

Volatility Plugin for Detecting Cobalt Strike Beacon - JPCERT/CC Eyes

By JPCERT/CC

Published: 2018-08-02 · Archived: 2026-04-05 15:02:22 UTC

- [Python](#)

JPCERT/CC has observed some Japanese organisations being affected by cyber attacks leveraging “Cobalt Strike” since around July 2017. It is a commercial product that simulates targeted attacks [1], often used for incident handling exercises, and likewise it is an easy-to-use tool for attackers. Reports from LAC [2] and FireEye [3] describe details on Cobalt Strike and actors who conduct attacks using this tool.

Cobalt Strike is delivered via a decoy MS Word document embedding a downloader. This will download a payload (Cobalt Strike Beacon), which will be executed within the memory. Since Cobalt Strike Beacon is not saved on the filesystem, whether a device is infected cannot be confirmed just by looking for the file itself. There is a need to look into memory dump or network device logs.

This article is to introduce a tool that we developed to detect Cobalt Strike Beacon from the memory. It is available on GitHub - Feel free to try from the following webpage:

JPCERTCC/aa-tools · GitHub

<https://github.com/JPCERTCC/aa-tools/blob/master/cobaltstrikescan.py>

Tool details

This tool works as a *plugin* for The Volatility Framework (hereafter “Volatility”), a memory forensic tool. Here are the functions of `cobaltstrikescan.py`:

- `cobaltstrikescan`: Detect Cobalt Strike Beacon from memory image
- `cobaltstrikeconfig`: Detect Cobalt Strike Beacon from memory image and extract configuration

To run the tool, save `cobaltstrikescan.py` in “contrib/plugins/malware” folder in Volatility, and execute the following command:

```
$python vol.py [cobaltstrikescan|cobaltstrikeconfig] -f <memory.image> --profile=<profile>
```

Figure 1 shows an example output of `cobaltstrikescan`. You can see the detected process name (Name) and process ID (PID) indicating where the malware is injected to.

Figure 1: Execution results of `cobaltstrikescan`

- Takuya Endo

(Translated by Yukako Uchida)

Reference

[1] Strategic Cyber LLC:COBALT STRIKE ADVANCED THREAT TACTICS FOR PANETRATION TESTERS

<https://www.cobaltstrike.com/>

[2] LAC: New attacks by APT actors menuPass (APT10) observed (Japanese)

https://www.lac.co.jp/lacwatch/people/20180521_001638.html

[3] FireEye: Privileges and Credentials: Phished at the Request of Counsel

<https://www.fireeye.com/blog/threat-research/2017/06/phished-at-the-request-of-counsel.html>

[4] Cybereason: Operation Cobalt Kitty: A large-scale APT in Asia carried out by the OceanLotus Group

<https://www.cybereason.com/blog/operation-cobalt-kitty-apt>

Appendix A

Table A: Configuration format

Offset	Length	Description
0x00	2	index (Refer to Table B)
0x02	2	Data length 1 = 2 byte, 2 = 4 byte, 3 = as specified in 0x04
0x04	2	Data length
0x06	As specified in 0x04	Data

Table B: Configuration

Offset	Description	Remarks
0x01	BeaconType	0=HTTP, 1=Hybrid HTTP and DNS, 8=HTTPS
0x02	Port number	
0x03	Polling time	

Offset	Description	Remarks
0x04	Unknown	
0x05	Jitter	Ratio of jitter in polling time (0-99%)
0x06	Maxdns	Maximum length of host name when using DNS (0-255)
0x07	Unknown	
0x08	Destination host	
0x09	User agent	
0x0a	Path when communicating HTTP_Header2	
0x0b	Unknown	
0x0c	HTTP_Header1	
0x0d	HTTP_Header2	
0x0e	Injection process	
0x0f	Pipe name	
0x10	Year	Stops operating after the specified date by Year, Month, Day
0x11	Month	
0x12	Day	
0x13	DNS_idle	
0x14	DNS_Sleep	
0x1a	HTTP_Method1	
0x1b	HTTP_Method2	
0x1c	Unknown	
0x1d	Process to inject arbitrary shellcode (32bit)	
0x1e	Process to inject arbitrary shellcode (64bit)	

Offset	Description	Remarks
0x1f	Unknown	
0x20	Proxy server name	
0x21	Proxy user name	
0x22	Proxy password	
0x23	AccessType	1 = Do not use proxy server 2 = Use IE configuration in the registry 4 = Connect via proxy server
0x24	create_remote_thread	Flag whether to allow creating threads in other processes
0x25	Not in use	



[JPCERT/CC](#)

Please use the below contact form for any inquiries about the article.

Related articles

```

*key = 0x427c4867;
*key[1] = 0x015832c2;
*key[2] = 0x66472834;
*key[12] = 0x89d97969;
Dv[0] = 0x3250421;
Dv[1] = 0x44805488;
Dv[2] = 0x3b788529;
Dv[3] = 0x00788082;
** m_ret_argOffset0x350(a1 + 3);
if ( !(&CrypAcquireContext)(a1, 0, "Microsoft Enhanced RSA and AES Cryptographic Provider", 0x18, 0xf0000000) )
    return 0;
v1 = m_ret_argOffset0x350(a1 + 3);
handlHashob = a1 + 1;
if ( !(&CrypCreateHash)(a1, 0x8004, 0, 0, a1 + 1) )
{
    LABEL_0:
    if ( !*a1 )
        return 0;
    v6 = m_ret_argOffset0x350(a1 + 3);
    (&CrypReleaseContext)(a1, 0);
    return 0;
}
if ( !CrypHashData("handlHashob", key, 16u, 0) )
{
    v4 = m_ret_argOffset0x350(a1 + 3);
    v5 = a1 + 1;
    !(&CrypDeriveKey)(a1, 0x600f, "handlHashob", 0x80000, a1 + 2) // CALD_AES_128
}
if ( "handlHashob" )
{
    v5 = m_ret_argOffset0x350(a1 + 3);
    (&CrypDestroyHash)("handlHashob");
}
goto LABEL_0;
v8 = m_ret_argOffset0x350(a1 + 3);
(v10-&CrypSetKeyParam)(v8, 3, 0xnum1, 0); // XP_F402D86 + P6C54577
v11 = m_ret_argOffset0x350(a1 + 3);
(v11-&CrypSetKeyParam)(v8, 1, Dv, 0); // Dv = parameter
v12 = m_ret_argOffset0x350(a1 + 3);
(v12-&CrypSetKeyParam)(v8, 4, 0xnum2, 0); // XP_F0D0E + CBC
return v8;
    
```

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

```

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](#)

```

movx eax, cs:num7
movd xmm1, eax
cvtdq2pd xmm1, xmm1
movx eax, cs:num3
movd xmm0, eax
cvtdq2pd xmm0, xmm0
addsd xmm0, xmm0
subsd xmm1, xmm0
mulsd xmm1, xmm2
movsd [rbp+1410h+ph0prev], xmm1
call ret2
movx r9d, al
call ret0
movx ecx, al
imul r9d, ecx
call ret7
movx eax, al
add eax, r9d
movx ecx, cs:num9
add ecx, ecx
movx ecx, cs:num8
xor edx, edx
div ecx
movx ecx, cs:num1
cmp eax, ecx
jz short loc_7FF85B1895C0
call ret1
movx edx, al
movx eax, cs:num0
imul edx, eax
lea r8d, [rdx+rdx*2]
add r8d, r8d
call ret9
movx ecx, al
sub r8d, ecx
call ret6
movx ecx, al
add r8d, ecx
movx ecx, cs:num3
add ecx, r8d
    
```

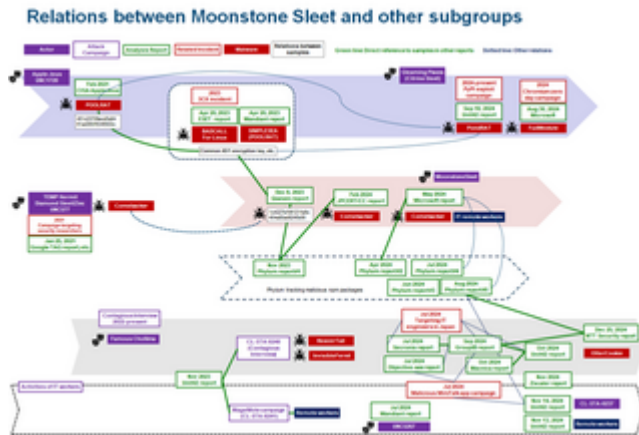
[Malware Identified in Attacks Exploiting Ivanti Connect Secure Vulnerabilities](#)

```

__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;
}
    
```

[DslugdRAT 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.jpccert.or.jp/en/2018/08/volatility-plugin-for-detecting-cobalt-strike-beacon.html>