

# SPAWNCHIMERA Malware: The Chimera Spawning from Ivanti Connect Secure Vulnerability - JPCERT/CC Eyes

By 増淵 維摩(Yuma Masubuchi)

Published: 2025-02-19 · Archived: 2026-04-29 02:11:26 UTC

February 20, 2025

In January 2025, Ivanti published an advisory[1] regarding the vulnerability CVE-2025-0282 in Ivanti Connect Secure. JPCERT/CC has confirmed multiple cases of this vulnerability being exploited in Japan since late December 2024, prior to the disclosure of the vulnerability, and published a security alert[2]. **This vulnerability has already been used by multiple attack groups.**

Among these cases, JPCERT/CC has confirmed that SPAWN malware family[3][4], which infects after exploiting the vulnerability, according to a report by Google, had been updated. This article explains the updated malware family (hereafter referred to as “SPAWNCHIMERA”).

## Overview of SPAWNCHIMERA’s behavior

Figure 1 shows an overview of SPAWNCHIMERA’s behavior. It is malware with the functions of SPAWNANT, SPAWNMOLE, and SPAWNSNAIL all updated and integrated. Therefore, there is no significant difference in the way malware is installed or injected into other processes compared to SPAWN family reported by Google[4]. On the other hand, as shown in Figure 1, SPAWNCHIMERA can be injected into various processes and run in each of them. The major changes are as follows.

- Change in inter-process communication
- Function to fix vulnerability CVE-2025-0282
- New decode functions added
- Deleted debug message

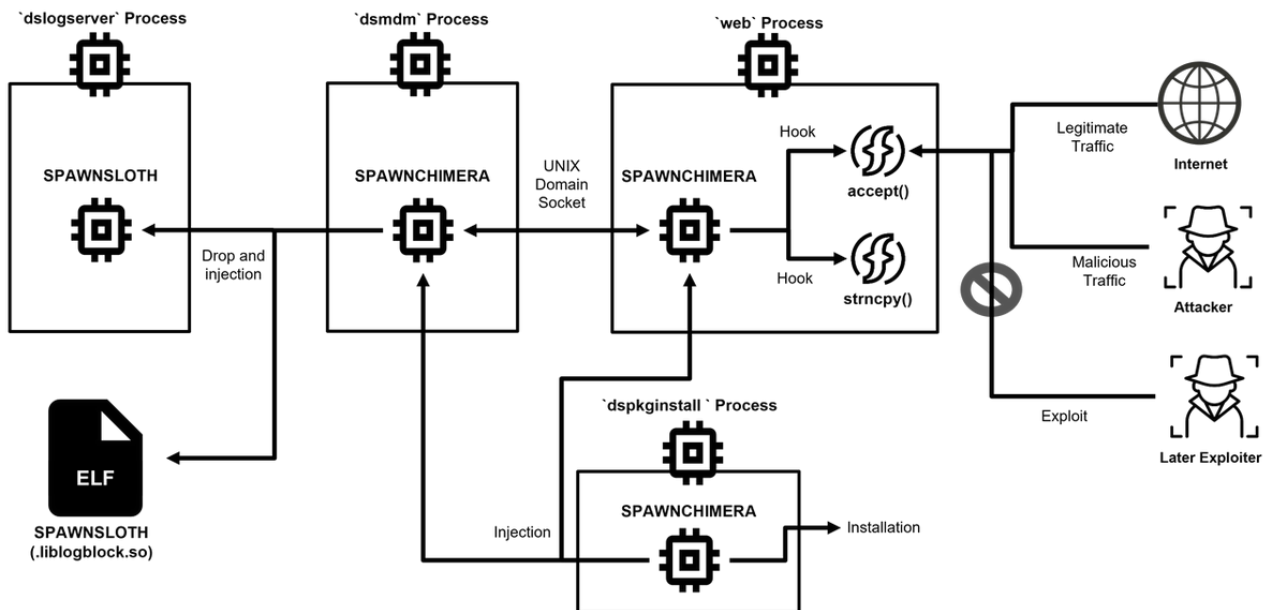


Figure 1: Flow of SPAWNCHIMERA's behavior.

### Inter-process communication through UNIX domain sockets

In the previous SPAWN family, the malicious traffic received by SPAWNMOLE was sent to port **8300** on **127.0.0.1**, and SPAWNSNAIL processed it. With the abovementioned update, this inter-process communication method was altered to use UNIX domain socket. It is created in the below path, and malicious traffic is sent and received between SPAWNCHIMERA injected into the **web** process and that injected into the **dsmdm** process. This change made it more difficult to detect the malware, as **netstat** command results from the Integrity Checker Tool (ICT) may not be displayed.

```
/home/runtime/tmp/.logsrv
```

### Function to fix the vulnerability CVE-2025-0282

SPAWNCHIMERA has a new function to fix the CVE-2025-0282 vulnerability. CVE-2025-0282 is a buffer overflow vulnerability[5] caused by the **strncpy function**, and the malware dynamically fixes it by hooking the **strncpy function** and limiting the copy size to **256**. Figure 2 shows the replaced strncpy function. SPAWNCHIMERA converts its process name to hexadecimal and verifies the added value. The fix is triggered when the process name is "**web**". The fix is programmed to be disabled when the first byte of the source copied to the strncpy function matches **0x04050203**. Due to this function, if another attacker uses this vulnerability to attempt penetration or executes a PoC[6] for scanning purposes, the attack may not succeed.

```

n256 = size;
progrname = *_progrname;
if ( !(_BYTE)progrname )
    goto LABEL_5;
_progrname_1 = _progrname + 1;
n0x13E = 0;
do
{
    ++_progrname_1;
    n0x13E += progrname;
    progrname = *(_progrname_1 - 1);
}
while ( (_BYTE)progrname );
if ( n0x13E == 0x13E && (dest & 0xFF000000) == 0xFF000000 && size > 256 )//
    // "web" = hex(0x77+0x65+0x62) = 0x13E
{
    src_1 = *src;
    if ( *src == 3 && *(_WORD *)(src + 1) == 0x502 )// pass if the src is 0x04050203
    {
        if ( src[3] != 4 )
            n256 = 256;
    }
    else
    {
        n256 = 256;
    }
}
else
{
LABEL_5:
    if ( !size )
        return dest;
    src_1 = *src;
}
i = 0;
while ( src_1 ) // fixed strncpy
{
    *(_BYTE *)(dest + i++) = src_1;
    if ( n256 <= i ) // dest will return if i is longer than 256
        return dest;
    src_1 = src[i];
}
if ( n256 > i )
    memset((void *)(dest + i), 0, n256 - i);
return dest;
}

```

Figure 2: The strncpy function replaced through hook

### New decode functions added

In the previous samples, the private key for SSH server functionality was hardcoded in plaintext within the samples and exported to **/tmp/.dskey**. On the other hand, in SPAWNCHIMERA, it is now encoded and hardcoded within the sample. The key is used after being decoded with an XOR-based decode function. Since it is not exported as a file, traces are less likely to be left. The decoded private key is shown below.

```

-----BEGIN OPENSSH PRIVATE KEY-----
b3B1bnNzaC1rZXktdjEAAAAABG5vbmUAAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAAtzc2gtZW
QyNTUxOQAAACB5yHbNy5qrd638t2dCLQ08TJb3D8m0+vifkGmBRho6+QAAAJB08wxcdPMM

```

```
XAAAAAtzc2gtZWQyNTUxOQAAACB5yHbNy5qrd638t2dCLQ08TJb3D8m0+vifkGmBRho6+Q  
AAAEbqjrwB7thqk5LnigfsE8EqLKrmWNhy82k5GTv8BBVlDXnIds3LmqT3rfy3Z0ItDTxM  
lvcPybT6+J+QaYFGGjr5AAAACWthbGLAa2FsaQECaWQ=  
-----END OPENSSH PRIVATE KEY-----
```

Additionally, while the previous sample identified malicious traffic in replaced **accept function**, by matching a part of the received buffer with a hard-coded value, SPAWNCHIMERA has a new decode function and determines whether the traffic is malicious based on its calculation result. The decode function is shown in Figure 3.

```
buf1 = buf;  
decodedBuf = -1;  
do  
{  
    decodedBuf ^= *buf1;  
    n8 = 8;  
    do  
    {  
        decodedBuf = (decodedBuf >> 1) ^ -(decodedBuf & 1) & 0xEDB88320;  
        --n8;  
    }  
    while ( n8 );  
    ++buf1;  
}  
while ( &buf[i] != buf1 );  
return decodedBuf;
```

Figure 3: Decode function used to identify malicious traffic

### Deleted debug message

While there are only minor differences in functionality between the previous SPAWNSLOTH and that dropped by SPAWNCHIMERA, some functions related to debug messages were deleted from the entire sample, possibly with the aim of complicating analysis and preventing hunting. This modification is also seen in the main sample of SPAWNCHIMERA. Figure 4 shows an example of the deleted functions.

```
int __cdecl mal_is_elffile(FILE *stream, unsigned __int8 *ptr)
{
    int *v2; // eax
    char *v3; // eax
    int v5; // eax

    if ( fread(ptr, 0x34u, 1u, stream) == 1 )
    {
        if ( !memcmp(ptr, &s2__4, 4u) )
        {
            v5 = ptr[4];
            if ( v5 == 1 )
            {
                return 0;
            }
            else
            {
                if ( v5 == 2 )
                {
                    msg("64-bit target process isn't supported by 32-bit process.");
                }
                else
                {
                    msg("Invalid ELF class: 0x%x", ptr[4]);
                    return -9;
                }
            }
        }
        else
        {
            msg("Invalid ELF header: 0x%02x,0x%02x,0x%02x,0x%02x", *ptr, ptr[1], ptr[2],
                return -10;
            }
        }
    }
    else
    {
        v2 = __errno_location();
        v3 = strerror(*v2);
        msg("failed to read ELF header. (error: %s)", v3);
        return -10;
    }
}

1 int __cdecl mal_is_elffile(FILE *stream, void *ptr)
2 {
3     if ( fread(ptr, 0x34u, 1u, stream) != 1 )
4         return -10;
5     if ( memcmp(ptr, &elfheader, 4u) )
6         return -10;
7     if ( *((_BYTE *)ptr + 4) == 1 )
8         return 0;
9     return -9;
10 }
```

Figure 4: Deleted debug message (left: previous version, right: current version)

## In closing

SPAWNCHIMERA has evolved into more sophisticated malware by changing various functions of SPAWN family in a way that leaves less traces, and SPAWN family is expected to remain in use. We hope that the information in this article will help your malware analysis. The hash values and file paths of the confirmed malware are listed in the Appendix.

Yuma Masubuchi

(Translated by Takumi Nakano)

## References

[1] Ivanti

Security Advisory Ivanti Connect Secure, Policy Secure & ZTA Gateways (CVE-2025-0282, CVE-2025-0283)

[https://forums.ivanti.com/s/article/Security-Advisory-Ivanti-Connect-Secure-Policy-Secure-ZTA-Gateways-CVE-2025-0282-CVE-2025-0283?language=en\\_US](https://forums.ivanti.com/s/article/Security-Advisory-Ivanti-Connect-Secure-Policy-Secure-ZTA-Gateways-CVE-2025-0282-CVE-2025-0283?language=en_US)

[2] JPCERT/CC

Ivanti Connect Secureなどにおける脆弱性 (CVE-2025-0282) に関する注意喚起

<https://www.jpCERT.or.jp/at/2025/at250001.html>

[3] Google

Ivanti Connect Secure VPN Targeted in New Zero-Day Exploitation

<https://cloud.google.com/blog/topics/threat-intelligence/ivanti-connect-secure-vpn-zero-day?hl=en>

[4] Google

Cutting Edge, Part 4: Ivanti Connect Secure VPN Post-Exploitation Lateral Movement Case Studies

<https://cloud.google.com/blog/topics/threat-intelligence/ivanti-post-exploitation-lateral-movement?hl=en>

[5] watchTowr Labs

Do Secure-By-Design Pledges Come With Stickers? - Ivanti Connect Secure RCE (CVE-2025-0282)

<https://labs.watchtowr.com/do-secure-by-design-pledges-come-with-stickers-ivanti-connect-secure-rce-cve-2025-0282/>

[6] Stephen Fewer

CVE-2025-0282.rb

<https://github.com/sfewer-r7/CVE-2025-0282/blob/main/CVE-2025-0282.rb>

### Appendix A: Hash values of the malware

- SPAWNCHIMERA 94b1087af3120ae22cea734d9eea88ede4ad5abe4bdeab2cc890e893c09be955
- SPAWNSLOTH 9bdf41a178e09f65bf1981c86324cd40cb27054bf34228efdcfee880f8014baf

### Appendix B: File paths of the malware confirmed

- SPAWNCHIMERA /lib/libsupgrade.so
- SPAWNSLOTH /tmp/.liblogblock.so



[増淵 維摩\(Yuma Masubuchi\)](#)

Yuma has been engaged in malware analysis in JPCERT/CC Cyber Security Coordination Group since 2020.

### Related articles

```

*key = 0x217c7400;
*key[4] = 0x21593322;
*key[8] = 0x04472834;
*key[12] = 0x00007909;
iv[0] = 0x12474421;
iv[4] = 0x48014408;
iv[8] = 0x00798029;
iv[12] = 0x99388807;
v2 = m_ret_arg1offft0x350(a1 + 3);
if ( !CrypAcquireContext(&a1, 0, "Microsoft Enhanced RSA and AES Cryptographic Provider", 0x10, 0xffffffff) )
return 0;
v3 = m_ret_arg1offft0x350(a1 + 3);
HandleHashobj = a1 + 1;
if ( !CrypFreeContext(&a1, 0x0004, 0, 0, a1 + 1) )
{
LABEL_0:
if ( *a1 )
return 0;
v6 = m_ret_arg1offft0x350(a1 + 3);
CrypReleaseContext(&a1, 0);
return 0;
}
if ( !CrypHashData(HandleHashobj, key, 16u, 0) )
{
v8 = m_ret_arg1offft0x350(a1 + 3);
v9 = a1 + 2;
!CrypDeriveKey(&a1, 0x0000, HandleHashobj, 0x000000, a1 + 2) // CACB_AES_128
{
if ( HandleHashobj )
{
v5 = m_ret_arg1offft0x350(a1 + 3);
v5->CrypDestroyHash(HandleHashobj);
}
goto LABEL_0;
}
v10 = m_ret_arg1offft0x350(a1 + 3);
v10->CrypSetKeyParam(&v9, 2, 0x0000, 0); // SP_PADDING = PADDING
v11 = m_ret_arg1offft0x350(a1 + 3);
v11->CrypSetKeyParam(&v8, 1, iv, 0); // iv = parameter
v12 = m_ret_arg1offft0x350(a1 + 3);
v12->CrypSetKeyParam(&v5, 4, 0x0000, 0); // SP_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 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 GT9UcalhAkpM4QAG
000080 52 6e 36 4e 77 36 52 48 6e 56 35 54 2f 69 48 4a Rn6Nw6RHhVST/1HJ
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 +IzYpXnU7pMs15d
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----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCB1QKBgQCNS381HP2V3JD4GT9UcalhAkpM4QAGRn6Nw6
RHhVST/1HJ+zHLH82q7Xkmo+U+IzYpXnU7pMs15dq+cRxMoTlMhNoq2UTWK9o9RodcZtZXsk
bM7TzK7UZjyapTIJfcq6BwMdsMx6gH4Os1B/Swnc3wQxUbOaqEokKorZwmhU3wIDAQAB
-----END PUBLIC KEY-----
```

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

```

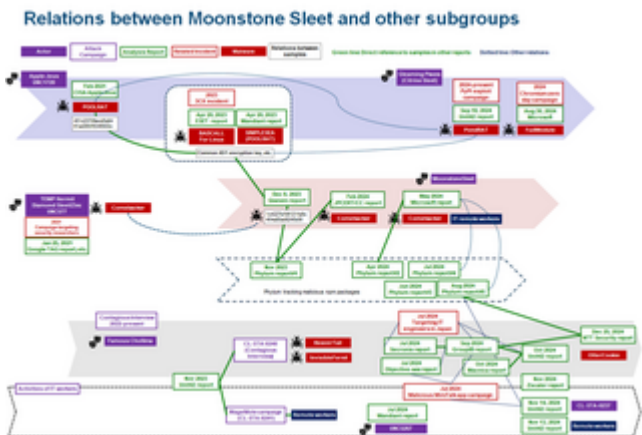
* 0F 04 00 0C 0A 04 00 movsx eax, cs:num7
* 06 0F 0E C8          movd  xmm1, eax
* 73 0F 16 C9          cvtdq2pd xmm1, xmm1
* 0F 0E 05 0C 0A 04 00 movsx eax, cs:num3
* 06 0F 0E C8          movd  xmm0, eax
* 73 0F 16 C9          cvtdq2pd xmm0, xmm0
* 72 0F 38 C8          addsd  xmm0, xmm0
* 72 0F 5C C8          subsd  xmm0, xmm0
* 72 0F 5A CA          mulsd  xmm1, xmm2
* 72 0F 11 40 00      movsd  [rbp+1410h+qPprev], xmm1
* 18 05 C8 FF FF      call  ret2
* 44 0F 08 C8          movsx  r9d, al
* 18 0C C8 FF FF      call  ret0
* 0F 05 C8          movsx  ecx, al
* 44 0F 40 C9          imul  r9d, ecx
* 18 00 C8 FF FF      call  ret7
* 0F 0E C8          movsx  eax, al
* 41 03 C1          add   eax, r9d
* 0F 0E 00 0F 0A 04 00 movsx  ecx, cs:num9
* 03 C1          add   eax, ecx
* 0F 0E 00 05 0A 04 00 movsx  ecx, cs:num8
* 33 02          xchg  edx, ebx
* 77 F1          div  ecx
* 0F 0E 00 07 0A 04 00 movsx  ecx, cs:num1
* 3C C1          cmp  eax, ecx
* 74 30          jz   short loc_7FF85B1895C0
* 18 06 C8 FF FF      call  ret3
* 0F 0E 00          movsx  edx, al
* 0F 0E 00 0C 0A 04 00 movsx  eax, cs:num0
* 0F 05 00          imul  edx, eax
* 44 00 04 52          lea  r8d, [rdx+edx*2]
* 45 03 C8          add  r8d, r8d
* 18 00 C8 FF FF      call  ret9
* 0F 0E C8          movsx  ecx, al
* 44 2B C1          sub  r8d, ecx
* 18 72 C8 FF FF      call  ret6
* 0F 0E C8          movsx  ecx, al
* 44 03 C1          add  r8d, ecx
* 0F 0E 00 4E 0A 04 00 movsx  ecx, cs:num3
* 41 03 C8          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;
}
```

[DslgdRAT 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/2025/02/spawnchimera.html>