Analyzing the Propagation of IoT Botnets from DNS Leakage

Anonymous Author(s)

ABSTRACT

Mirai and Hajime are two large botnets that came to prominence in the Fall of 2016, notably due to Mirai's launching of several large DDoS attacks. The propagation method of the two botnets is similar, drawing upon poor security measures in IoT devices. While reverse-engineering efforts have detailed the propagation logic, measuring the actual growth of each botnet remains difficult, with current methods based on honeypot traffic analysis.

This paper presents an alternative method for measuring the macro behavior of Mirai and Hajime that is based on failed infection attempts that leak into the DNS requests of the intended victim. This unexpected backscatter is due to the subtleties of a remote code execution vulnerability that Mirai and Hajime exploit for propagation. We analyze the growth and behavior of the two botnets using passive DNS collection at a root nameserver, and using active measurements taken by participating in the BitTorrent DHT, which Hajime uses to help disseminate its payloads. In contrast to honeypots, our methods provide a unique longitudinal and global view of the botnets' evolution.

1 INTRODUCTION

The proliferation of the Internet of Things (IoT) has resulted in a sudden increase in the number of network-connected devices in people's homes, spanning devices such as network-accessible webcams to network-controllable light bulbs and thermostats. IoT is intended to be a boon for users, promising unprecedented interoperability and control.

However, IoT is also proving to be a boon for attackers. Recent events have demonstrated that many IoT devices are subject to relatively straightforward attacks. Notably, one of the primary attack vectors of the Mirai botnet is to log into open telnet servers on IoT devices that still use default passwords. The combination of the ease of infection with the fast proliferation of IoT devices has led to large botnets and powerful attacks. In 2016, the Mirai botnet was used to launch some of the largest recorded DDoS attacks in history, including a 623 Gbps attack [2] against `krebsonsecurity.com`, and a 1.2 Tbps attack [6] against Dyn, a large DNS provider.

In the wake of the release of Mirai's source code, an interesting new botnet named *Hajime* was released late in 2016 [1, 7, 9]. Hajime exploits the same attack vectors as Mirai (see Section 2), but it has several key differences that indicate a higher level of sophistication than its predecessor. For instance, rather than make use of a centralized command and control (C&C) infrastructure, Hajime makes use of existing peer-to-peer infrastructure—the BitTorrent distributed hash table (DHT)—that its bots use to determine from whom to download new C&C commands.

Although there have been several honeypot-based studies of Hajime [1, 7], little is known about its novel network dynamics. In particular, these prior studies focused on short periods of time, and because of their nature as honeypots, are only able to see who attacked them, providing narrow insight into the overall attack patterns or rates of attack.

In this paper, we develop novel datasets that allow us to perform a *global* and *longitudinal* analysis of Hajime and, to a lesser extent, Mirai. Our key insight, at a high level, is as follows: Hajime makes use of a buffer overflow vulnerability in how some devices process NTP servers in particular configuration files: a vulnerable host will run the injected command, which includes a `wget` or `tftp` to the attacker's IP address. However, a *non-vulnerable* host will not overrun its buffer, and will instead treat the shell injection as a domain name. When it issues a DNS query for this domain name, the query itself will contain the attacker's IP address, and it will be an invalid domain name. As a result, the DNS query will be sent to a root DNS server—a "last-resort" server in DNS that is tasked with serving queries for unknown top-level domains (TLDs).

Our dataset comprises passive packet captures to one of the root DNS servers.[1] This root server is replicated across over 100 sites throughout the world, and has been collecting sampled query data for years, well before the emergence of Hajime. We complement and cross-validate our passive DNS dataset with active scans of the P2P distributed hash table (DHT) that Hajime uses to disseminate its attack payloads in a decentralized manner.

With these datasets, we are able to see millions of instances in which non-vulnerable hosts are targeted for attack, the IP address of the targets' local DNS resolver, and the IP address of the host attacking them. This offers a unique view into the spread of a botnet: rarely is it the case that one can track attempts against the *safe* targets.

We analyze this data to provide accurate, longitudinal analysis of the network dynamics of the Hajime botnet. Compared to prior studies [1, 7], we are able to identify an order of magnitude more Hajime bots. We use our datasets to investigate a wide swath of Hajime's characteristics, including: the number of active bots over time, their rate of infection, the geographic spread of their infection, and the bots' lifetimes. We also correlate Hajime's use of the peer-to-peer DHT with the botnet activity we observe through our passive DNS data.

In sum, we make the following contributions:

- We present a novel dataset that provides insight into the spread and activity of vulnerable hosts and how *non*-vulnerable hosts are targeted.
- We present the first longitudinal analysis of an advanced P2P-based IoT botnet, Hajime, and compare it to its predecessor, Mirai.
- We cross-validate our findings with active scans of the DHT that Hajime uses to disseminate its payloads.

[1] We do not mention which root server so as to preserve the anonymity of this paper submission.

The rest of this paper is organized as follows. In Section 2, we describe the Mirai and Hajime botnets and detail the vulnerability that we are able to track, and how we use it to obtain our passive DNS datasets. In Section 3, we describe our data collection methodology and our datasets. We analyze the size, location, and churn of Hajime bots in Section 4, and their attack patterns and targets in Section 5. In Section 6, we analyze Hajime's use of the BitTorrent DHT, and discuss attacks that relying on the DHT exposes Hajime to in Section 7. Finally, we conclude in Section 8.

2 BACKGROUND

In this section, we provide an overview of the Mirai and Hajime botnets. We also describe a remote code execution vulnerability that both botnets use to propagate, and discuss why failed exploitation attempts are observable at a root nameservers.

2.1 Mirai and Hajime Overview

The Mirai botnet was discovered in August 2016, and the Hajime botnet in early October 2016. Both botnets propagate by scanning the Internet for hosts with specific open ports. Initially, the scanning behavior was limited to Telnet; a bot would attempt to login to a Telnet service using a list of factory default username/password combinations, and once logged in, would issue the commands to transfer and execute the bot binary to the victim. Over time, both bots have added a propagation vector scanning for, and exploiting, the TR-064 service, which we describe next.

2.2 TR-064 Vulnerability

In addition to the Telnet propagation vector, Mirai and Hajime also propagate by exploiting a remote code execution vulnerability in some implementations of TR-064 [5]. TR-064 is a deprecated technical report published by the Broadband Forum that specifies the method for configuring a home router from within the LAN. The management protocol is based on SOAP over HTTP, and further mandates that requests to change a configuration value be password protected.

The enabling problem is that routers continue to run the TR-064 protocol, and moreover run the protocol without requiring authentication for configuration changes. The specific vulnerability is in the implementation of the request handler for the TR-064 time service, which allows for the configuration of the device's NTP client. The following HTTP SOAP message shows a request to set the domain name or IP address that the device should use as its primary NTP server.

```
POST /UD/act?1 HTTP/1.1
Host: VICTIM_HOST:VICTIM_PORT
User-Agent: FAKE_USER_AGENT
Content-Type: text/xml
Content-Length: BODY_LENGTH
SOAPAction: urn:dslform-org:service:Time:1#SetNTPServers

<?xml version="1.0"?>
<SOAP-ENV:Envlope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle=
   "http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
  <u:SetNTPServers xmlns:u="urn:dslforum-org:service:Time:1">
   <NewNTPServer1>SHELL_INJECTION</NewNTPServer1>
   <NewNTPServer2></NewNTPServer2>
   <NewNTPServer3></NewNTPServer3>
   <NewNTPServer4></NewNTPServer4>
   <NewNTPServer5></NewNTPServer5>
  </u:SetNTPServers>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

In an infection attempt, instead of specifying a legitimate domain name for an NTP server, a bot will specify a sequence of shell commands, shown above as `SHELL_INJECTION`. When run, the commands download a file, change the permissions of the file to be executable, and execute the file. Once the file is executed, the vulnerable device is infected and becomes part of the botnet.

If a device is vulnerable, then the value for `NewNTPServer` in the SOAP message is given to the shell for interpretation. The sequence of shell commands in the injection is either:

```
`cd /tmp;tftp -lX -rY -g HOST PORT;chmod 777 X;./X`
```

which makes a `tftp` connection to the attacker and downloads the remote file `Y` to a local file names `X`, or

```
`cd /tmp;wget http://HOST:PORT/X;chmod 777 X;./X`
```

which downloads the remote file `X` over HTTP. In both cases, `HOST` and `PORT` are the *attacker's* host and port, with `HOST` represented as a either a domain name or an IP address (`PORT` is optional).

As Mirai uses a centralized command and control (C&C), a Mirai shell injection typically uses the hostname of the C&C and omits the port number. Hajime, however, lacks a centralized C&C, and instead uses a peer-to-peer (P2P) architecture. As such, the Hajime shell injections use an IP address and an ephermal port number, where the IP address is that of the bot performing the injection.

2.3 DNS Backscatter

Consider what happens when the above injection attack is launched against a *non-vulnerable* device. Safe against an overflow attempt, the device will treat the `SHELL_INJECTION` as an NTP server hostname, and will thus perform a DNS lookup for its IP address. This involves sending a DNS query to its local resolver (or to a publicly available open resolver), which in turn recursively issues queries on behalf of the client. Of course, the `SHELL_INJECTION` lacks a valid top-level domain (TLD)—instead of `.com` or `.org`, it ends in `./X` for some number `X`. In such cases that a query has an invalid TLD, local resolvers must go to a *root DNS server* to get an authoritative answer.

In other words, whenever an attacker targets a non-vulnerable device, this will trigger a DNS query to a root DNS server for the payload of the attack—*which includes the attacker's IP address.* This resulting "backscatter" effect makes it possible to track unsuccessful injection attempts, provided querty data from a root name server is available.

2.4 Root Nameservers

The root of the DNS hierarchy comprises 13 lettered root name servers, A through M, and each DNS resolver bootstraps with information about the addresses of each root name server. The role of each root name server is to provide information about name servers for valid top level domains, e.g., for all names ending in .com or .edu. Name servers that ask the root for such information about valid names will cache what they learn, leaving most queries that the root receives to be for names that are not valid or not registered. Each letter typically comprises a set of geographically distributed replica servers reached via anycast, whereby internet routing chooses a particular server that will reply to the query.

2.5 Hajime's P2P Network

For peer-to-peer communication, Hajime makes use of BitTorrent's distributed hash table (DHT) overlay network. To start downloading a file, BitTorrent clients traditionally start by retrieving a list of peers from a tracker, a server that tracks peers hosting the file. Trackerless Torrents do not use a centralized tracker, instead storing this information in a DHT.

BitTorrent uses the Kademlia DHT protocol. Each node is identified by a unique 20-byte Node ID. To join the DHT, a new node must know at least one other node that is already a member. As a node discovers other nodes in the DHT, it stores their Node ID and IP address in a routing table. Values are mapped to a key in the same space as the Node IDs, usually some type of hash of the value data. A value is stored at the k closest nodes in the DHT to its key, where k is a system-wide replication parameter, for instance 20. The distance between two identifiers in Kademlia is calculated as a simple XOR of the two. A node storing a value periodically looks up the k closest nodes to the value's key and ensures all are storing the value.

To find a value in the DHT, a client performs a lookup of the value's key. Lookup is a recursive function that proceeds as follows. If a node is storing the value for that key, it returns the value. Otherwise it forwards the lookup request to the peer with the Node ID closest to the key that the client is aware of. In BitTorrent, the values being stored in the DHT are lists of peers. The key the peer lists are mapped to is the corresponding torrent's info_hash, the SHA1 hash of the torrent's metadata uniquely identifying the torrent. Thus to start downloading a torrent, a BitTorrent client will lookup that torrent's info_hash in the Kademlia DHT.

2.5.1 Kadnode. Hajime uses an additional feature from KadNode, a DNS resolver extension built on top of Kademlia. With KadNode, a node can announce that it owns some domain name, which other nodes can then resolve to the announcer's IP address. In the announce function, KadNode will store the announcer's IP address mapped to the domain identifier in the DHT, meaning the k nodes closest to the identifier will store the identifier to IP address mapping. A node that looks up the domain identifier will then get that IP address in return from one of those k nodes.

2.5.2 Hajime's Protocol. Hajime uses this DNS resolution feature to find peers hosting Hajime files. On being infected, Hajime bots join BitTorrent's Kademlia DHT. It is important to note that this DHT is the same ring used by BitTorrent, so it is not solely populated Hajime bots. Periodically, bots download a config file that directs the bot's behavior. To find a peer hosting the config file, the bot looks up an identifier through KadNode, then attempts to connect to each returned IP address until it successfully downloads the file. The identifier consists of daily time string concatenated with the SHA1 hash of the filename [7].

The DHT is used to find nodes that announce the config. files. The files themselves are downloaded using a direct unicast. For direct P2P communication, Hajime bots use uTorrent Transport Protocol (uTP), which implements congestion control over UDP. The initial messages of the uTP protocol exchange in Hajime exchange a key, and subsequent packets are encrypted with the RC4 stream cipher.

2.6 Related Work

The most immediately related prior work are the studies of the Hajime botnet performed by Kaspersky Labs [7], Radware [1], and Symantec [9] in the wake of the Hajime discovery. These prior studies involved a combination of honeypots and reverse engineering of the botnet payloads. By comparison, we perform a broader study of the network dynamics of Hajime, introducing a novel dataset that spans the entire lifetime of Hajime's use of the TR-064 vulnerability, and that includes far more bots. As we demonstrate in Section 4, for instance, we are able to identify an order of magnitude more bots than prior studies.

More broadly, there have been many studies of live botnets and the spread of malware. Staniford et al. [13] measured the quick spread of Internet worms; it turns out that Hajime differs considerably in that it does not appear to outpace human reaction, but rather to simply compromise as many devices as possible, and thus, as we will show, its infection rate is more modest. Stone-Gross et al. [14] took control of a botnet's centralized C&C infrastructure for a short period of time, and used it to measure the number of bots and the various sensitive information that they report back to their bot master. Conversely, we study Hajime longitudinally, over the course of months; so doing allows us to see the spread of a botnet over time. Moreover, we perform an extensive analysis of Hajime's *decentralized* C&C infrastructure, to the best of our knowledge this is a novel feature of this botnet.

3 DATASETS AND METHODOLOGY

We make use of two novel datasets to analyze Hajime's network dynamics.

3.1 Passive Root DNS Backscatter

Our passive DNS data set consists of sampled queries from one of the root DNS letters that comprises over 100 replicas in different geographic locations. The samples capture 20% of all queries to these replicas, corresponding to approximately 30K queries per second. All told, our traces include roughly one fifth of one thirteenth of the queries that reach root name servers in general. (The precise ratio may vary due to imbalance between letters.) Although not every query is captured, since attacking hosts attack many targets, the attacking bot IP address and port information can (and does) appear in our dataset.

Recall that the only hosts that will trigger a DNS query after an attack are the *non-vulnerable* devices. This provides a unique insight into the activity of a botnet, in that it provides us with data on attack *attempts*. However, it is not without its limitations. DNS query packets are not always large enough to contain the entire attack payload; as we discuss in Section 4, some of the attacker IP addresses in the queries are truncated (fortunately, this only represents 6.2% of the queries we analyze).

3.2 Active BitTorrent DHT Measurement

In order to analyze Hajime's novel use of an existing P2P infrastructure—as well as understand the error in the passive DNS measurements—we also perform small active measurements of the P2P network. To this end, we collect two datasets:

- **Seeder dataset:** Hajime bots download C&C configuration files from one another. To learn about which bots are serving it, they leverage BitTorrent's P2P distributed hash table. In essence, Hajime bots act like "seeders" by announcing their IP address and port, keyed by that day's C&C configuration file. We periodically (roughly every 5 minutes) download the list of these "seeders" over span of several days. To mitigate any potential geographic bias, we issue these queries from 7 distinct locations.
- **Leecher dataset:** After downloading the list of seeders, bots open a uTP connection to them to download that day's configuration file, similar to BitTorrent's "leechers." To understand the set of active bots, we announce our own hosts for several months' worth of Hajime configuration files, and record the IP addresses and ports of the bots who connect to us (we do not serve any files in return). We have collected these for several days, as well.

4 HAJIME BOTNET SIZE AND LOCATION

The first broad set of questions we ask pertain to the overall size and locations of the Hajime botnet over time. Although these broad statistics have been addressed in recent reports on Hajime [1, 7, 9], we offer a longitudinal view of Hajime's growth and change over time.

4.1 How large is the Hajime botnet?

We begin by investigating the overall size of Hajime over time. To this end, we analyze the distinct number of *full* IP addresses we obtain over time, ignoring an IP address if we are unable to determine whether or not it was truncated. Across all of the Hajime attack attempts we see, we obtain a total of 3,475,896 distinct IP address strings; 93.8% (3,278,579) of them are full IP addresses, and the other 6.2% (215,140) contain at least three full octets. Therefore, although our results are a lower bound, we believe the error to be low.

Figure 1 shows the cumulative number of distinct attacker IP addresses that we have observed. We make three key observations. *First*, the initial injection attempt occurs on December 28, 2016, nearly two full weeks before Hajime began to spread. Given the gap in time, it is possible that this was part of Hajime's initial testing phase for the `tftp` and `wget` vulnerabilities, but we are unable to confirm this. The IP address of this first attacker was located in Amsterdam.

Second, by the end of our window of data,[2] we have observed 3,260,756 distinct IP attacker addresses. This is one to two orders of magnitude greater than what has been previously reported [1, 7, 9].

Third, the overall growth rate in Figure 1 indicates a faster initial spread followed by a smooth, nearly linear, sustained rate of growth for months thereafter. At first glance, it may seem surprising that

[2]We continue to collect these data, and will update the results in future versions of this paper.

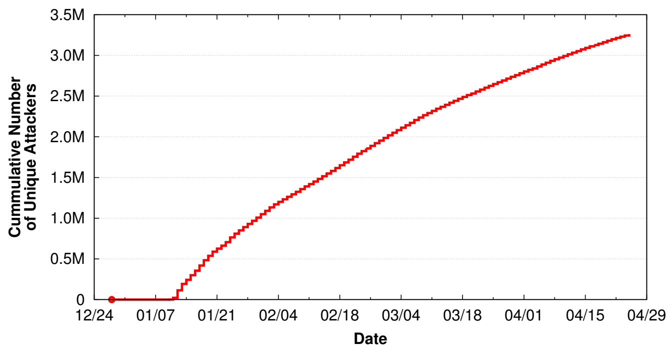

Figure 1: Number of unique Hajime IP addresses (cumulative).

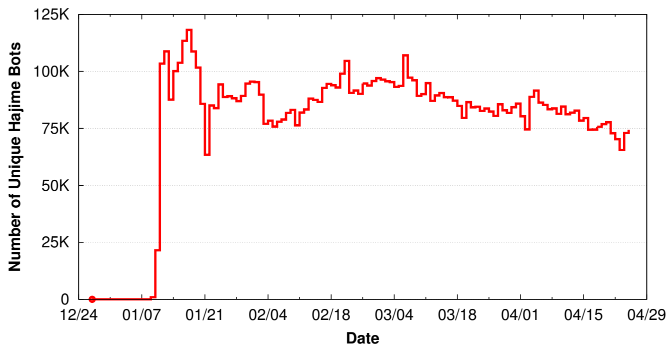

Figure 2: Number of unique Hajime IP addresses (daily).

this does not follow the more traditional logistic growth function that modern worms typically have [13]. To understand this result, it is important to keep in mind that many of the attackers in these results were compromised before our data captured them: the attack vectors that we are able to collect were introduced months after Hajime was first observed. Thus, we are unlikely to witness the *initial* propagation, but should be able to see the activity over time.

We can envision several possible explanations for the smoothness in the growth rate. One possible explanation for this is that this linear growth is by design: Hajime may be conservative in how rapidly it spreads, favoring a slow, broad coverage over a fast spread that may harm low-resource IoT devices. As we will see in Section 5, however, some Hajime bots attack in a very bursty, consistent manner, making it unlikely that that they are particularly cautious in their rate of spread.

Another possible explanation is that we are not witnessing the spread of new Hajime bots per se, but rather the spread of which Hajime bots are employing the attack vectors we are able to see. To investigate whether this is the case, we present in **Figure 2** the number of unique Hajime attacker IP addresses we observe on a daily basis. These results show that the new attack vector spread at an exponential rate among infected hosts; within three days, over 100K hosts began using the `tftp` or `wget` attacks. After this

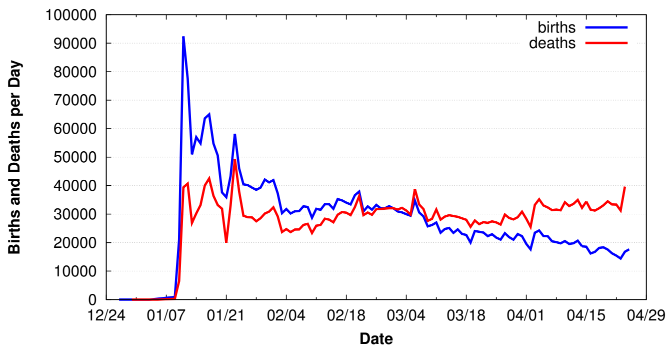

Figure 3: Daily attacker churn in Hajime. "Births" denote the first time we saw a given address, "deaths" denote the last time.

initial spike, we observe that the overall number of daily attackers is largely consistent within the range of 75K–100K.[3]

Rather, as we show next, the smooth growth of Hajime can be largely attributed to its churn.

4.2 What is Hajime's churn?

A distributed system's *churn* is the rate at which hosts enter and leave the system. To study churn in Hajime, we define the *birth* of an attacker's IP address to be the time at which we first observe the address. Likewise, the *death* of a bot is the time at which we last see its IP address in our data.

Figure 3 shows the number of births and deaths observed on a daily basis. We draw two conclusions from this data that confirm our above results. *First*, the rate at which bots are leaving the system has, as of early March 2017, overtaken the rate at which bots are entering the system. This is reflected in the slight overall downward trend in Figure 2 since March. *Second*, the birth rate changes at a roughly linear rate and does not vary widely over small windows of time (after the initial surge). This helps explain the smooth growth in Figure 1.

To be precise, these trends reflect the variability of Hajime's use of the exploit that appears in our passive DNS dataset; to what extent do they reflect the variability in the number of bots themselves? Ostensibly, Hajime bots could remain infected (and not "dead"), but shift away from using these attack vectors. For instance, one might speculate that Hajime bots launch this attack for only short periods of time and then move on to another attack vector altogether. To evaluate this hypothesis, we present in **Figure 4** the cumulative fraction of bot lifetimes (the difference in their death date and birth date). We see that the median lifetime of a bot is just under 10 hours; 39% of bots appear for more than a day's worth of data, and 17% appear for more than a month out of our nearly four-month trace. We therefore believe it is likely that, as of the beginning of 2017, Hajime bots use these attack vectors regularly, and thus our datasets reflect true bot behavior and persistence.

[3]There appears to be a small decline at the end of this data; as we process more data, we will be able to determine whether this is a diurnal trend.

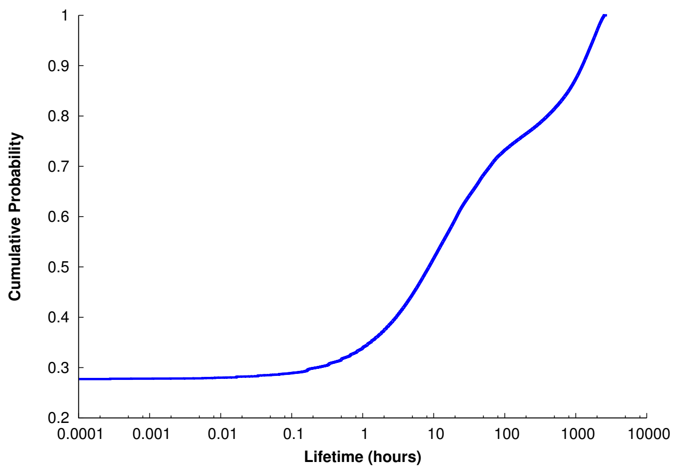

Figure 4: CDF of a Hajime bot's lifetime.

Country	Bots
Iran	1,071,399
Russia	328,269
Italy	295,377
Pakistan	197,921
India	177,682
Turkey	150,127
China	144,470
Australia	121,571
Thailand	112,023
Vietnam	110,376
Brazil	97,962
United Kingdom	52,740
France	46,308
Unknown	42,812
Trinidad and Tobago	25,147

Table 1: Top 15 countries by number of Hajime bots in that country

Taken together, these results indicate that Hajime's spread is currently on the decline; although it continues to introduce new attackers on a daily basis, it is losing bots faster than it is recovering.

4.3 Where are Hajime bots located?

To understand where Hajime bots are located, we apply the MaxMind IP geolocation database[4] to each of the attacker IP addresses we observe. **Table 1** shows the top 15 countries by the number of attacker IP addresses we have observed. Overall, these results show a globally distributed set of bots, with a particularly heavy concentration in a small number of countries.

Our results in Table 1 differ from previous honeypot-based studies [1, 7], both in terms of raw number (recall that we see one to two orders of magnitude more bots) and in terms of relative ranking of countries. In our datasets, Iran is by far the most prevalent attacker, constituting 32.8% of all of the unique attacker IP addresses we see,

[4]https://www.maxmind.com/en/geoip2-databases

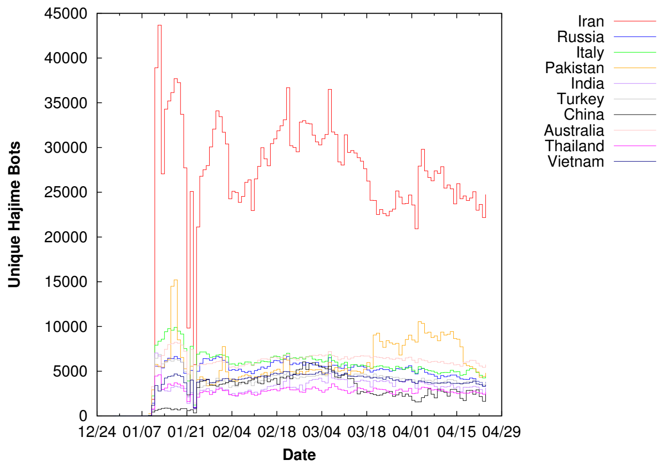

Figure 5: Unique Hajime bots for the top-ten countries (daily)

whereas Radware's honeypot study found only 9% of their hosts from Iran. Conversely, Vietnam and Brazil have appeared in the top three of other studies' most prevalent countries, yet in our data they constitute only 3.4% and 3.0%, respectively. The methodological distinction between our study and these prior studies is that while they rely on fixed honeypots themselves receiving attack traffic from compromised sources, we leverage a distributed, global "honeypot" of thousands of machines that are not vulnerable. Iran is known to have a sophisticated system for detecting and throttling communication using unknown protocols [3], and this system may interfere more with the attack traffic used by prior studies than with the DNS traffic used here.

To understand whether our overall numbers are representative over time, **Figure 5** shows the daily unique number of Hajime bots from each of the top ten countries from Table 1. We make four key observations from these results. *First*, it is immediately clear that Iran constitutes a large portion of the overall population; indeed, the shape of the aggregate number of bots in Figure 2 is largely determined by the number of Iranian bots on a given day. *Second*, while most countries exhibit a relatively consistent number of bots across days, Iran and Pakistan see considerable deviation. Although Pakistan is ranked fourth in terms of overall bots, it had the second most number of bots in April 2017.

Third, there is a pronounced dip in late January 2017 that affected multiple countries. We have verified that this is not a measurement artifact (we collected no less data on those days than on others). Interestingly, this dip is correlated with a BrickerBot [8] attack we also observe with our passive DNS dataset. As of this time, however, we are unable to verify the root cause.

Finally, the results in Figure 5 show that our overall country ranking is reflective of true bot activity over time. It is possible that the raw number of IP addresses overestimates the number of attackers because IP addresses can (and often are) reassigned to home network addresses; sometimes on the order of every few hours [12]. However, using RIPE Atlas probes in Iran, we verified that ISPs in Iran do not appear to change IP addresses every day,

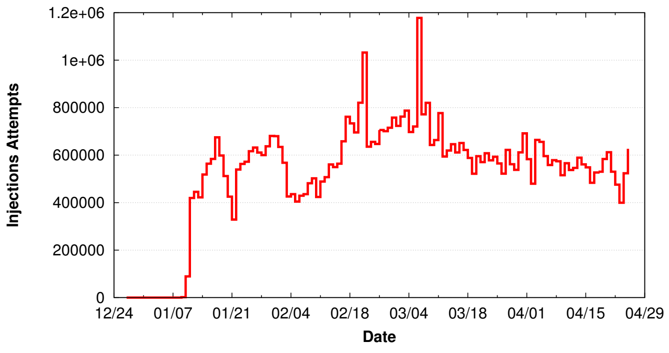

Figure 6: Hajime injections per day.

lending confidence in our results on the number of unique bots per day.

Summary The results in this section provide the most complete, longitudinal view of the Hajime botnet to date, yielding an order of magnitude more attacker IP addresses than previous studies. Our main results point to a decline in the number of daily active Hajime bots, dipping from a peak of 115K at the beginning of the year to approximately 75K today. We find that these dynamics are largely attributable to moderately high churn rate; tens of thousands of new bots are "born" every day, but a nearly equivalent amount "die." Next, we turn our analysis to what these bots do while alive.

5 ATTACK ACTIVITY

In this section, we investigate the attack attempts made by the Hajime botnet. Because our passive DNS dataset only captures the `tftp` and `wget` attacks, we limit our analysis to those. We also compare Hajime's attack dynamics to those of Mirai.

5.1 How often do bots attack?

We begin our study into Hajime's attack behavior by looking at the daily number of injection attempts made by Hajime over time, as shown in **Figure 6**. From this figure we draw three key observations. *First*, soon after adding the attack vector to Hajime, its bots started employing it quickly, reaching over 400K daily injections attempt within three days after rollout. *Second*, the overall number of injection attempts on a daily basis has remained quite high, peaking at nearly 1.2M attempts, but typically averaging roughly 600K per day (on average roughly 7 injection attempts per active bot per day). *Finally*, the overall distribution reflects the number of bots, including the two pronounced spikes in mid-February and Mid-March.

Because the number of daily injection attempts tracks so well with the daily number of bots, this would seem to indicate a consistent distribution of injection attempts across bots. **Figure 7** shows the number of attacks issued by each bot as a distribution across all 3.2M Hajime bots in our dataset. These results indicate that the number of injection attempts per attacking IP address exhibits a skewed distribution, with a median of 3, a 99th percentile of 126, and one IP address in France sources 28,776 attacks. Combining

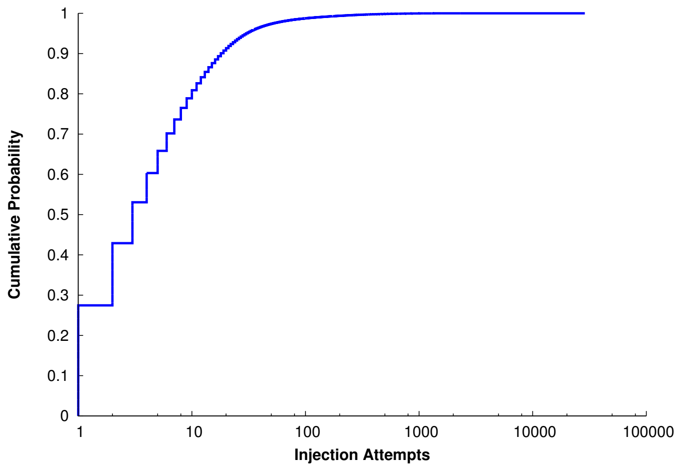

Figure 7: Distribution of the number of attack attempts issued per Hajime bot. (Note the x-axis is log-scale.)

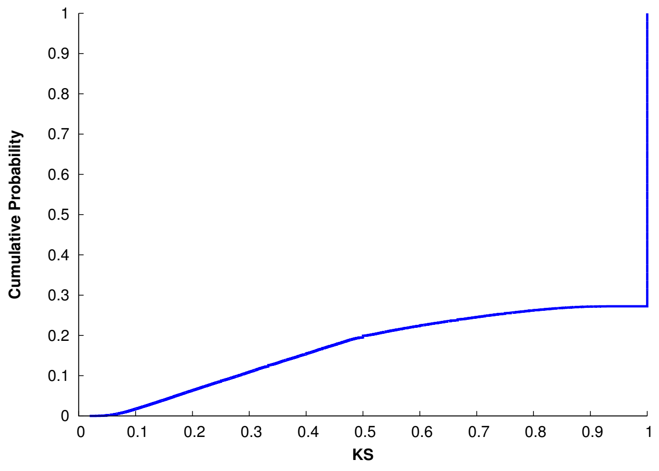

Figure 8: The Kolmogorov-Smirnov (KS) score comparing the distribution of Hajime bots' rate of attack versus a perfect uniform attack rate. A value of zero represents a consistent rate over time; a value of one represents bursty attacks all at once.

this result with the fact that Hajime bot lifetimes tend to be somewhat short (Figure 4) leads us to conclude that many Hajime bots enter the system, issue a small number of injection attempts, and then depart (or, at the very least, cease to launch more `tftp`/`wget` attacks).

This leaves the question of how more persistent, active bots pattern their injection attempts: do they send in quick bursts, or do they spread their injection attacks over time? As an evaluation metric, we use the Kolmogorov-Smirnov (KS) test: for each attacker, we compute the CDF of the attacker's injection attempts over the course of its lifetime, and we compare it to the CDF that would occur if the bot were to spread its attacks out perfectly evenly in time. The KS test reports the largest magnitude difference between the CDFs: a value of zero means there is no difference in the distributions (the attacks were evenly dispersed throughout the bot's lifetime), while a value of one in this case means the bot performed virtually all of its attack injections at its birth date.

Figure 8 shows the distribution of these KS scores for all bots that have lifetimes of at least 10 hours and that issue at least 10 injection attempts. We make two key observations. *First*, 72% of the bots have a KS score near one, confirming our hypothesis above that many bots perform almost all of their injection attempts in a bursty manner. *Second*, the other 28% of hosts have surprisingly low KS scores, indicating that they take a more measured approach to attacking others. Roughly 5% of the hosts have a KS score near zero, indicating they spread their injection attacks very evenly over their 10 or more hours of lifetime.

To understand from where the injection attempts originate, we show in **Table 2** the top 15 countries by the number of injection attempts made *from* that country. The number of attacks from a given country is largely a function of the number of bots within that country, with some exceptions. For instance, Australia is responsible for the second most number of attacks, and yet has only the 8th most bots (8.8× fewer bots than Iran, but only 3.1× fewer injection attempts). Whether or not we obtain a given injection attempt is partly due to the location (and latency and resolver software) of the target's local resolver; thus, to determine whether these results may be biased, we next turn to where the locations of the *targets* of the attacks.

Country	Injection Attempts
Iran	7,773,806
Australia	2,498,408
Italy	2,227,875
Russia	1,921,586
Pakistan	1,856,696
China	1,814,055
Vietnam	1,809,609
India	1,660,489
Turkey	1,560,802
Republic of Korea	1,545,808
Thailand	1,202,764
United States	769,120
Brazil	680,073
United Kingdom	529,321
Trinidad and Tobago	521,882

Table 2: Top 15 countries, sorted by the number of Hajime injection attempts from that country

5.2 Whom does Hajime target?

In reasoning about the attack targets, it is important to recall that our passive DNS data provides us with *failed* attempts to *non-vulnerable* hosts. Thus, there will also be many injection attempts that succeed; because these successful attacks will create new attackers, we can in essence infer where the successful attacks took place (where the attackers are located).

We present in **Table 3** the 15 countries that are targeted most often in our dataset. Note that these do *not* align with the countries with the most attackers—Iran does not even show up in the top 15.

Country	Resolvers
United States	1489
France	621
United Kingdom	619
Russia	556
Turkey	320
Germany	292
Australia	209
Canada	200
Argentina	195
Vietnam	178
Italy	171
Brazil	158
India	141
Netherlands	132
Philippines	132

Table 3: Top 15 countries, sorted by the number of (non-public) resolvers within that country that are targeted by Hajime.

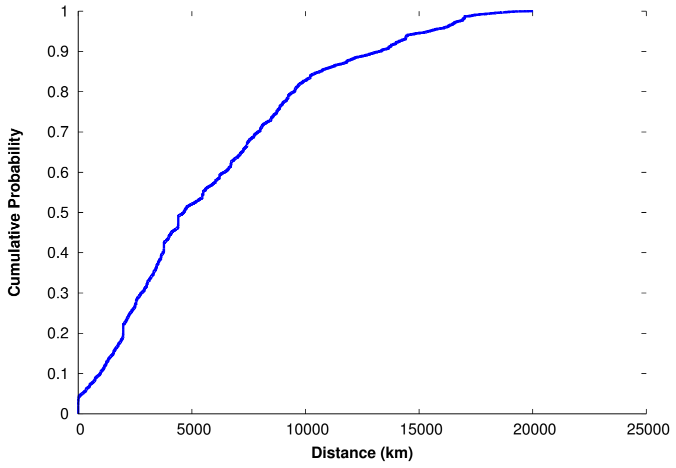

Figure 9: Distribution of Hajime bots' distances to the resolver of their intended victims (not including open resolvers).

Intuitively, the reason for this may be rooted in the fact that we are only seeing the injection attempts against the *non-vulnerable* hosts. This suggests that the fraction of vulnerable hosts varies considerably across countries. An interesting area of future work would be to fingerprint devices on a country-by-country basis to investigate the relative vulnerability rates.

Having explored the most likely sources and targets of attacks, we now seek to understand how attackers and targets are paired. For each attack attempt, we IP-geolocate the attacker (whose IP address is in the DNS query) and the victim's DNS resolver (the source IP address of the DNS query), and compute the great-circle distance between them. We ignore public ("open") resolvers—ones that service queries for any host—in the expectation that the non-public resolvers are geographically proximal to their clients (the targets of the attack injection) [10]. As a result, we believe these distances to be representative of the distance from attacker to target. Of the 63,094,223 total resolvers from which we receive backscatter Hajime attacks, we were able to identify 15,913,183 (25.2%) of them as open resolvers; our analysis focuses on the other 47,181,040 (74.8%).

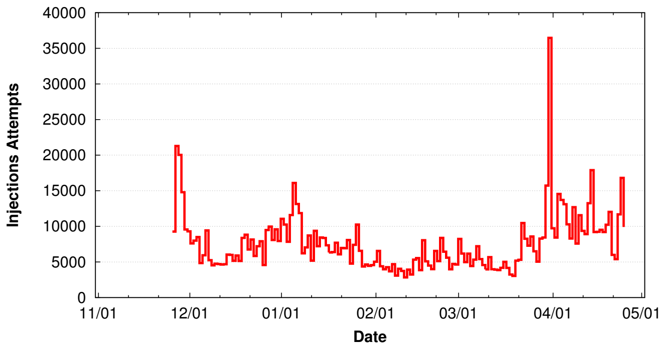

Figure 10: Mirai injections attempts per day.

Figure 9 shows the distribution of these distances. We make three key observations. *First*, a sizeable fraction (5%) have an effective distance of 0 km, indicating they are attacking their own country or even their own network. *Second*, there is no strict bound on the distance an attacker's query can travel; in fact, we see some queries travel to the opposite side of the Earth (the longest query travels half the circumference of the Earth).

Finally, and most importantly, there does not appear to be any geographic preference in the injection attempts. With a median of 5000 km (3106 miles), attackers do not have a strong preference to remain close. Although Figure 9 does not show a uniform distribution, Hajime bots may still be choosing attack targets uniformly at random; the difference in distribution is likely due to the fact that the location of attackers is not uniformly distributed (from most hosts, there are more targets within 3000 km than there are 9,000–12,000 km away).

5.3 How does Hajime compare to Mirai?

We close this section by comparing Hajime's attack patterns to those of its predecessor, Mirai. Because Mirai's attack vector uses hostnames instead of IP addresses, we are unable to geolocate attackers. However, we can still obtain counts of injection attempts and geolocate the target; we analyze these both here.

Figure 10 shows the number of daily injection attempts from Mirai (to non-vulnerable hosts). Compared to Hajime (Figure 6), Mirai has far fewer injection attempts: typically by an order of magnitude. Mirai does not appear to be subject to the same dip in the third week of January that Hajime is; this may indicate that it was a Hajime-specific event, but we are unable to confirm that at this time.

Although the raw number of injection attempts differ greatly between Hajime and Mirai, their targets have surprisingly similar features. **Table 4** shows Mirai's top targeted countries, which, in terms of rank, tracks very similarly to Hajime's (Table 3). One notable difference is that Mirai unsuccessfully targets Iran disproportionately more often than Hajime. It is possible that Mirai fails in

Country	Resolvers
United States	728
United Kingdom	490
Russia	378
Turkey	308
Germany	227
France	223
Italy	181
Vietnam	168
India	142
Brazil	135
Argentina	132
Iran	126
Australia	118
Canada	112
Spain	112

Table 4: Top 15 countries by number of Mirai victim (non-public) resolvers

these attempts because the Iranian nodes are already compromised by Hajime (its most common country).

Summary The results in this section show that Hajime attempts to infect hosts uniformly at random, and with relatively few attacks per bot on average. We note that our results in this section are a strict lower-bound, as our dataset comprises only one of the 13 root DNS servers, and the data we get is further sampled. It is possible that the true rate of infection attempts is therefore 13× or more what we are able to directly observe. Nonetheless, our view into infection targets is far wider and more longitudinal than honeypot-based approaches, because of our use of the DNS "backscatter" that arises from attacks on non-vulnerable hosts. All together, these results demonstrate that Hajime has truly supplanted Mirai, and continues to actively (attempt to) spread; as new vulnerable devices come online, Hajime is likely to continue to proliferate.

6 COMMAND & CONTROL ACTIVITY

One of the unique aspects of Hajime is its use of an existing P2P infrastructure to coordinate command and control. We close our analysis by investigating how Hajime's bots make use of this P2P network. More precisely, we study the bots' activity on this network (how many of them join, upload, or download), not the C&C payloads themselves. Recall that active bots download a new configuration file periodically. Configuration files contain pointers to new executable modules, which can be used to change the bot's behavior after deployment. Hajime configuration filenames have a standard format [7], which changes once per day. Configuration files are distributed using the BitTorrent DHT, which implements the Kademlia DHT protocol. In our experiments, we adapt the KadNode client (version 1.0.0 [5]) to search for seeders, and to announce configuration files as described below.

We structure our analysis around the role the bots play: *seeders* host configuration files that *leechers* periodically download.

6.1 Seeder activity

We begin our analysis of Hajime's C&C by focusing on the seeders: the bots that host the configuration files that other bots download.

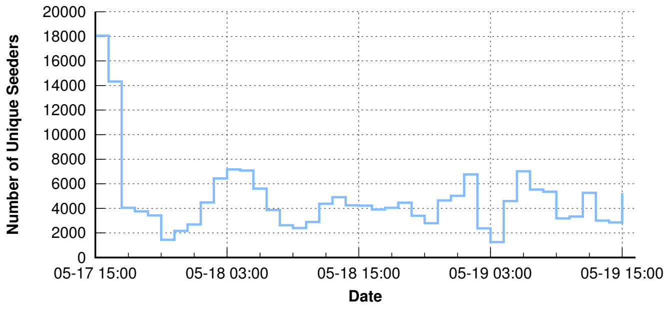

Figure 11: Number of seeders in the DHT over time.

Bots download new configuration files periodically. The set of nodes that host new configuration files changes, and the BitTorrent DHT is used to synchronize the bots with the current "seeders" of the configuration files. We conducted periodic searches of the BitTorrent DHT for names corresponding to Hajime configuration files. Each peer we ran was an EC2 instance, and they were identified by EC2 to be located in data centers in California, Ireland, Mumbai, Seoul, Sydney, Frankfurt, London, Sao Paulo, Singapore, and Northern Virginia. The BitTorrent DHT is built over the Kademlia protocol: each query uses the `Lookup` command, and can take several minutes to complete. We issued queries at most once every five minutes.

Figure 11 shows the number of unique seeders we discovered binned into 72 minute periods. We chose that binning interval since, after the few initial searches, we received information about new nodes once every 72 minutes. (Subsequent searches during the same epoch did not return new information.) We believe, but have not yet confirmed, that this is due to a rate limiting feature employed by BitTorrent DHT peers. Our data shows that the config files are seeded by many peers: over 2.5 days, we retrieved 42, 793 distinct seeder IP addresses, with 2000–6000 seeders being present at all times.

The change in numbers of seeders over time show that they are not persistent, and we analyzed the lifetime of seeders by noting the interval over which their IP address was retrieved over the DHT: this is plotted in Figure 12. The seeder lifetime plot shows a few interesting features. First, the nearly vertical line after one hour corresponds directly to the 72 minute interval over which we received new data. The data to the left of the vertical line corresponds to the data collected during startup, when our queries were not rate-limited allowing us to confirm lifetimes shorter than 72 minutes. The 72-minute interval accounts for fully 80% of the nodes (i.e., after startup, we saw these nodes only once in our logs.) Seeder lifetime is nearly linear beyond the hour, with 10% of the nodes persisting over 10 hours. No seeder in our data persisted beyond one day: 7 out of 42, 794 showed a lifetime of 86, 392 seconds (eight seconds less than one day). This leads us to believe that a bot is nominated to be a seeder for at most one day.

[5]Available at `https://github.com/mwarning/KadNode`

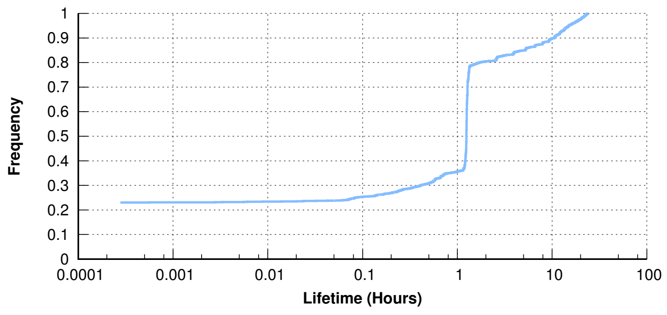

Figure 12: Lifetime of nodes that seed config files.

6.2 Leecher activity

We now turn to the leechers: the bots that download the configuration files.

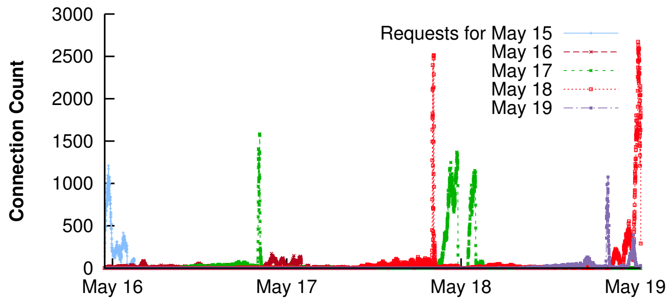

Figure 13: Leecher download requests separated by config file date.

Kademlia allows DHT clients to "announce" named content. Peers seeking the named content search for the name; if the name had been announced, the DHT search returns a list of announcing peers. The actual data transfer is conducted outside the DHT itself using the μtp protocol.

During May 15, 2017 (2300 hours UTC) until May 19, 2017 (0100 hours UTC), we ran twelve BitTorrent DHT peers. They were colocated with our seeder-searcher nodes, but we ran one extra instance each in Frankfurt and Northern Virginia. These peers announced the availability of 200 Hajime config files, starting from Nov. 1, 2016. We assigned a different download port for each config file, enabling us to track requests for individual files. Over the 74 hours when we gathered data, we received 1.12 million requests for config file downloads from 51,039 unique IP addresses.

The bots requested files for 115 different days, though 1.05 million (93.7%) of the requests were for files for five days starting May 15, 2017. Figure 13 shows requests over time for these five files. Prior analysis had suggested that Hajime bots download a config file every ten minutes [7]. Our data shows much burstier behavior, with large activity peaks around midnight UTC. Most download requests for a given config file corresponds cleanly with the date; however, for each file, there are two peaks: one around midnight of the file's date in UTC, and the second around midnight the next day. Note that our data shows a much smaller *second* request spike for May 16th, but the other days with high traffic volume (15th, 17–19th) follow the trend of two request peaks.

6.3 Correlating DNS and DHT data

Finally, we present a correlation comparison of our passive data, and the data collected by actively participating in the DHT. Figure 14 shows how the bots identified using the DNS data relate to seeders and leechers in the BitTorrent DHT. The figure counts all bot IP addresses that we had identified in our DNS data (starting December 2016): these addresses are intersected with the seeder and leecher data collected over three days in May. (We also analyzed data from all three sources over a twelve hour period. The number of leechers and passive nodes reduced by a factor of 3, but the number of seeders only reduced by a third. The intersections reduced proportionately.) The intersection data shows the validity and utility of both approaches, as they both locate bots not found by the other. The passive DNS approach finds many more bots, partly because any bot that tries to infect another causes it to be logged depending on whether the attempt is successful (and, in our case, the probability that the DNS query is routed to the root we monitor and is sampled by our data collection.) Aggressive efforts to propagate the worm causes the probability of detection to rise commensurately, and whether the DNS query is issued or not is not in control of the bot (author), as this is a pre-programmed reaction from the victim. Searching the DHT for the config file seeders is also an effective way to find bot addresses—about 30% of the seeders were also seen to launch active attacks that were logged by the DNS data. Hosting configuration files and inviting downloads is a seemingly less effective way to find bots, undoubtedly because there are many thousands of seeders, which reduces the probability that a particular bot will visit a specific seeder (or seeder-honeypot).

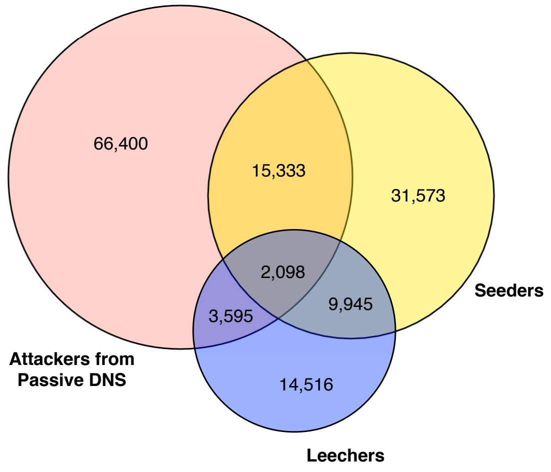

Figure 14: Hajime Bots in the DHT and in the DNS data.

7 DISCUSSION

In relying on a peer-to-peer substrate for command and control, Hajime represents an evolution in the design of botnets. Relying on the DHT enables a further level of indirection that further obfuscates the command and control structure; however, the very same design choice also exposes Hajime to generic attacks that DHTs are vulnerable to. In the rest of this section, we discuss a few of these attacks, along with a weaknesses that are specific to Hajime.

Decentralized DHTs are vulnerable to Sybil attacks [4], whereby malicious nodes join the DHT multiple times as different peers, with a view to "taking over" part of the namespace. In general, nodes are hashed to points in the DHT namespace uniformly at random using a cryptographically strong hash function, and it is not usually not easy to join a DHT at a specific point in the namespace. However, a sufficiently determined attacker could possibly hijack parts of the namespace that host the configuration file, thus stopping bots from obtaining files on a given day. Since the filename changes daily, this attack would have to be sustained in order to be effective. Encrypting or obfuscating the filename is not a particularly viable solution, since the seeders have to be looked up by DHT nodes, which is not controlled by Hajime. Finally, note that the attacker does not have to specifically host the key that corresponds to the configuration file: the attacker can be effective if it can generate "nearby" keys that are on the lookup path. The attacker would simply discard lookups for the configuration file key—this type of attack is called an "insertion" attack, whereby the attacker inserts themselves on the lookup path, and has been shown to be effective against Kademlia DHTs in the wild [11].

A somewhat more blunt approach is to simply look up the current set of seeders, and then disable them, perhaps by mounting a Denial of Service or an Eclipse attack [11]. Our combined seeder-leecher data points out an interesting contradiction. The number of seeders is at a minimum when the number of configuration file requests peak. It is currently unclear whether this behavior is by design (seeders are allowed to voluntarily stop seeding after they have uploaded a sufficient number of bytes, perhaps to evade detection), or whether it is a result of a yet unforeseen interaction. Our data strongly suggests that bots seed for at most one day (none of our 42K+ seeders lasted more than one day). If there is also a bandwidth limitation along with a time limit, then an easy way to reduce the seeder set is to simply repeatedly download the configuration file.

Hajime is also vulnerable to attacks specific to its code base and choice of DHT. There are previously published attacks on Kademlia [11, 15], which show that a single attacker with 100 Mbps of bandwidth can disrupt 75% of all lookups on a Kademlia DHT. Other attacks, such as the publish attack [11], can be both very successful against Hajime and easily mounted. The publish attack notes that a node holding a key (for the configuration file, in our case) returns, at most, the latest 300 announcements it receives for that key. An attacker could simply flood the network with fake "publish" messages, and thus disrupt the seeder-location process.

8 CONCLUSION

Hajime is a new, large IoT botnet that is distinguished by its use of a existing DHT for command-and-control. In this paper, we have analyzed the spread and operation of Hajime using two different methods. Our first method employs "DNS backscatter": we observe that a specific form of *unsuccessful* Hajime attacks (that exploit the so-called TR-064 vulnerability) can result in a DNS query for an invalid top-level domain. This query cannot be resolved by any regular DNS resolver, and is routed to the DNS roots. Using packet captures from a DNS root server, we uncover millions of bots and attacks since Hajime's use of TR-064. Our unique vantage point enables us to track the spread of the bot globally, and gain insight into its macroscopic behavior at a scale that is not easily replicable using existing methods such as honeypots.

Simultaneously, we present a second set of experiments that focus on Hajime's use of the DHT. We identify Hajime bots on the DHT using two methods, and provide a detailed analysis of their lifetime and command and control behavior. Finally, we present an analysis of vulnerabilities in Hajime itself due to its reliance on a DHT. Our two measurement methods complement, enabling both a global view of the bot's spread, and a closer view of individual bot's lifetime and behavior.

REFERENCES

[1] Radware ERT Threat Advisory. 2016. Hajime – Friend or foe? Online: `https://security.radware.com/ddos-threats-attacks/hajime-iot-botnet/`. (2016).

[2] Akamai. 2016. Akamai's State of the Internet / Security, Q3 2016 Report. Online: `https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/q3-2016-state-of-the-internet-security-report.pdf`. (2016).

[3] Simurgh Aryan, Homa Aryan, and J Alex Halderman. 2013. Internet Censorship in Iran: A First Look.. In *Proceedings of the 3rd USENIX Workshop on Free and Open Communications on the Internet.*

[4] John R. Douceur. 2002. The Sybil Attack. In *First International Workshop on Peer-to-Peer Systems (IPTPS '01).*

[5] DSLHome-Technical Working Group. 2004. DSL Forum TR-064: LAN-Side DSL CPE Configuration. Online: `https://www.broadband-forum.org/technical/download/TR-064.pdf`. (2004).

[6] Dyn. 2016. Dyn Analysis Summary of Friday October 21 Attack. Online: `https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/`. (2016).

[7] Sam Edwards and Ioannis Profetis. 2016. Hajime: Analysis of a decentralized internet worm for IoT devices. Online: `https://security.rapiditynetworks.com/publications/2016-10-16/hajime.pdf`. (2016).

[8] Dan Goodin. 2016. BrickerBot, the permanent denial-of-service botnet, is back with a vengeance. Online: `https://arstechnica.com/security/2017/04/brickerbot-the-permanent-denial-of-service-botnet-is-back-with-a-vengeance/`. (2016).

[9] Waylon Grange. 2016. Hajime worm battles Mirai for control of the Internet of Things. Online: `https://www.symantec.com/connect/blogs/hajime-worm-battles-mirai-control-internet-things`. (2016).

[10] Krishna P. Gummadi, Stefan Saroiu, and Steven D. Gribble. 2002. King: Estimating Latency between Arbitrary Internet End Hosts. In *ACM Internet Measurement Workshop (IMW).*

[11] Thomas Locher, David Mysicka, Stefan Schmid, and Roger Wattenhofer. 2010. Poisoning the Kad Network. In *11th International Conference on Distributed Computing and Networking (ICDCN).*

[12] Ramakrishna Padmanabhan, Amogh Dhamdhere, Emile Aben, kc claffy, and Neil Spring. 2016. Reasons Dynamic Addresses Change. In *ACM Internet Measurement Conference (IMC).*

[13] Stuart Staniford, Vern Paxson, and Nicholas Weaver. 2002. How to 0wn the Internet in Your Spare Time. In *USENIX Security Symposium.*

[14] Brett Stone-Gross, Marco Cova, Lorenzo Cavallaro, Bob Gilbert, Martin Szydlowski, Richard Kemmerer, Christopher Kruegel, and Giovanni Vigna. 2009. Your Botnet is My Botnet: Analysis of a Botnet Takeover. In *ACM Conference on Computer and Communications Security (CCS).*

[15] Peng Wang, James Tyra, Eric Chan-Tin, Tyson Malchow, Denis Foo Kune, Nicholas Hopper, and Yongdae Kim. 2008. Attacking the Kad Network. In *Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks (SecureComm '08).*