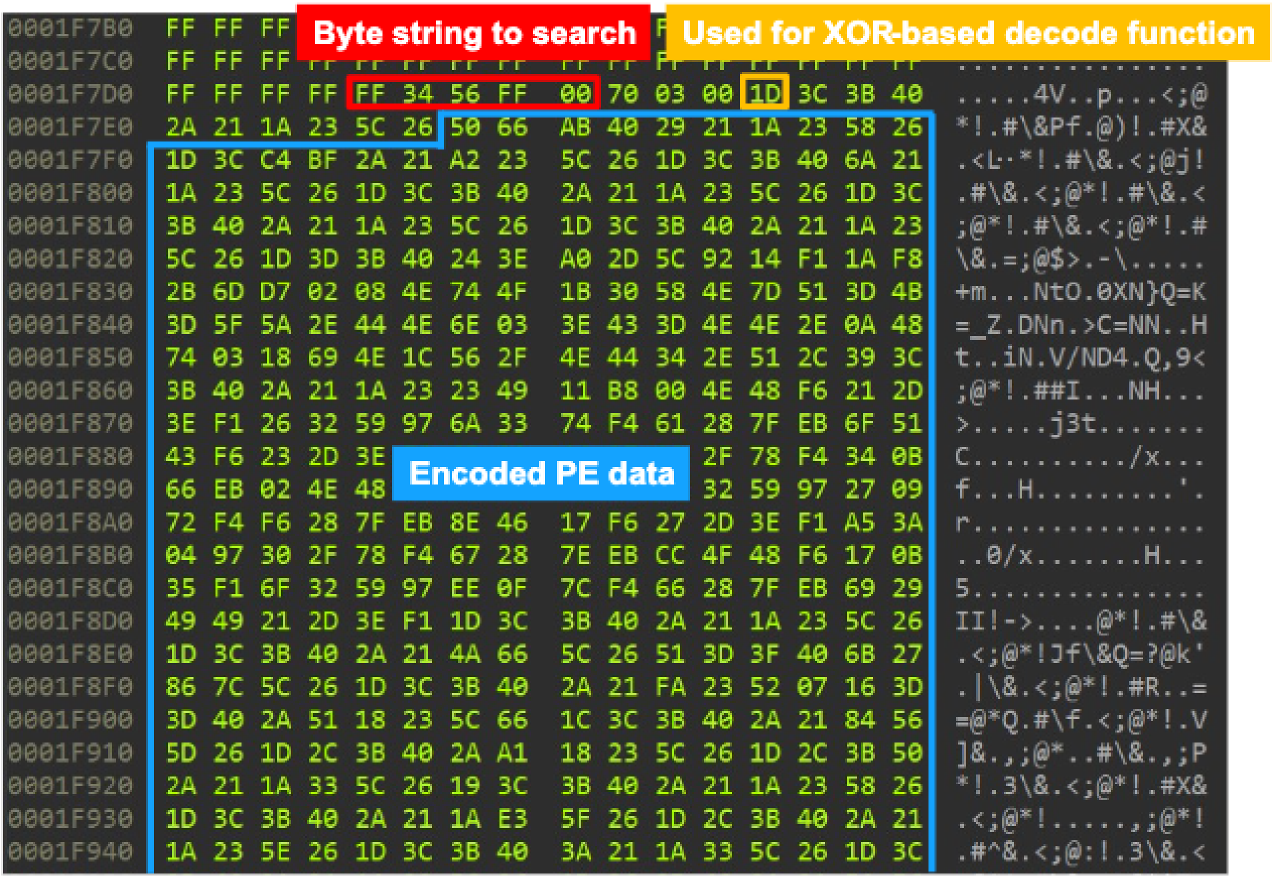


```
loc_406B7B:
cmp     esi, 4
jnz     short loc_406B82

xor     esi, esi

loc_406B82:
mov     eax, [esp+8+input_data]
mov     edx, ptr_data_YYYY ; YYYY
add     eax, ecx
mov     dl, [edx+esi]
xor     [eax], dl
inc     esi
inc     ecx
cmp     ecx, [edi]
jb      short loc_406B7B

loc_406B99:
push    1
pop     eax
pop     edi
pop     esi
retn
aa_xor_enc endp
```

Figure 7：A function for XOR-based encoding when accessing ".a" file

Functions of modules loaded on memory

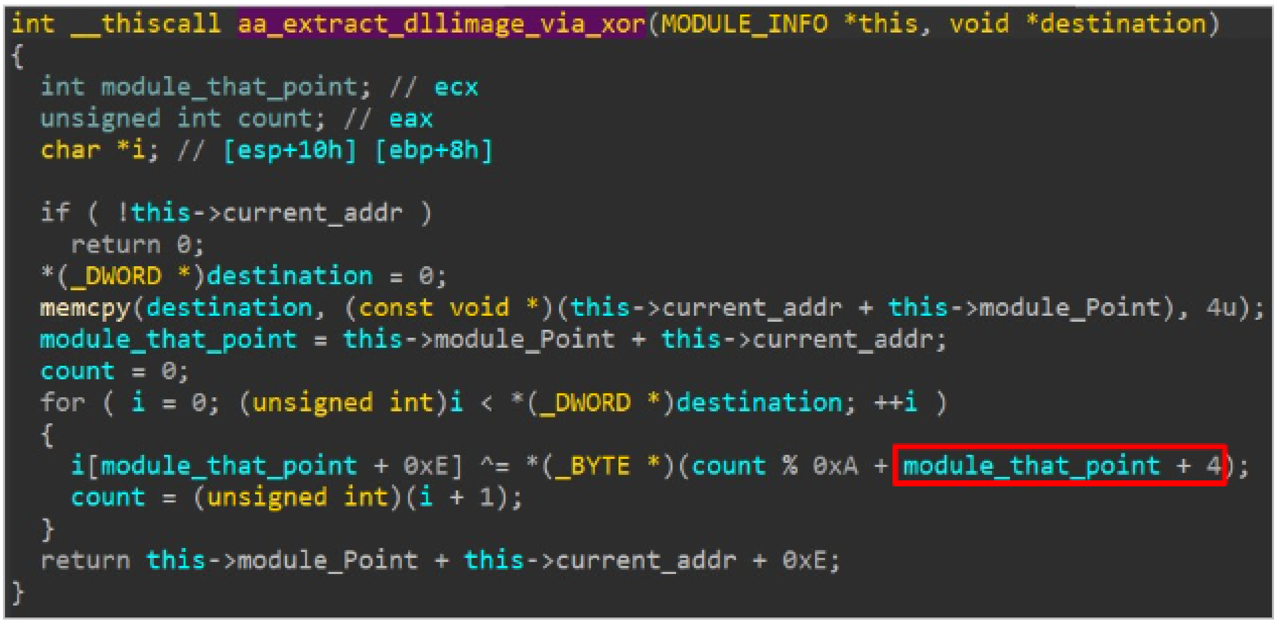
Once launched, WinDealer loads a DLL module in a PE format (encoded in the sample) on the memory and executes it (Figure 8).



```
0001F7B0  FF FF FF
0001F7C0  FF FF FF                                          ................
0001F7D0  FF FF FF FF FF 34 56 FF  00 70 03 00 1D 3C 3B 40  .....4V..p...<;@
0001F7E0  2A 21 1A 23 5C 26 50 66  AB 40 29 21 1A 23 58 26  *!.#\&Pf.@)!.#X&
0001F7F0  1D 3C C4 BF 2A 21 A2 23  5C 26 1D 3C 3B 40 6A 21  .<L·*!.#\&.<;@j!
0001F800  1A 23 5C 26 1D 3C 3B 40  2A 21 1A 23 5C 26 1D 3C  .#\&.<;@*!.#\&.<
0001F810  3B 40 2A 21 1A 23 5C 26  1D 3C 3B 40 2A 21 1A 23  ;@*!.#\&.<;@*!.#
0001F820  5C 26 1D 3D 3B 40 24 3E  A0 2D 5C 92 14 F1 1A F8  \&.=;@$>.-\.....
0001F830  2B 6D D7 02 08 4E 74 4F  1B 30 58 4E 7D 51 3D 4B  +m...NtO.0XN}Q=K
0001F840  3D 5F 5A 2E 44 4E 6E 03  3E 43 3D 4E 4E 2E 0A 48  =_Z.DNn.>C=NN..H
0001F850  74 03 18 69 4E 1C 56 2F  4E 44 34 2E 51 2C 39 3C  t..iN.V/ND4.Q,9<
0001F860  3B 40 2A 21 1A 23 23 49  11 B8 00 4E 48 F6 21 2D  ;@*!.##I...NH...
0001F870  3E F1 26 32 59 97 6A 33  74 F4 61 28 7F EB 6F 51  >.....j3t.......
0001F880  43 F6 23 2D 3E                    2F 78 F4 34 0B  C........../x...
0001F890  66 EB 02 4E 48                    32 59 97 27 09  f...H.........'.
0001F8A0  72 F4 F6 28 7F EB 8E 46  17 F6 27 2D 3E F1 A5 3A  r...............
0001F8B0  04 97 30 2F 78 F4 67 28  7E EB CC 4F 48 F6 17 0B  ..0/x.......H...
0001F8C0  35 F1 6F 32 59 97 EE 0F  7C F4 66 28 7F EB 69 29  5...............
0001F8D0  49 49 21 2D 3E F1 1D 3C  3B 40 2A 21 1A 23 5C 26  II!->....@*!.#\&
0001F8E0  1D 3C 3B 40 2A 21 4A 66  5C 26 51 3D 3F 40 6B 27  .<;@*!Jf\&Q=?@k'
0001F8F0  86 7C 5C 26 1D 3C 3B 40  2A 21 FA 23 52 07 16 3D  .|\&.<;@*!.#R..=
0001F900  3D 40 2A 51 18 23 5C 66  1C 3C 3B 40 2A 21 84 56  =@*Q.#\f.<;@*!.V
0001F910  5D 26 1D 2C 3B 40 2A A1  18 23 5C 26 1D 2C 3B 50  ]&.,;@*..#\&.,;P
0001F920  2A 21 1A 33 5C 26 19 3C  3B 40 2A 21 1A 23 58 26  *!.3\&.<;@*!.#X&
0001F930  1D 3C 3B 40 2A 21 1A E3  5F 26 1D 2C 3B 40 2A 21  .<;@*!.....,;@*!
0001F940  1A 23 5E 26 1D 3C 3B 40  3A 21 1A 33 5C 26 1D 3C  .#^&.<;@:!.3\&.<
```

Figure 8： Encoded module data

The malware obtains its file path, searches for a byte string "0xFF3456FF00" and extracts data from its offset 0xE. Using its offset 0x4 value and a XOR-based decode routine (Figure 9), a DLL module is loaded on the memory and then executed.



```
int __thiscall aa_extract_dllimage_via_xor(MODULE_INFO *this, void *destination)
{
  int module_that_point; // ecx
  unsigned int count; // eax
  char *i; // [esp+10h] [ebp+8h]

  if ( !this->current_addr )
    return 0;
  *(_DWORD *)destination = 0;
  memcpy(destination, (const void *)(this->current_addr + this->module_Point), 4u);
  module_that_point = this->module_Point + this->current_addr;
  count = 0;
  for ( i = 0; (unsigned int)i < *(_DWORD *)destination; ++i )
  {
    i[module_that_point + 0xE] ^= *(_BYTE *)(count % 0xA + module_that_point + 4);
    count = (unsigned int)(i + 1);
  }
  return this->module_Point + this->current_addr + 0xE;
}
```

Figure 9： Decoding module

The loaded DLL module is named as “MozillaDll.dll”. There are 3 Export functions as follows:

- AutoGetSystemInfo: Steal data
- GetConfigInfo: Set configuration
- partInitOpt: Set commands

The loaded DLL module monitors the below items, saves related items in a separate file and obtains them to send out to a C2 server.

- Files stored in a USB memory
- Files under Documents, desktop and recycle bin
- Files under folders related to SNS applications

Please see Appendix D for the details of commands that C2 server sends and its contents.

In closing

Besides WinDealer, it has been confirmed that LuoYu uses other kinds of malware that operate in various platforms. We will report if we observe a new type of malware.
For your reference, SHA256 hash values of similar samples are listed in Appendix F.

- Yuma Masubuchi
(Translated by Yukako Uchida)

Reference

[1] “LuoYu” The eavesdropper sneaking in multiple platforms
https://jsac.jpcert.or.jp/archive/2021/pdf/JSAC2021_301_shui-leon_en.pdf

[2] Japan Security Analyst Conference 2021 -3rd Track-
https://blogs.jpcert.or.jp/en/2021/02/jsac2021report1.html

Appendix A WinDealer configuration

File path	String in malware	Contents
C:\ProgramData\923b5fd7	remark	-
C:\ProgramData\ad5f82e8	remotedomain	Domain name
C:\ProgramData\8fe4c114	password	-

C:\ProgramData\1c76cbfe	remoteip	C2 server IP
C:\ProgramData\9c3b6294	reverseip	C2 server IP (reconfigured)
C:\ProgramData\789406d0	-	Result of connection to a dummy host
C:\ProgramData\c25549fe	otherinfo	-
C:\ProgramData\f46d373b	-	Created when launched
C:\ProgramData\windows.inf	-	-
C:\ProgramData\Destro	-	Name information to register in run key

Table A：List of configuration

Appendix B WinDealer Contents of data exchanged

Offset	Length (byte)	Contents
0x00	4	0x91DA8106
0x04	4	0x439FC7CE
0x08	4	Victim PC identifier
0x0C	1	Generated based on the contents of a configuration file "789406d0"
0x0D	3	0x001400
0x10	128	AES key + RSA-encrypted data of AES key’s CRC32 value

Table B-1： Format of data sent for first communication

Offset	Length (byte)	Contents
0x00	4	0x91DA8106
0x04	4	0x439FC7CE
0x08	4	Victim PC identifier
0x0C	1	Generated based on the contents of a configuration file "789406d0"
0x0D	1	Type
0x0E	2	0x1400
0x10	1	Length
0x11	1	0x6
0x12	1	remark length
0x13		remark
-	1	0x3
-	1	password length
-	-	password
-	1	0x5
-	1	otherinfo length
-	-	otherinfo
-	-	System information

Table B-2：Format of data sent for second communication onwards

Offset	Length (byte)	Contents
0x00	4	0x91DA8106
0x04	4	0x439FC7CE
0x0D	1	Commands
0x10	2	command data length
0x12	2	Unused
0x14	2	Unused
0x16	2	Unused
0x18	Command data length	Command data

Table B-3：Format of data received

Appendix C WinDealer List of commands

Value	Parameter string*	Contents
0x06	content-length: 2	uninstall
0x09	content-length, filename, time	Delete files under %TEMP%
0xC	filename, flg	CreateProcess
0x1F	speed	Configure Sleep time
0x2D	filepath	Obtain contents of selected file

0x50	filename, md5	Delete selected file
0x51	filepos,filename, filelen, block, md5	Write on selected file
0x5A	datastate	Write on "C:\ProgramData\windows.inf"
0x5B	-	Perpetuation settings for registries
0x5C	list	Perpetuation after process check
0x5D	yes	Set a value to SType of {HKCU}\\Softwaware\Microsoft
0x5E	otherinfo	Write on "c25549fe"
0x60	headsign, 1, 2	Write on "789406d0"
0x61	reverseip	Write on "9c3b6294"
0x63	-	Obtain configuration
0x64	-	Read time
0x66	remoteip, remark, password	Write on configuration files
0x67	sessionid:	-
0x8F	Hkey, subkey, valuename, classesroot, currentuser, localmachine, users, currentconfig	Execute RegQueryValue
0xAA	pname	Screen capture
0xAB	-	Configuration on screen capture

0xAD	-	Configuration on screen capture

Table C：List of commands

*Parameter string: These strings are parsed from the received command and used as a command parameter

Appendix D List of commands of loaded modules

Value	Parameter string*	Contents
0x02	-	Related to screen capture
0x03	bootdir, filetype	Related to folder/files
0x05	filename, monitortype, begpos, block	Send files
0x07	-	Obtain drive information
0x0A	-	Configure for lnk files
0x0D	-	Execute commands 0xC0, 0xC5, 0xC3, 0xC1, 0xC2, 0xC4, 0xC6
0x12	freq, storetm, quality, type	Configure parameter
0x1E	srhdir, srhcont, srhnum, sessid	-
0x28	filename	Obtain file information
0x29	filefilter, settype, usbfilter, checkdirfilter	Configure parameter for monitoring
0x2A	monitortype, monitorvalue	Obtain files of monitoring results
0x2B	-	-

0x30	-	Write contents such as "c:\windows", "c:\program files" on "~BF24"
0x32	freq, storetm	Configure parameter
0x3E	file	Create jpeg file under %TEMP%
0x65	filename, fileoffset	Obtain contents from selected files and offsets
0x69	filename, delete, yes	Delete selected files
0x7A	cmdtype, command: ,reset, downfile, getmypath, dealmd5	Execute cmd.exe
0x7B	session, command, reset, downfile, exit, getmypath	Execute remote shell
0xC0	-	Write list of processes on "28e4-20a6acec"
0xC1	-	Write list of applications on "28e4-20a6acec"
0xC2	-	Write keyboard information on "28e4-20a6acec"
0xC3	-	Write SNS-related registry contents on "28e4-20a6acec"
0xC4	-	Write configuration of Skype, QQ, WeChat and wangwang on "28e4-20a6acec"
0xC5	-	Write MAC address etc. on "28e4-20a6acec"
0xC6	-	Write network configuration on "28e4-20a6acec"

Table D：List of commands

*Parameter string: These strings are parsed from the received command and used as a command parameter

Appendix E List of generated directories

ID	Path	String in malware
(none)	%TEMP%\\~FEFEFE	-
0x01	%TEMP%\\070a-cf37dcf5	-
0x02	%TEMP%\\d0c8-b9baa92f	audio
0x03	%TEMP%\\~B5D9	keylog
0x04	%TEMP%\\632c-0ef22957	-
0x05	%TEMP%\\8e98-fb8010fb	filelist
0x06	%TEMP%\\7a4a-90e18681	-
0x07	%TEMP%\\d4a5-30d3fff6	-
0x08	%TEMP%\\d4dc-3165f4cf	-
0x09	%TEMP%\\~CE14	monitortype
0x0A	%TEMP%\\~CE2E	-
0x0B	%TEMP%\\~B5BE	skypeaudio
0x0C	%TEMP%\\~B61A	skypeshoot
0x0E	%TEMP%\\5a7e-42ccdb67	-
0x0F	%TEMP%\\~BF24	browser
0x10	%TEMP%\\65ce-731bffbb	md5filter

0x11	%TEMP%\\~BF34	browsercookie
0x12	%TEMP%\\28e4-20a6acec	systeminfo
0x61	%TEMP%\\~FFFE	otherfile
0x62	%TEMP%\\FFFF	otherdata
0x63	%TEMP%\\63ae-a20cf808	-

Table E： List of directory

Appendix F SHA256 hash values of similar samples

EXE

- 1e9fc7f32bd5522dd0222932eb9f1d8bd0a2e132c7b46cfcc622ad97831e6128
- b9f526eea625eec1ddab25a0fc9bd847f37c9189750499c446471b7a52204d5a

DLL

- 0c365d9730a10f1a3680d24214682f79f88aa2a2a602d3d80ef4c1712210ab07
- 2eef273af0c768b514db6159d7772054d27a6fa8bc3d862df74de75741dbfb9c

Email

Author

増渕 維摩(Yuma Masubuchi)

Yuma has been engaged in malware analysis and coordination of cyber security incidents in JPCERT/CC Incident Response Group since November 2020.