HNS Botnet Recent Activities

blog.netlab.360.com/hns-botnet-recent-activities-en/

RootKiter July 6, 2018

6 July 2018 / HNS
Author: Rootkiter, yegenshen

HNS is an IoT botnet (Hide and Seek) originally discovered by BitDefender in January this year. In that report, the researchers pointed out that HNS used CVE-2016-10401, and other vulnerabilities to propagate malicious code and stole user information. The HNS communicates through the P2P mechanism, which is the second IoT botnet that we know uses P2P communication after Hajime.

P2P-like botnets are hard to take down, and the HNS botnet has been continuously updated over the past few months, some major updates we see:

- Added exploits for AVTECH devices (webcam, webcam), CISCO Linksys router, JAWS/1.0 web server, Apache CouchDB, OrientDB; with the two devices mentioned in the original report, HNS currently supports 7 exploiting methods all together
- Hard-coded P2P node addresses have been increased to 171;
- In addition, we observed that the HNS botnet adds a cpuminer mining program, it is not functioning properly yet.
- In particular, with the added support of OrientDB and CouchDB database servers, HNS is no longer just an IoT botnet, but a cross-platform botnet now.

Malware Samples

We analyzed following samples:

```
c1816d141321276cd4621abcd280ee40    #hns x86
0770ff1a6e90eb5d083c16452e45abd5    #hns arm
30ebaaeb61a4ecae3ade7d1d4e5c7adb    #hns_miner
```

Network Scanning and Exploits

HNS botnet looks for potential victims by initiating a network scanning. In this scanning, HNS borrows code from mirai botnet, and shares the same characters.

The scanning target ports include fixed TCP port 80/8080/2480/5984/23 and other random ports.

```
HIBYTE(dport_pool[0]) = 0x50;           // tcp-80
dport_pool[1] = 0x901Fu;                // tcp-8080
dport_pool[4] = 0xB009u;                // tcp-2480
dport_pool[5] = 0x6017;                 // tcp-5984
HIBYTE(dport_pool[2]) = 0x17;           // tcp-23
HIBYTE(dport_pool[3]) = 0x17;           // tcp-23
HIBYTE(dport_pool[7]) = dword_20288;
LOBYTE(dport_pool[0]) = dword_20288;
LOBYTE(dport_pool[2]) = dword_20288;
LOBYTE(dport_pool[3]) = dword_20288;
LOBYTE(dport_pool[6]) = dword_20288;    // tcp-[rand_port]
HIBYTE(dport_pool[6]) = dword_20288;
LOBYTE(dport_pool[7]) = dword_20288;    // tcp-[rand_port]
random_tmp = maybe_hns_rand_what();
dport = *((_BYTE *)&v47 + 2 * (random_tmp & 7) - 20) | (*((_BYTE *)&v47 + 2 * (random_tmp & 7) - 19) << 8);
if ( !dport )                           // if dport==0; dport=rand_num
{
  for ( i = random_tmp >> 4; (i & 0x3FFF) > 0x2710; i >>= 1 )
    ;
  dport = ((unsigned __int8)i << 8) | ((i & 0x3FFF) >> 8);
}
```

These ports and their corresponding well known services are:

```
23    Telnet
80    HTTP Web Service
2480  OrientDB
5984  CouchDB
8080  HTTP Web Service
Random-Port   NA
```

HNS will try to implant itself on these ports, utilizing the following exploits. In these exploits, 1 and 2 are listed in bitdefender original report. The other 5 are newly integrated.

1. TPLink-Routers RCE
2. Netgear RCE
3. new: AVTECH RCE
4. new: CISCO Linksys Router RCE
5. new: JAW/1.0 RCE
6. new: OrientDB RCE
7. new: CouchDB RCE

171 Hard-coded P2P peer

HNS node contacts to other P2P peers with the following 3 methods, in which, 2 and 3 are listed in bitdefender original report.

1. From a hard-coded built-in list of 171 peer addresses
2. From the command-line args
3. From the other P2P peers

These 171 peers will interact during the check-in and following interaction process

The Check-in Process

When started with no command-line args, HNS node will send lots of UPD check-in packets. IP addresses of these packets are randomized, while some others are set based on the build-in list. Part of the list is shown as follows, with a full list can be seen in the IoC section.

```
.rodata:0805CF20 stru_805CF20    db 0DDh, 0D5h, 7Bh, 0FCh; peer_ip
.rodata:0805CF20                                         ; DATA XREF: sub_8056800:loc_
.rodata:0805CF20                                         ; sub_8056800:loc_805681B↑r .
.rodata:0805CF20                 db 2Fh, 7Ah             ; peer_port   221.213.123.252:12154
.rodata:0805CF20                 db 5Ah, 96h, 0CBh, 44h  ; peer_ip
.rodata:0805CF20                 db 3Eh, 0BFh            ; peer_port   90.150.203.68:16063
.rodata:0805CF20                 db 7Dh, 7Ah, 0Dh, 0D1h  ; peer_ip
.rodata:0805CF20                 db 0CCh, 0B0h           ; peer_port   125.122.13.209:52400
.rodata:0805CF20                 db 0DAh, 6Ch, 18h, 7Eh  ; peer_ip
.rodata:0805CF20                 db 36h, 34h             ; peer_port   218.108.24.126:13876
.rodata:0805CF20                 db 2Ah, 73h, 0F8h, 0Dh  ; peer_ip
.rodata:0805CF20                 db 1Eh, 0F5h            ; peer_port   42.115.248.13:7925
.rodata:0805CF20                 db 1Fh, 0A2h, 90h, 7Fh  ; peer_ip
.rodata:0805CF20                 db 3Bh, 0E9h            ; peer_port   31.162.144.127:15337
.rodata:0805CF20                 db 1Fh, 0A2h, 0BAh, 0B4h; peer_ip
```

The Interaction Process

HNS has the following characteristics when doing P2P interaction between its nodes:

- The check-in packet is a UDP packet with random length and content. So no significant feature here. The stands out is in the second packet from the upstream nodes, which has a length of 2 and the first byte is the uppercase letter ‘O’. In addition, the second byte of the return packet is actually a checksum value calculated from the request packet, so as can also be used to ID HNS communications. The detailed verification algorithm is shown as below.
- After joining into the P2P network, HNS needs to perform address synchronization constantly to ensure nodes connections. This synchronous operation has strong network characteristics. First, the downstream node sends a request packet with the length 1 with content "~" to the upstream node, and then the upstream node replies with a packet of length 8 with the initial letter "^".
- HNS nodes interact frequently and have lots of data exchange. The request packet length is 69 with initial letter "y", and the reply packet has a length of 261 with initial letter uppercase "Y".

The algorithm for service repond packets, based on python 2.7

When sending the following request packet:

```
data_pointer = "5b 02 d7 ff 52 02 61 e9 a5 7e 0b 07 c5 43
5b".replace("","").decode('hex')
```

The expected response packet from the upstream node would be:

```
ack="4f b6".replace(" ","").decode('hex')
```

Where chr(0x4f) is fixed and chr(0xb6) is calculated as follows:

```
def calc_crc(data_pointer):
    data_len_1 = len(data_pointer)-1
    tmp = 88
    for i in range(0,data_len_1):
        tmp += ord(data_pointer[i])
        tmp = tmp &0xff
    tmp2 = tmp^0x3d
    print hex(tmp2),hex(ord(data_pointer[data_len_1]))
    if(tmp2 == ord(data_pointer[data_len_1])):
        return (2*tmp2&0xfe)
    return -1;
```

Contact Us

If readers have new discoveries, feel free to contact us on twitter.

IoC

Malware Sample md5

```
c1816d141321276cd4621abcd280ee40    #hns x86
0770ff1a6e90eb5d083c16452e45abd5    #hns arm
30ebaaeb61a4ecae3ade7d1d4e5c7adb    #hns_miner
```

All 171 hard coded P2P peer address list is here.