

# GobRAT malware written in Go language targeting Linux routers - JPCERT/CC Eyes

By 増淵 維摩(Yuma Masubuchi)

Published: 2023-05-28 · Archived: 2026-04-05 16:11:08 UTC

- [Tool](#)

JPCERT/CC has confirmed attacks that infected routers in Japan with malware around February 2023. This blog article explains the details of the attack confirmed by JPCERT/CC and GobRAT malware, which was used in the attack.

## Attack flow up to malware execution

Initially, the attacker targets a router whose WEBUI is open to the public, executes scripts possibly by using vulnerabilities, and finally infects the GobRAT. Figure 1 shows the flow of the attack until GobRAT infects the router.

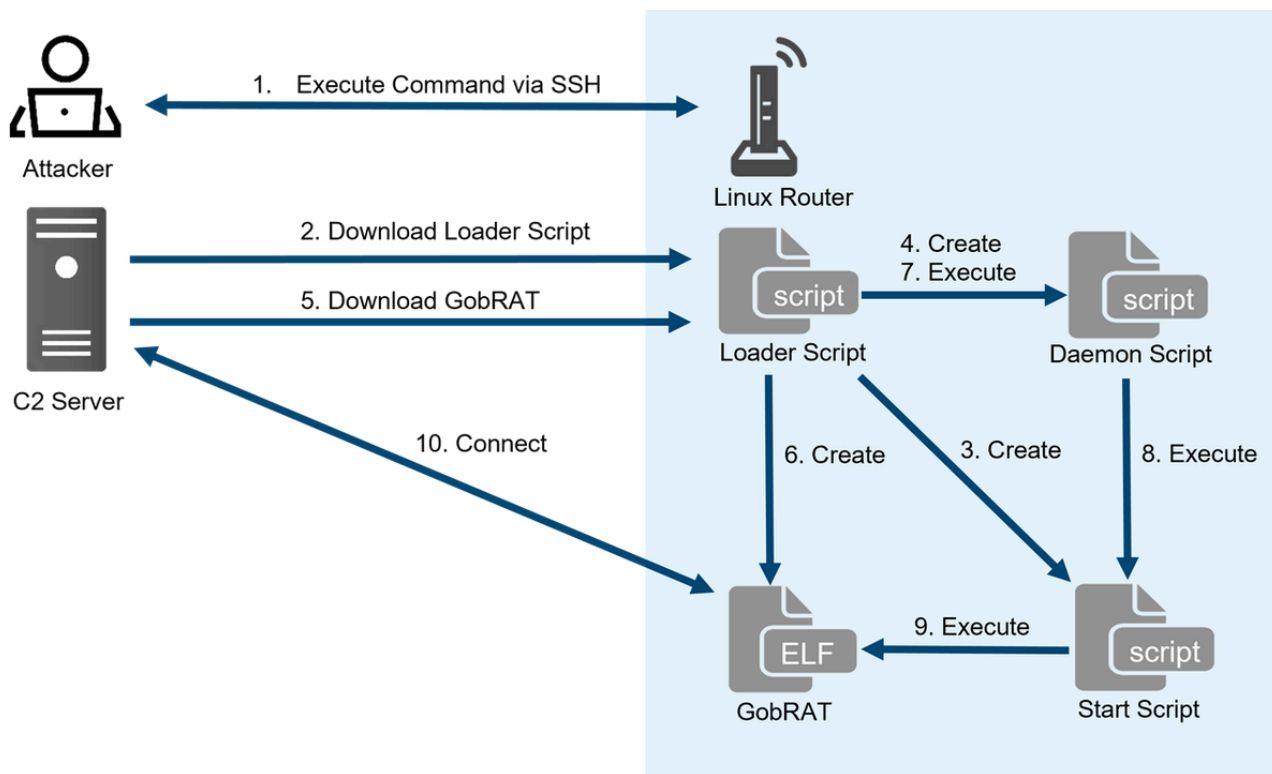


Figure 1: Attack Flow

**Loader Script** works as a loader, containing functions such as generating various scripts and downloading GobRAT. The SSH public key, which is assumed to be used for the backdoor, is hard-coded in the script. In

addition, since **Loader Script** uses crontab to register the file path of **Start Script** for persistence, GobRAT does not have such function. The functions of **Loader Script** are as follows:

- Disable Firewall function
- Download GobRAT for the target machine's architecture
- Create **Start Script** and make it persistent
- Create and run **Daemon Script**.
- Register a SSH public key in /root/.ssh/authorized\_keys

Figure 2 is the code of **Start Script** that executes GobRAT. The script is unique in that it writes the startup time to a file named **restart.log**. In addition, this script executes GobRAT under the file name **apached** to make it look like a legitimate process.

```
#!/bin/sh
cd /tmp/env/.qnapd
file_name="/tmp/env/.qnapd/restart.log"
if (ps -ef || ps) | grep '/tmp/env/.qnapd/apached' | grep -v grep; then
  echo
else
  if type nohup; then
    nohup /tmp/env/.qnapd/apached -d >/dev/null &
  else
    /tmp/env/.qnapd/apached -d >/dev/null &
  fi
  echo `date` >> $file_name
fi
```

Figure 2: Start Script

Figure 3 is the code of **Daemon Script**. This script checks whether **Start Script** is running or not every 20 seconds, and if not, it starts the script. This code has been possibly prepared in case **Start Script** is terminated unexpectedly.

```
#!/bin/sh
while true;do
  if ! pidof apached; then
    /tmp/env/.qnapd/sshd.sh
  fi
  sleep 20
done
```

Figure 3: Daemon Script

## GobRAT Overview

GobRAT is a RAT written in Go language and communicates with C2 server via TLS and executes various commands. It is packed with UPX version 4 series, and samples for various architectures such as ARM, MIPS, x86, and x86-64 have been confirmed. GobRAT performs the following checks at startup and keeps the information within the sample itself.

- IP address and MAC address of itself
- Uptime by uptime command
- Network communication status by /proc/net/dev

The following sections describes the GobRAT’s communication method, encryption method, and commands to be executed.

## Communication method

GobRAT uses TLS to send and receive data with its C2 server. Figure 4 shows an example of communication with the C2 server. The first 4 bytes indicate the size of the data, and the rest is gob[1] data. gob is a data serialization protocol available only in Go language. GobRAT uses gob for receiving commands and sending the results of command execution.

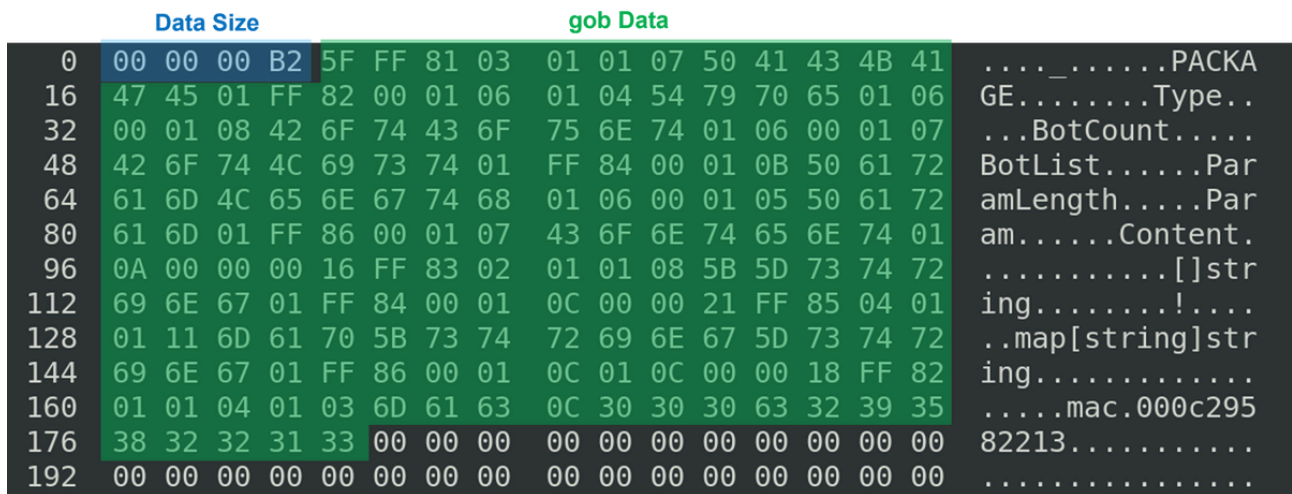


Figure 4: Example of communication content

GobRAT defines gob data as a PACKAGE structure in the sample as follows.

```

type PACKAGE struct {
    Type uint8           // CommandID
    BotCount uint16      // Parameter
    BotList []string     // Command Parameter
    ParamLength uint16  // Length of Param
    Param map[string]string // Command Parameter
}
    
```

```
Content []uint8 // Command Parameter, Command Execution Result, etc
}
```

The fields used are different depending on the type of command, and string arrays, maps, and binary data are supported so that various types of parameters can be passed. In addition, while binary data can be stored in Content of the PACKAGE structure, map data with string is converted to binary data by encoding it with the json.Marshal function. The PACKAGE structure is used in various ways depending on the command, such as storing the data in Content, or converting the defined structure to binary data in the same way and storing it in Content.

## Encryption Method

Strings such as C2 and Linux commands are encrypted and stored in the sample. Figure 5 shows the GobRAT's decryption function. AES128 CTR mode is used to decrypt strings, and the key and IV are hard-coded in the sample. The same key (**050CFE3706380723433807193E03FE2F**) and IV ("**12345678abcdefgh**") are used in all the confirmed samples. In addition, as shown in Figure 6, the codes that have probably been developed by the attacker, such as this decryption function, has a unique folder structure like **aaa.com/bbb/me~**.

```
__int64 __golang aaa_com_bbb_mecrypt_AesEncrypt(
    __int64 ENCDATA,
    signed __int64 ENCDATA_SIZE,
    __int64 ENCDATA_SIZE_1,
    int AESKEY,
    __int64 KEYSIZE)
{
    __int64 v5; // r14
    __int64 KEY; // rax
    __int64 v7; // rcx
    _16_uint8 *IV; // rax
    RTYPE **AES_CTR; // [rsp+0h] [rbp-30h]
    __int64 Decrypted; // [rsp+18h] [rbp-18h]
    __int64 KEY_1; // [rsp+20h] [rbp-10h]
    void *retaddr; // [rsp+30h] [rbp+0h] BYREF

    if ( &retaddr <= *(v5 + 16) )
        JUMPOUT(0x608158LL);
    KEY = (crypto_aes_NewCipher)(AESKEY, KEYSIZE);
    if ( v7 )
        return 0LL;
    KEY_1 = KEY;
    IV = runtime_newobject(&RTYPE__16_uint8);
    qmemcpy(IV, "12345678abcdefgh", sizeof(_16_uint8));
    AES_CTR = crypto_cipher_NewCTR(KEY_1, KEYSIZE, IV, 0x10uLL);
    Decrypted = (runtime_makeslice>(&RTYPE_uint8, ENCDATA_SIZE, ENCDATA_SIZE);
    (AES_CTR[3])(KEYSIZE, Decrypted, ENCDATA_SIZE, ENCDATA_SIZE, ENCDATA);
    return Decrypted;
}
```

Figure 5: String decryption function

```
String
aaa.com/bbb/mecrypt.AesEncrypt
aaa.com/bbb/mecrypt.Unvisual
aaa.com/bbb/mecrypt/mecrypt.go
aaa.com/bbb/menet
aaa.com/bbb/menet.(*CONN).Close
aaa.com/bbb/menet.(*CONN).Read
aaa.com/bbb/menet.(*CONN).RemoteAddr
aaa.com/bbb/menet.(*CONN).Write
aaa.com/bbb/menet.GetLocalAddress
aaa.com/bbb/menet.GetMacAddress
aaa.com/bbb/menet.IPString2Uint32
aaa.com/bbb/menet.Receive
aaa.com/bbb/menet.Send
aaa.com/bbb/menet/menet.go
aaa.com/bbb/meutil
aaa.com/bbb/meutil.Daemon1
aaa.com/bbb/meutil.Daemon2
aaa.com/bbb/meutil.Debug
aaa.com/bbb/meutil.DebugError
aaa.com/bbb/meutil.NewDaemon
aaa.com/bbb/meutil.RegisterLogFile
aaa.com/bbb/meutil.SimpleCommand
aaa.com/bbb/meutil.SimpleCommand.func1
aaa.com/bbb/meutil.UniqueAppendString
aaa.com/bbb/meutil._debug
aaa.com/bbb/meutil.init
aaa.com/bbb/meutil/meutil.go
```

Figure 6: Characteristic folder structure

### Commands executed

GobRAT has 22 commands that are executed by the commands from the C2 server, and we have identified the following commands. Since the malware targets routers, you can see that most functions are related to communication, such as frpc, socks5, and reconfiguration of C2. See Appendix A for command details.

- Obtain machine Information
- Execute reverse shell
- Read/write files
- Configure new C2 and protocol
- Start socks5
- Execute file in /zone/frpc
- Attempt to login to sshd, Telnet, Redis, MySQL, PostgreSQL services running on another machine

### GobRAT Analysis Tools

Since GobRAT uses gob for communication, if you want to emulate its communication with C2 to check commands, you need to create a program using Go language. Our C2 emulation tool that supports GobRAT

analysis is available on GitHub. Please download it from the following webpage for your analysis.

### JPCERTCC/aa-tools/GobRAT-Analysis - GitHub

<https://github.com/JPCERTCC/aa-tools/tree/master/GobRAT-Analysis>

### In Closing

In recent years, different types of malware using Go language have been confirmed, and the GobRAT malware confirmed this time uses gob, which can only be handled by Go language, for communication. Please continuously beware of malware that infects routers, not limited to GobRAT, since they are difficult to detect. Please refer to Appendix B for C2 of the malware, Appendix C for the hash value of the script, and Appendix D for the hash value of the malware.

Yuma Masubuchi

Translated by Takumi Nakano

### Appendix A: Commands

TableA: GobRAT commands

Value	Contents
0x0	Update json data held in malware and acquire update results
0x1	Retrieve json data held in malware
0x3	Start reverse shell
0x4	End of reverse shell connection
0x6	Confirmation of reverse shell connection
0x7	Execute shell command for daemon
0x8	Execute shell command
0xD	Read/write specified file
0x10,0x11	Read/write specified file
0x16	Obtain various machine information such as df command
0x17	Set new communication channel for TCP
0x18	Execute SOCKS5 proxy with specified port and password
0x19	Execute SOCKS5 proxy on specified port
0x1a	New communication channel setting for UDP

Value	Contents
0x1b	Execute frpc after executing SOCKS5 proxy on port 5555
0x1f	Check for the existence of the specified file
0x25	Login attempts for SSH, telnet, redis, mysql, postgres
0x27	Configuration of specified goroutine
0x2a	Scan to HTTP/HTTPS service of specified IP
0x2D	Dictionary attack to HTTP/HTTPS service of specified IP
0x30	C2 configuration related
0x31	DDoS attacks on SYN, TCP, UDP, HTTP, ICMP

### Appendix B: C2

- <https://su.vealcat.com>
- <http://su.vealcat.com:58888>
- <https://ktlvz.dnsfailover.net>
- <http://ktlvz.dnsfailover.net:58888>
- [su.vealcat.com](https://su.vealcat.com)
- [ktlvz.dnsfailover.net](https://ktlvz.dnsfailover.net)
- [wpxsi.mefound.com](https://wpxsi.mefound.com)

### Appendix C: Hash values of the scripts

- 060acb2a5df6560acab9989d6f019fb311d88d5511f3eda0effcbd9fc6bd12bb
- feaef47defd8b4988e09c8b11967e20211b54e16e6df488780e2490d7c7fa02a
- 3e44c807a25a56f4068b5b8186eee5002eed6f26d665a8b791c472ad154585d1
- 60bcd645450e4c846238cf0e7226dc40c84c96eba99f6b2cffcd0ab4a391c8b3

### Appendix D: Hash values of the malware

- a8b914df166fd0c94106f004e8ca0ca80a36c6f2623f87a4e9afe7d86b5b2e3a
- aeed77896de38802b85a19bfc8f2a1d567538ddc1b045bcdb29cb9e05919b60
- 6748c22d76b8803e2deb3dad1e1fa7a8d8ff1e968eb340311fd82ea5d7277019
- e133e05d6941ef1c2e3281f1abb837c3e152fdeaffefde84ffe25338fe02c56d
- 43dc911a2e396791dc5a0f8996ae77ac527add02118adf66ac5c56291269527e
- af0292e4de92032ede613dc69373de7f5a182d9cbba1ed49f589ef484ad1ee3e
- 2c1566a2e03c63b67fbd80b4a67535e9ed969ea3e3013f0ba503cfa58e287e3
- 98c05ae70e69e3585fc026e67b356421f0b3d6ab45b45e8cc5eb35f16fef130c
- 300a92a67940cfafeed1cf1c0af25f4869598ae58e615ecc559434111ab717cd

- a363dea1efda1991d6c10cc637e3ab7d8e4af4bd2d3938036f03633a2cb20e88
- 0c280f0b7c16c0d299e306d2c97b0bff3015352d2b3299cf485de189782a4e25
- f962b594a847f47473488a2b860094da45190738f2825d82afc308b2a250b5fb
- 4ceb27da700807be6aa3221022ef59ce6e9f1cda52838ae716746c1bbdee7c3d
- 3e1a03f1dd10c3e050b5f455f37e946c214762ed9516996418d34a246daed521
- 3bee59d74c24ef33351dc31ba697b99d41c8898685d143cd48bccdff707547c0
- c71ff7514c8b7c448a8c1982308aaffed94f435a65c9fdc8f0249a13095f665e

## References

[1] Gobs of data

<https://go.dev/blog/gob>



[増渕 維摩\(Yuma Masubuchi\)](#)

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

## Related articles

```
 *key = 0x327c080;
 *key[4] = 0x015813c2;
 *key[8] = 0x0d472834;
 *key[12] = 0x00007909;
 *key[16] = 0x1374421;
 *key[20] = 0x48805a68;
 *key[24] = 0x00798129;
 *key[28] = 0x00000027;
 v3 = m_ret_arg1ffset0x358(a1 + 3);
 if ( !((v3 << CryptAcquireContext)(a1, 0, "Microsoft Enhanced RSA and AES Cryptographic Provider", 0x18, 0xf0000000) )
 return 0;
 v3 = m_ret_arg1ffset0x358(a1 + 3);
 handlehashob = a1 + 3;
 if ( !((v3 << CryptCreateHash)(a1, 0x0004, 0, 0, a1 + 3) )
 {
 LABEL_0:
 if ( "a1"
 return 0;
 v6 = m_ret_arg1ffset0x358(a1 + 3);
 (v6 << CryptReleaseContext)(a1, 0);
 return 0;
 }
 if ( !CryptHashData("handlehashob", key, 16a, 0)
 { (v8 = m_ret_arg1ffset0x358(a1 + 3));
 v9 = a1 + 2;
 (v8 << CryptDeriveKey)(a1, 0x0004, "handlehashob", 0x000000, a1 + 2) } // CACB_AES_128
 {
 if ( "handlehashob" )
 {
 v8 = m_ret_arg1ffset0x358(a1 + 3);
 (v8 << CryptDestroyHash)(handlehashob);
 }
 goto LABEL_0;
 }
 v10 = m_ret_arg1ffset0x358(a1 + 3);
 (v10 << CryptSetKeyParam)(v9, 3, 0x0001, 0); // SP_PADDING = PKCS047
 v11 = m_ret_arg1ffset0x358(a1 + 3);
 (v11 << CryptSetKeyParam)(v9, 1, IV, 0); // IV = parameter
 v12 = m_ret_arg1ffset0x358(a1 + 3);
 (v12 << CryptSetKeyParam)(v9, 4, 0x0001, 0); // SP_MODE = CBC
 return v9;
 }
```

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

```

λ 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+cRxMoTLmhNoq2U
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-----
MIGFMA0GCSqGS1b3DQEBAAQUAA4GNADCB1QKBgQCNS381HP2V3JD4GT9UcaLhAkPmDQAGRN6Nw6
RHNvST/1HJ+zHLH82q7Xkmo+rU+IzYpXmU7pMs1Sdq+cRxMoTLmhNoq2UTWK9o9RodcZtZXsk
bM7TzK7UZjyapTIJfcq6BwMdsMx6gH4Os1B/Swnc3wXubOaqEokKorZumHU3wIDAQAAB
-----END PUBLIC KEY-----

```

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

```

* 73 0F 58 C9
* 86 0F 58 C9
* 73 0F 58 C9
* 72 0F 58 C8
* 72 0F 58 C8
* 72 0F 59 CA
* 72 0F 51 40 08
* 18 05 C1 FF FF
* 18 0C C1 FF FF
* 0F 0E C8
* 44 0F AF C9
* 18 00 C1 FF FF
* 0F 0E C8
* 41 03 C1
* 0F 05 00 3F 0A 04 00
* 03 C1
* 0F 0E 00 35 0A 04 00
* 33 02
* 77 F1
* 0F 0E 00 87 0A 04 00
* 10 C1
* 74 38
* 18 05 C1 FF FF
* 0F 0E D0
* 0F 0E 05 8C A0 04 00
* 0F AF D0
* 44 00 04 52
* 45 03 C9
* 18 00 C1 FF FF
* 0F 0E C8
* 44 28 C1
* 18 72 C1 FF FF
* 0F 0E C8
* 44 03 C1
* 0F 0E 00 42 0A 04 00
* 41 03 C8
movsx eax, cs:num7
movd xmm1, eax
cvtdq2pd xmm1, xmm1
movsx eax, cs:num3
movd xmm0, eax
cvtdq2pd xmm0, xmm0
addsd xmm0, xmm0
subsd xmm1, xmm0
mulsd xmm1, xmm2
movsd [rbp+1410h+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 ecx, ecx
movsx ecx, cs:num8
xor edx, edx
div ecx
movsx ecx, cs:num1
cmp eax, ecx
jz short loc_7FF85B1895C0
call ret1
movsx edx, al
movsx eax, cs:num0
imul edx, eax
lee r8d, [rdx+rdx*2]
add r8d, r8d
call ret9
movsx ecx, al
sub r8d, ecx
call ret6
movsx ecx, al
add r8d, ecx
movsx ecx, cs:num3
add ecx, r8d

```

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

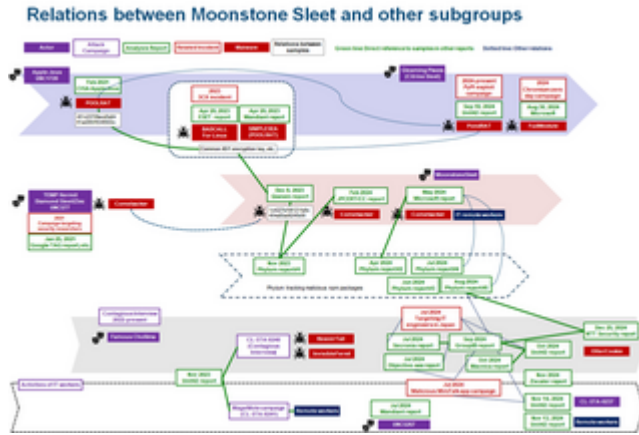
```

__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;
}

```

[DslodgRAT 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/2023/05/gobrat.html>