Quantifying Autonomous System IP Churn using Attack Traffic of Botnets

Harm Griffioen
Hasso Plattner Institute for Digital Engineering
University of Potsdam, Germany
harm.griffioen@hpi.de

Christian Doerr
Hasso Plattner Institute for Digital Engineering
University of Potsdam, Germany
christian.doerr@hpi.de

ABSTRACT

To connect to the Internet, hosts are assigned an IP address by their network provider by which they exchange data. As such, IP addresses are frequently used as a proxy metric to count the number of hosts on a network, or to quantify particular phenomena such as the size of botnets or the infection statistics of malware. Although a single host is typically linked to a single IP address at a given moment, this relationship is frequently not stable over time due to IP churn. As network operators dynamically assign IP addresses to clients for a specific lease duration, after expiry of this lease a host obtains a new IP address, thereby leading to overestimations of active host counts or malware infections.

In this paper, we present a novel method to detect and quantify IP churn in autonomous systems on the Internet by exploiting a weakness in the packet generation algorithm and random number generation of the Mirai IoT malware. These design shortcomings allow us to re-identify the same IoT infection when the host resurfaces on the Internet with a different IP address with very high confidence, and thereby characterize how IP addresses in provider netblocks churn over time. As Mirai is widespread with hundreds of thousands of infected devices worldwide and uses the faulty RNG output to actively scan the Internet, our methods enables world-wide measurements of IP churn to be done efficiently and completely passively.

KEYWORDS

IP churn, Mirai, network telescope, cyber threat intelligence

ACM Reference Format:
Harm Griffioen and Christian Doerr. 2020. Quantifying Autonomous System IP Churn using Attack Traffic of Botnets. In *The 15th International Conference on Availability, Reliability and Security (ARES 2020), August 25–28, 2020, Virtual Event, Ireland*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3407023.3407051

1 INTRODUCTION

In the Internet and Internet Protocol-based networks, networked hosts are assigned an IP address as an identifier, using which they exchange data packets. As every host has to have an IP address to participate in the network, the number of IP addresses is often taken as a proxy to count the number of hosts in particular network. Such measurements include Internet surveys on deployment statistics in home, enterprise or cloud environments [11, 19], studies on how the address space that is assigned to network operators is actively used in the Internet [5], or to obtain insights into the spread and installation sizes of botnet infections worldwide [4, 17].

While every host requires an IP address to send and receive packets, this relationship between IP address and host is not strictly bijective – especially not beyond the short term. The is primarily due to two reasons: first, networks may issue IP addresses dynamically to its hosts, for example using the Dynamic Host Configuration Protocol (DHCP). If a host reconnects after the lease has expired, it gets issued a new network address. If counting hosts with a certain characteristic based on the number of network addresses, we would thus overestimate the actual occurrence of the phenomenon given this so-called IP churn. Second, also in the reverse direction there is not a solid one-on-one relationship: as the IPv4 address space is exhausted, network operators – especially in mobile networks – place connecting customers behind a carrier-grade network address translation (NAT), and all hosts behind such NAT appear to others as a single host as they all have the same IP address. (Carrier-grade) NATs would therefore lead to a significant underestimation.

These two operational practices in networks in result mean that IP addresses cannot be used as an effective surrogate to quantify the behavior and characteristics of hosts on the Internet. For example, in [18] the authors showed using sink holing that the number of bots in the Torpig network is overestimated by an order of magnitude. Similarly, [9] showed that bots based in Brazil had on average 1.3 IP addresses per day. These measurement biases could however be addressed if we would have detailed information per autonomous system (AS) or network address block on the presence of IP churn.

Despite this importance, only relatively little work has been done on measuring and characterizing IP churn in networks, primarily due to the fact that assessing this phenomenon at Internet scale is challenging and obtaining a ground truth from the plethora of network operators is difficult. While some works have empirically observed IP churn [10, 15, 16, 19, 20], only two works to date provide a methodology that can be used to test networks for the presence of churn by actively sending sequences of ICMP and TCP/UDP probing packets [7, 13]. Active probing is however costly and often frowned upon by network operators, for whom large scale measurements mean additional – and in their view unnecessary – system load on their network infrastructure, and may lead to false alarms in the network monitoring systems.

In this paper, we present a novel mechanism to efficiently quantify IP churn and NAT aggregation across the Internet *without* active measurement probes. We accomplish this by leveraging scanning patterns from malware-infected IoT devices, which due to the prevalence of insufficiently secured IoT devices and the many malware strains that descended from the IoT malware Mirai are omnipresent through autonomous systems worldwide. For our measurement method we utilize a flaw in the random-number generation of Mirai and how it populates port scanning packets using this data, to reliably link and re-identify ongoing infections even when the IP address of the host has changed, as well as pinpoint if multiple infected hosts sit behind the same public IP address. This allows us to continuously obtain reliable estimates for IP churning and the location of NATs in autonomous systems across the Internet.

In this paper, we make the following three contributions:

- We show that using a flaw in the random number generation and packet generation of Mirai it is possible to link together infections over time and across different IP addresses to the same host with very high confidence, and thereby identify IP churn behavior in networks.
- We demonstrate that this flaw can be exploited at scale across the entire Internet given the prevalence of infected IoT devices. With this, we introduce the first technique for passive measurement of IP churn in the Internet. Passive assessment has a major advantage over currently used techniques, as it does not cost network operators any computational resources or bandwidth, nor does it trigger alarms in their network monitoring infrastructure. The technique proposed in this paper has thus no adverse effects on networks at all.
- We show that we can identify churn rates for a large portion of the Internet due to the wide spread of Mirai infections and how we can use this to demonstrate how autonomous systems allocate their IP blocks, implement NATs, and how network prefixes can drastically change due to churn.

The remainder of this paper is structured as follows: section 2 provides an overview of related work on IP churn and current methods to quantify it in communication networks. Section 3 describes the port scanning behavior of the IoT botnet Mirai, the random number generation used and how flaws allow us to reidentify a particular infected host even if it has been assigned a new IP address. Section 4 describes the data set used for our study. Section 5 explains the methodology we put forward in this paper, which section 6 validates against established techniques in terms of accuracy. In Section 7 we show results from the application of our measurement technique on the usage of IP churn in networks across the Internet. Section 8 summarizes and concludes our work.

2 RELATED WORK

The first work attempting to automatically quantify the volatility of IP addresses in ASes was the work of Xie et al. [20], where the authors propose a methodology that relies on IP-user mappings and routing information, where IP-user mappings are used as identifiers of which device is being used, even after a routing change. Similar methodology, leveraging uniquely identifiable features after an address change, is used in several other studies measuring churn rates on the Internet [3, 10, 12]. The proposed methods are however dependent on software logs, and do not provide a way to perform these measurements without access to these logs.

[13] employs an active method to measure IP volatility, enabling the measurement of any AS. They create and verify a high-performance, scalable method to probe a large portion of the Internet, where the authors consider an IP address as changed if it does not respond on their probes anymore. The authors show their method is able to correctly estimate 72.3% of the DHCP churn rates in a mid-sized ISP network.

While successful, active probing measurements are often frowned upon by network operators, as it generates an unwanted stream of packets that they deem unnecessary. But passive methods have not yet been able to measure a large part of the total network space. The largest semi-passive measurement study is done by Padmanabhan et al. [15], who used data gathered from 3,038 RIPE Atlas probes hosted across 929 ASes and 156 countries to give an overview of the reasons Internet addresses change, and how different operator policies show in the changes. The proposed methodology is however dependent on device logs located in a number of ASes, requiring access to such a datasource.

In this work, we propose a method based on solely passive traces that can be observed from any network and does not require additional datasources such as logs. We use uniquely identifiable features of the widespread Mirai botnet that remain constant after an address change. We extend on current works and provide a new method to passively measure this churn, that can be used to investigate over 12,000 networks in our study of 9 months. We show that due to the large amount of probes within networks, we are able to accurately map the structure of different networks to prefix allocations, and are therefore able to provide insight in differences between ASes in terms of network block allocation.

3 THE MIRAI BOTNET AND ITS PACKET GENERATION

Over the past decade, computer networks and especially mobile networks have experienced a drastic transformation with the pervasive introduction of low cost devices of limited, dedicated functionality. Often referred to as the “Internet of Things” (IoT), this development has often been seen with concern, as the economics of the IoT, such as the low cost and very deep supply chain mean that security is frequently of little to no concern during system design, deployment and operation. Given the plethora of devices now connected to the Internet and their relative state of insecurity, this has created the perfect storm for massive compromises.

In 2016, this issue became mainstream media, when the IoT-targeting malware Mirai launched a major distributed denial-of-service (DDoS) attack on major Internet infrastructure operators, surpassing all previously known attack volumes by a factor of two, only to double this record again a few weeks later. Soon after the Mirai source code had been posted on the Internet, copycats entered the scene, a behavior also seen in other DDoS vectors [6]. In this case, a variety of actors recycled Mirai’s source and introduced minor alterations to create their own IoT botnets. While new bot masters typically changed the way the malware identified itself, the passwords it used or the ports it targeted, they all effectively left the way their IoT botnet generated the scan and probe packets unchanged from the original Mirai. This means that there is today an entire ecosystem of IoT malware which shares behavioral characteristics that we exploit in this paper to quantify IP churn.

The way Mirai generates its scan and attack packets exhibits some particularities. Figure 1 shows an excerpt of the Mirai source code responsible for the packet creation and random number generation. After start-up of the non-persistent malware, the software initializes a custom-built random number generator (RNG) based on the current epoch time, the process ID(s), and the time in clock ticks since the program has started. Mirai then spins off an additional thread responsible for scanning the entire Internet for potentially vulnerable hosts, where the destination addresses are chosen from the output of the random number generator. The source port and window size of all scanning probes are randomly chosen from the RNG but fixed throughout the entire execution of the malware, in other words all scanning packets from the same host will feature the same header values until the device is cleaned up or rebooted, as Mirai only exists in memory and does not store its state persistently. On the other hand, if the host gets assigned a new IP address, we can recognize and link a host to an earlier IP address as it will keep sending probes with the same configuration values.

While packet features such as a session-constant source port and window size as well as a sequence number identical to the destination IP address are necessary conditions that a probe packet was generated by Mirai, it is not sufficient to conclusively link it to a particular infection. Thus, in order to establish that a train of packets comes from the same host – especially when we are aiming to link together hosts assuming new IP addresses over time due to IP churn –, we need to establish that the data which was used to populate the packets came from the same instance and initial seed.

As discussed above, Mirai's RNG output is used to choose the session-permanent source port and window size, and for each probe two numbers are drawn to set the packet's IP ID and destination IP. While the internal state of Mirai's RNG is four times a 32 bit value, in practice the algorithm does not start with 128 bits of entropy. As shown in the listing in figure 1 and in the schematic representation in figure 2, the state is seeded using the epoch, process IDs, and clock ticks since startup. As Mirai is aggressively scanning the Internet at high speed, we experimentally determined that we will receive a scan packet in our measurement infrastructure after approximately 15 minutes, but even if we conservatively estimate that scan probes arrive only within a day after startup of the malware, the overall entropy in the epoch value is merely 16 bits. Process IDs are limited to 16 bits. Clock() when called in a thread returns the number of ticks since the fork(), which has only taken place four instructions earlier and is thus of limited value as well. In glibc prior to v2.18 which is often used in IoT devices, clock()'s resolution was limited to a granularity of 10,000 ticks, thus this value is either 0 or 1. This means that the overall complexity of the internal state is a mere 33 bits, which we can even trivially brute-force and determine for incoming scanning packets which the internal seed of the random number sequence is.

4 DATASET

In order to quantify IP churn using Mirai's faulty random number generation process, we need a measurement infrastructure to capture the connection attempts by Mirai bots. As Mirai connects to

```
// main.c
int main(int argc, char **args) {
   …
   rand_init();
   …
   scanner_init();
   …
   killer_init();
   …
   while (TRUE) {
         …
         establish_connection(); // CNC connection
   }
}
// scanner.c
void scanner_init(void) {
   int i;
   uint16_t source_port;
   …
   // Let parent continue on main thread
   scanner_pid = fork();
   …
   LOCAL_ADDR = util_local_addr();

   rand_init();
   fake_time = time(NULL);
   …
   do {
      source_port = rand_next() & 0xffff;
   } while (ntohs(source_port) < 1024);

   struct iph = (struct iphdr *)scanner_rawpkt;
   struct tcphdr *tcph = (struct tcphdr *)(iph + 1);
   …
   iph->id = rand_next();
   …
   tcph->dest = htons(23);
   tcph->source = source_port;
   tcph->doff = 5;
   tcph->syn = TRUE;
   …
   while (TRUE) {
      …
      for (i = 0; i < SCANNER_RAW_PPS; i++) {
         struct sockaddr_in paddr = {0};
         struct iphdr *iph = (struct iphdr *)scanner_rawpkt;
         struct tcphdr *tcph = (struct tcphdr *)(iph + 1);
         iph->id = rand_next();
         iph->saddr = LOCAL_ADDR;
         iph->daddr = get_random_ip();
         iph->check = 0;
         iph->check = checksum_generic((uint16_t *)iph,
            sizeof (struct iphdr));

         if (i % 10 == 0) {
            tcph->dest = htons(2323);
         }else{
            tcph->dest = htons(23);
         }

         tcph->seq = iph->daddr;
         …
         sendto(rsck, scanner_rawpkt, sizeof (scanner_rawpkt),
            MSG_NOSIGNAL, (struct sockaddr *)&paddr, sizeof (paddr));
}

static ipv4_t get_random_ip(void) {
   uint32_t tmp;
   uint8_t o1, o2, o3, o4;
   do {
      tmp = rand_next();
      o1 = tmp & 0xff;
      o2 = (tmp >> 8) & 0xff;
      o3 = (tmp >> 16) & 0xff;
      o4 = (tmp >> 24) & 0xff;
      while (o1 == 127 ||// 127.0.0.0/8           - Loopback
         (o1 == 0) ||        // 0.0.0.0/8          - Invalid address space
         (o1 == 15 || o1 == 16) || // 15.0.0.0/7 - Hewlett-Packard Company
      … ); // Truncated, includes more blacklists
   return INET_ADDR(o1,o2,o3,o4);
}
// rand.c
static uint32_t x, y, z, w;
void rand_init(void) {
   x = time(NULL);
   y = getpid() ^ getppid();
   z = clock();
   w = z ^ y;
}

uint32_t rand_next(void) { //period 2^96-1
   uint32_t t = x;
   t ^= t << 11;
   t ^= t >> 8;
   x = y; y = z; z = w;
   w ^= w >> 19;
   w ^= t;
   return w;
}
```

Figure 1: Excerpt of the random number generation and packet creation routines of the Mirai malware.

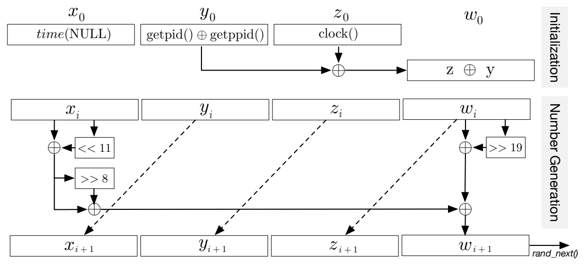

Figure 2: Random number generation of Mirai.

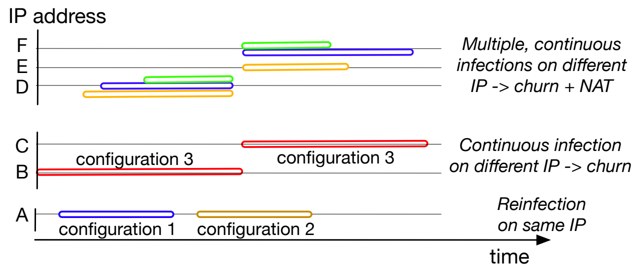
IP address
F
E
D
C
B
A
Multiple, continuous infections on different IP -> churn + NAT
configuration 3
configuration 3
Continuous infection on different IP -> churn
configuration 1
configuration 2
Reinfection on same IP
time

Figure 3: Three cases of infections and churn. For IP address *A*, no churn occurred, whereas IP address *B* shows a churn. IP address *D* is located in a NAT, as multiple infections are running on the same IP.

random IP addresses in the IPv4 space, this only requires passive listening on any IP address for incoming frames matching the signature used by Mirai. Of course, while a quantification by monitoring one IP address would in principle work, it will take long before bots from all autonomous systems in the world will be visible.

While a single IP address is sufficient to quantify churn, the measurement speed and reliability grows exponentially with the amount of IP addresses used for monitoring. [1] derives an estimation for the number of IP addresses required and the resulting measurement confidence, which shows that already with a few hundred IP addresses we can obtain estimates below 1% error margin. In this study, we rely on a network telescope of some 65,000 unused IP addresses, which is therefore able to quantify the infection of the Mirai botnet within an error margin below 0.1%.

In order to quantify churn over time, we collected probe packets for a duration of 9 months from March until December 2018 across these 65,000 IP addresses. As discussed in the previous section, Mirai's bruteforcing behavior can be distinguished based on the use of specific header values for the TCP sequence number, destination ports and random but session-fixed values for the window size and source port. Based on these filters we received a total of 2.5 billion packets during the observation period from 12,112 autonomous systems worldwide. As after reboot and reinfection the IoT devices choose a new random window size and source port, we can cluster these 2.5 billion packets into 24,042,833 individual flows and thus unique infections. 4,034,454 of these infections continued at a different IP address, thereby allowing us to quantify churn across autonomous systems worldwide.

5 PASSIVELY MEASURING CHURN

As discussed in section 3, separate Mirai infections can be distinguished based on the source port and the window size of scanning packets sent out by the malware. These two session-constant values are both 16-bit in length and generated by the Mirai PRNG, creating 32-bits of entropy with which separate infections can be distinguished from one another. As we can distinguish between infections using the session constants, we are also able to identify whether a session stops on an IP address and continues on another.

Figure 3 shows three major scenarios of the progression of Mirai infections on IP addresses. Consider the case of an IP address *A*, which shows a particular configuration in source port/window size and thus RNG seed, and sends scanning packets towards random Internet hosts using these values. If we observe these flows to stop and later on resume using a different configuration, we consider this as a separate infection on the same IP address.

Consider now the case of an infection on IP address *B* with a particular configuration. If packets stop to originate from *B* but continue using the same configuration from an IP address *C*, we can link these two IP addresses to the same infected IoT device, as this situation could only happen with a chance of one in 4 billion, not even considering the clear temporal relationship. In our dataset, we find pairs of IP addresses where a session stops at one IP address, but shows up at another within 2 hours. If we assume that a device scans the Internet with 25 packets per second, this would give us a 95% chance of this device hitting our telescope within these 2 hours. In reality, we observe consecutive packets arriving on average merely 421 sec apart, which means we will be able to capture most churns.

While the probability of finding a random matching between a session that stopped on one IP and another session that started on another is negligible, we take two additional verification steps to ensure that a session is continuous rather than randomly matched: (1) We only consider IP address changes within an AS, as IP addresses would not churn between different ASes. (2) We leverage seeding of the Mirai PRNG, as explained in section 3, to verify that the session-constant values could not have been generated within the last two hours, and is indeed a valid churn.

The third scenario we consider is that of IP churn in combination with a NAT, as shown at the top of figure 3. Imagine three infections at hosts behind that NAT with the public IP address *D*, which are identifiable based on the different configuration values. As Mirai infections close their point of entry, multiple infections originating from one IP address are unlikely, except for multiple Mirai variants that use different protocols to break into a device. While unlikely, this could still happen, and therefore multiple infections on one IP address alone are not enough to establish the use of a NAT. However, if one device would be infected by multiple others, an IP churn would change the location of all of these infections to a single IP address. When the infections instead churn to multiple IP addresses, the infections have to be in a NAT, located at different devices behind a single IP address. IP address *E* and *F* in figure 3 show this behavior, where the infections from *D* churn to two separate IP addresses. These scenarios thus allow us to identify an IP address in a NAT.

6 METHODOLOGY VALIDATION

As discussed in the related work, there are already methods to quantify IP churn based on active measurements. The key difference to the technique proposed in this paper is that our methodology does not require the installation of any test probes nor active testing of an AS, which thereby avoids any adverse effects for network operation. The assessment using backscatter from pervasive IoT infections has also the additional advantage that it quantifies IP churn continuously, at a much higher speed, and for all autonomous systems simultaneously. In this section, we will however first assess whether our proposed method delivers comparable results to active IP churn quantification presented before.

In [15], the authors characterized the IP churn behavior of 21 autonomous systems using 3,038 ATLAS probes in 2016. Table 1 presents the churn rate quantified in 2016 by [15] in hours in column d^*, together with the churn time that we obtained through our passive measurements in 2020 as column d. As we see from the table, these times are across the board identical, we see only minor differences in case of the networks of Orange and Digi Tavkozlesi where the churn time has decreased from 168 hours in 2016 to 24 hours in 2020. As both hour values are exact integer multiples of days, we attribute this to changes in operator policy over the 5 year period and not measurement errors. Five of the autonomous systems studied in [15] did not exist anymore, but are still included in the table for transparency.

In a typical operator network, not all IP addresses would be subject to IP churn however. IP addresses in blocks associated to enterprise customers would not exhibit churn behavior, similar difference in churn behavior might also exist between different customer groups. Table 1 hence also tabulates the share of flows in each autonomous system that demonstrated churn behavior, which provides an overall estimation of the number of IP addresses allocated to the DHCP pools. Column $f \leq d^*$ lists the percentage of IP addresses found in 2016 to be churning within a particular AS, the column $f \leq d$ tabulates the percentage in 2020 per our method. As can be seen in the table, not only does out method reliably detect IP churn duration compared to previously established methods, it also accurately performs these quantifications down to the level of IP blocks. The only noticeable differences are again the networks of Orange and Digi Tavkozlesi which likely changed their network operation within the 5 years timespan.

Based on these results, we can hence conclude that our proposed passive measurement technique generates results on par with established active IP churn quantification methods.

7 IP CHURN ACROSS THE INTERNET

Using the methodology described and evaluated in the previous sections, we evaluated the IP churn behavior of autonomous systems across the Internet. In total we have observed 24 million infections across 12,112 autonomous systems, of which 4 million changed their IP address during the lifetime of the infection. In this section, we are describing our observations on the utilization of this practice in the wild, demonstrate that we can distinguish clear demarcation lines between churning and non-churning IP blocks, and finally provide some quantification on how much IP-based estimations of phenomena such as botnets would be biased.

7.1 Churn rates in autonomous systems

When we look at the Internet in general, we find that IP churn is a very wide-spread practice. Earlier work usually mentions two reasons for operator-based IP churn: first, forceful disconnects were used by early operators to discourage the hosting of services on residential Internet connections [8], and second, operators want to accomplish a better utilization of generally scarce IP addresses [16]. With the development of the Internet landscape over the past 20 years, these reasons seem however less valid today: on the one hand, as IP transit bandwidth and IP hosting prices have been in a general decline for years [14], discouraging home-grown hosting might have been a sensible practice in the early Internet, but commercial hosting has become so inexpensive that it is commercially viable even for home users. On the other hand, with the move to pervasive and continuous connectivity and the resulting introduction of carrier-grade NATs, the urge to conserve publicly routable IP addresses would be less dire.

7.1.1 Commonalities of churning autonomous systems. Still, of the 12,112 autonomous systems that we observed to have Mirai infections, only 5,148 did not use this practice. IP churn is hence still an established technique, although its usage differs significantly by geographical regions. Table 2 shows the percentage of autonomous systems in a country that churn IP addresses, normalized by the total number of autonomous systems in that country that had Mirai infections and were therefore visible to us. As shown in the table, Poland has the least churning ASes, with only 5% of the observed ASes churning. In terms of IP churns however, more than a third of the IP space in this country churns, as the ASes that churn are magnitudes bigger than the non-churning ASes. The opposite is the case for Croatia, China, Ira n and Taiwan, as the percentage of churning ASes is much larger than the percentage of churning IPs, so only small ASes, or small parts of ASes churn. Germany, a country that has been identified by related work as a country with a high churn rate [18], has the largest IP churn rate, but not so high of an AS churn rate. Interestingly, in Brazil which has been identified by related work as a highly churning country [9], the number of churns in Brazil are 31% and 24% for IP and AS respectively, which does not rank it globally as one of the main users.

As discussed above, one of the motivations to allocate IP addresses only dynamically is to limit the amount of IPs that are “tied” up at a given moment, and thus potentially serve more customers with a smaller IP netblock. This could manifest itself in one of two ways: first, it could mean that we should over-proportionally see IP churn in smaller ASes that have a smaller allocation of IP addresses, or second, that IP churn is especially dominant in networks that have the bulk of their IP addresses in use and are hence short on resources. To test the first hypothesis, we analyzed the adoption of dynamic allocation with respect to the number of IP addresses belonging to an autonomous system, and found there is no relation between the size of an AS and its churn rate ($R = 0.03$, $p < 0.001$).

In order to obtain an estimation of the IP addresses that are actually in use within an autonomous system, we turned to the Internet surveys of Censys that scan all IPv4 addresses for a selection of well-known TCP and UDP ports as well as perform ICMP ping tests. For this analysis, we characterize an IP address as “active” if it has at least one TCP or UDP port open or respond to ICMP requests.

AS	ASN	Country	d	d^*	N	N^*	Flows	$f \leq d$	$f \leq d^*$	MAX
Orange	3215	France	24	168	7,913	122	78,259	73%	98%	6113
DTAG	3320	Germany	24	24	111,445	63	182,967	90%	90%	6466
Telefonica DE 1	13184	Germany	-	24	0	14	0	-	-	-
Telefonica DE 2	6805	Germany	24	24	8,482	17	15,242	99%	99%	1450
PJSC Rostelecom	8997	Russia	-	24	0	22	0	-	-	-
BT	2856	U.K.	-	337	3,813	67	28,748	-	79%	6364
Proximus	5432	Belgium	24	36	1,841	41	9,336	71%	79%	1418
A1 Telekom	8447	Austria	24	24	1,071	12	7,848	68%	68%	3506
Vodafone GmbH	3209	Germany	24	24	13,475	21	27,346	89%	89%	6562
Hrvatski	5391	Croatia	24	24	953	7	2,342	99%	99%	111
ISKON	13046	Croatia	-	24	28	6	89	-	97%	30
ANTEL	6057	Uruguay	12	12	64,856	6	94,430	96%	96%	2399
Global Village Telecom	18881	Brazil	48	48	104,244	6	250,393	99%	99%	3235
Mauritius Telecom	23889	Mauritius	24	24	4,981	6	11,433	93%	93%	252
JSC Kazakhtelecom	9198	Kazakhstan	24	24	1,616	15	11,103	80%	80%	3834
Orange Polska	5617	Poland	22	22	24,946	10	73,202	86%	86%	6355
VIPnet	31012	Croatia	-	92	91	7	236	-	100%	47
Digi Tavkozlesi	20845	Hungary	24	168	7,049	41	18,023	81%	99%	1243
Free SAS	12322	France	24	24	715	12	24,182	73%	73%	6392
SONATEL-AS	8346	Europe	24	24	1,792	7	5,090	93%	93%	231
Net by Net	12714	Russia	48	47	1,040	7	12,552	88%	87%	3123

Table 1: Measured churn times d compared to measured churn times d^* from [15] (in hours). N shows the total amount of churn events observed per AS, N^* shows the number of probes used in [15]. $f \leq d$ shows the percentage of flows with a lower duration than d and $f \leq d^*$ shows the duration of flows gathered in 2016 smaller than d^*. The largest flow duration in hours observed in an AS is given in *MAX*.

Country	IPs	ASes	Churning	
			IP	AS
PL	34352	567	37%	5%
US	69467	1192	30%	10%
UA	20305	653	12%	15%
RU	174237	1467	35%	15%
BG	23229	200	28%	16%
KR	77856	115	6%	30%
DE	69431	137	62%	31%
CN	1259021	150	20%	34%
IR	33309	139	19%	36%
TW	154880	55	13%	51%

Table 2: Selection of top and bottom five countries in terms of IP and AS churn. Similarly to other studies, we identify Germany as a highly churning AS [18], but we see that these churns actually originate from a smaller portion of ASes.

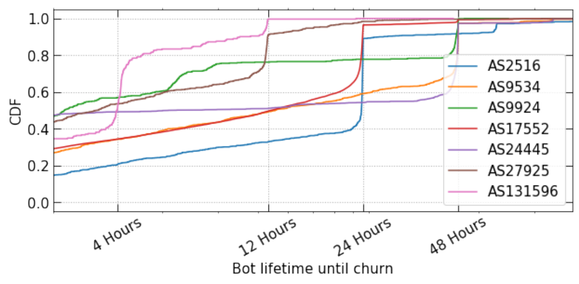

Figure 4: Churns for a selection of ASes, showing some ASes have multiple churn events.

While this provides a lower bound on the IP addresses in use, this is acceptable as an estimator here as we undersample everywhere and are only interested in relative differences between the autonomous systems. However, also for the second common hypothesis on why ASes would employ churning, we find no relation (R = 0.08, $p < 0.001$) between the fraction of the IP addresses used in an AS and the fraction of observed churns in the AS.

7.1.2 Churn rate of autonomous systems. Clients, home and enterprise routers may connect and disconnect to their Internet Service Provider at arbitrary times, for example a customer might close a laptop or unplug the router when no longer in use, similarly routers might reboot after a power outage or software crash, potentially obtaining a new public IP address from the ISP network. Aside from all of these "routine" activities, we are in this paper particularly interested in those moments when the Internet Service Provider breaks the connection and issues new IP addresses to its customers, which leads to predictable and large scale IP churn across an autonomous system. In order to automatically distinguish random population events from policy-based operator activities, we first quantify the lifetime of Mirai botnet infections at a given IP within a particular autonomous system, which is depicted in figure 4.

Consider for example the blue line belonging to AS2516, KDDI Corporation, a Japanese telecom operator. As we see in the graph, in about 40% of all cases where clients disconnect and reappear

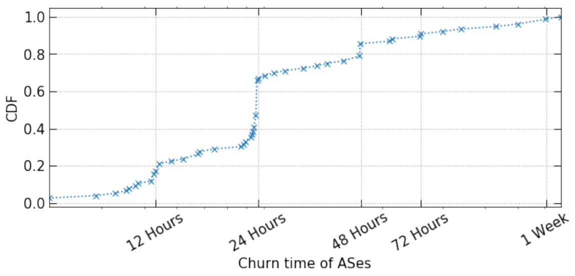

Figure 5: CDF of churn events for ASes. Most churn with a duration of 24 hours or a multiple of this.

on the Internet under a different IP address (but with the same ongoing Mirai infection) the client was online anywhere between seconds to almost one day. In the middle of the graph, we see a major spike: in approximately 45% of all situations where clients of KDDI disconnect and reappear with a new IP address on the Internet do so precisely in a 24 hour interval, which we identify due its consistent and network-wide application as an operator policy. Other networks apply different thresholds: AS27925 resets its connections after 12 hours, while AS9924 shows consistent re-connections after 2 days. We find that ASes occasionally operate different policies across the various parts of their networks. A good example of this is AS131596, in which most reconnects occur after exactly 4 hours, while some of its blocks churn in a 12 hour interval.

Using peak detection based on the relative growth of the churn CDF, we extract modes, or the churn interval(s) in use by a particular autonomous system. Figure 5 depicts the cumulative density function of these churn intervals for the churning 6,964 autonomous systems in this study. The most common policy found in the wild is a reset once a day, approximately 30% of autonomous systems disconnect their customers after 24 hours, to a lesser degree also for 2 days, 12 hours or 3 days. While these multiples of (half) day intervals seem to be in widespread use, in practice, we find many different and often highly unique policies across autonomous systems in the Internet. As can be seen by the dots and associated jumps in the CDF, we find churn intervals of every integer multiple of days, as well as integer multiples of hours as a dominant policy somewhere on the Internet. Overall, there is significant turnover, and on average, clients move to a new IP address after 33 hours.

7.2 Enterprise network blocks

While residential IP addresses are often allocated, for example with DHCP, this is not desirable for enterprise customers. To ensure continuity in IP addresses used in for example websites or other services, these customers are allocated static IP addresses, meaning that they are not prone to the churn such as residential hosts. To do so, Internet Service Providers (ISPs) allocate blocks of IP addresses for their enterprise customers, and do not use these IP addresses in the dynamic pool. Insight into how these blocks are allocated inside an AS yields information on which netblocks do churn, and in which netblocks IP addresses are static.

By aggregating churn rates of individual hosts in their /24 netblock, as enterprise networks are likely to be managed at /24 [2],

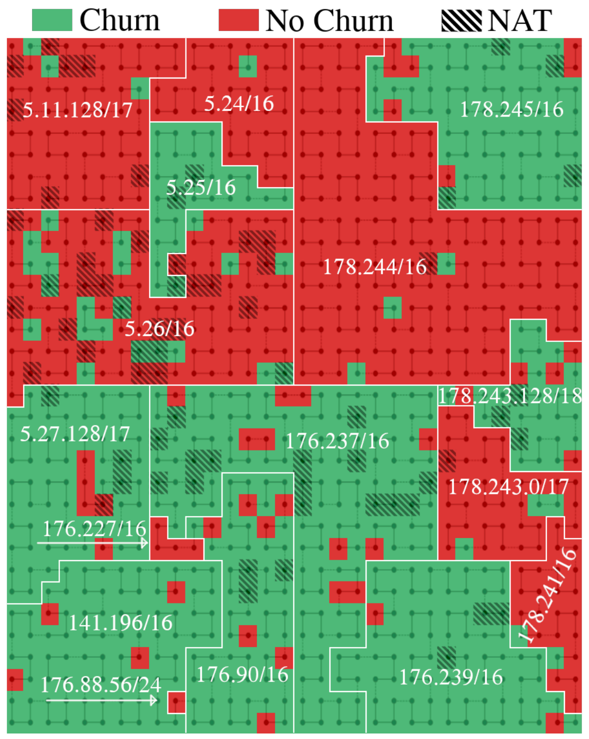

Figure 6: Hilbert curve of churn in netblocks of AS16135 per /24 netblock. Blocks are labeled with churn if there has been one or more churn events in the /24 over the duration of the whole study. The graph shows blocks of non-churning ranges, where enterprise customers are located. The Hilbert curve is drawn in the figure with a dotted line.

and labeling these netblocks as churning when we observe churn in this netblock, we can plot a Hilbert curve that accurately shows the allocation of netblocks in an AS. A Hilbert curve maps a 1-dimensional array into two dimensions, and preserves for a large part the distance information, where adjacent points on the curve are also adjacent in the 1D array. Figure 6 shows this Hilbert curve for AS16135 across a number of its prefixes separated by white lines, located in Turkey owning 197 IPv4 prefixes. In the figure, we only show a selection of these prefixes for visibility. Every point in the Hilbert curve is mapped to one /24, which we have observed in our data. We omit /24 netblocks that we have not observed. The figure shows that this particular AS allocates their network based on prefixes, and places their enterprise and residential customers in different prefixes. The allocation does not seem to be dependent on the prefix, as the 5.25/16 prefix is clearly allocated to non-enterprise customers, but all adjacent prefixes are not.

While some networks segregate their network such as in figure 6, not all operators allocate their network based on BGP prefixes. In AS3320, where 10% of flows exceed the churn time as shown in table 1 and thus are most likely located in enterprise netblocks, we do not see the clear distinction between netblocks as in AS16135 but rather enterprise blocks and residential IPs scattered through

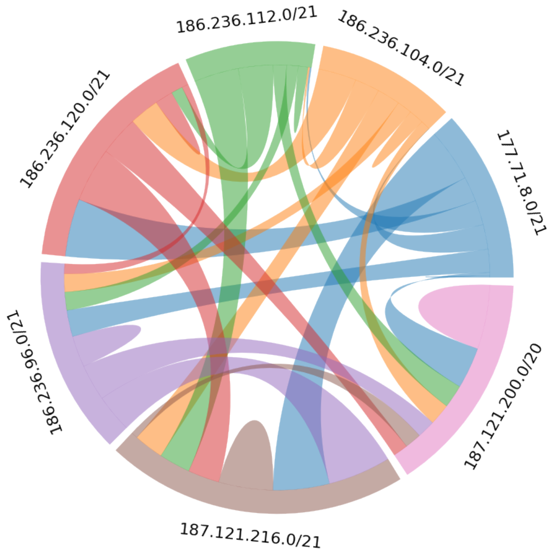

Figure 7: Churn behavior between netblocks in AS53131. Outgoing edges are colored by the originating netblock. Some netblocks, like the blue one, act at “sources”, churning to other netblocks but never receiving from other blocks. Others act as “sinks”, only receiving and churning within.

eachother, showing the difference in ISP policies for segregating their networks.

7.3 Prefix allocation for churning clients

When customers dynamically get assigned an IP address from a pool of IP addresses, the obvious question to ask is how this allocation strategy works. Imagine a larger network with multiple netblocks that were added over time and are therefore not consecutive in the IP address space. One way to realize this is to assign a pool of IP addresses to a particular network device that provides access to customers connecting through it. This would mean that IP address allocations are stable within netblocks, and can be associated to customer groups based on region or the way they connect to the network (DSL, fiber, cable, vs. dial-up). Such a semi-fixed strategy is however challenging to run, if the goal is to reduce address usage and thus conservatively assign netblocks while providing enough addresses during connection surges. An alternative strategy is to have netblocks “float” across devices handing them out to connecting customers, which provide the best utilization but is more difficult to do routing for.

By tracing over time at which IP addresses returning customers reconnect to the Internet – which we can identify as they still run the same Mirai infection –, we can obtain some insight into how the address pools are managed by the autonomous systems. Although some address mobility is expected also in the first scenario if a customer for example goes on business travel and connects with the same device and account from a different part of the network, if these events happen en masse across an autonomous system we can link this behavior clearly to the second strategy in use. Figure 7 shows the movements of clients between seven different

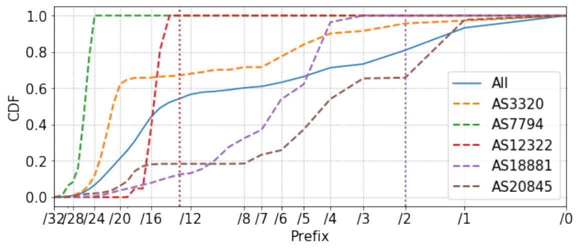

Figure 8: CDF of prefix change due to churn, vertical lines show the maximum jump of AS12322 and AS18881, the others can make jumps within /0. Different operator policies can be observed, with for example AS7794 only changing inside a /24 prefix.

/21 netblock in the network of AS53131, where the outgoing edges are filled in with the color of the netblock they originated from. As we can clearly see, for this particular operator there appears no clear relationship between clients and IP pools they get addresses from, but allocations jump across ranges.

While previous work on IP churn has also found common prefix changes of Internet endpoints [15], the necessity to infer this by active measurements limited the number of autonomous systems that could be investigated. In [13] for example, the authors found dynamic network-wide pools across four network operators, while [15] was able to test from 3,038 RIPE Atlas probes a total 929 ASes. The passive technique presented in this paper allows us to do such an investigation at much broader scale, and figure 8 shows the distance between consecutive IP address assignments for endpoints across more than 12,000 autonomous systems that shows the multitude of different policies across the Internet. A very clear example of the first allocation strategy is AS7794, Execulink Telecom from Canada, which always assigns IP addresses from the same /24 netblock – customers reappearing with a new IP address never jump outside of the block they connected from previously. This is not an artifact of operator size but operator policy, as AS7794 has a total of 136,192 IP addresses assigned to them. In the autonomous system AS12322, half of all new address assignments occur within a distance of /16, even though the operator has many netblocks that are both larger and spaced apart at a distance larger than this. When we look at IP address assignment across all autonomous systems, we see allocation to be mainly driven by custom, per-AS settings and no universally applied practices appear across autonomous systems. This can for example been seen by the complete absence of peaks for the blue line in figure 8 representing all churn distances, which is void of any noticeable peaks at typical blocks used for netblock allocations and routing such as /24 or /16.

7.4 NAT behavior

As discussed before, due to the shortage and resulting high value of IP addresses, in many operator networks clients are no longer connected with a publicly routable IP address to the Internet, but placed behind a (carrier-grade) network address translation (NAT) where a large number of hosts share the same public IP address. This practice will hence lead to an underestimation of phenomena such as botnet infections on the Internet, if prevalence is assessed based on IP address as a proxy. As the Mirai botnet binds a local port such as TCP23 on the host it has infected, it is not possible to have two concurrent infections on the same physical machine.[1] This however means that if we see Mirai scanning traffic with different session configurations originating from the same public IP address, we have clear evidence of the existence of at least two hosts behind the IP in a network address translation configuration. In this section, we will therefore use our methodology to quantify NAT deployments.

We find 9370 ASes containing IP addresses having concurrent infections, and therefore most likely being part of a NAT. Figure 6 shows NAT allocation on AS16135, with NATs spread out over various subnets. While we have observed many NATs in some subnets such as “5.26.x.x”, other subnets do not contain any sign of NATs being used.

NATs are used to connect a large amount of devices to the Internet, instead of limiting the total users on the Internet to the total IPv4 space. Intuitively, one would expect a NAT in a small AS, where the operators do not have enough IP addresses to cope with the number of devices that needs to connect to the Internet. To verify this hypothesis, we compare the total number of IP addresses allocated to an AS, to the number of NATs we have observed normalized by the total number of observed IP addresses at an AS, and find there to be a slight negative correlation ($R = -0.11$, $p < .00001$). So indeed, smaller NATs are more likely to have NATs in their network, while larger networks can do without.

Even in a NAT, network operators churn the IP addresses placed behind an IP address. As more devices, and therefore infections, can be located behind the same publicly routable IP address, infections in one NAT can be located in multiple other NATs after an operator churns the network as in the example of figure 3. We observe this behavior in 289 ASes and from these can identify 104 ASes where network operators churn IP addresses in a NAT-pool, where devices churn within a small pool of IP addresses.

7.5 Botnet estimation

Both churning IP addresses and the use of NATs have an effect on Internet based measurements where IP addresses are considered immutable single-source datapoints. While both effect the measurements, churn leads to an overestimation, while NATs lead to an underestimation in the measurements. In this section we will evaluate the effect of IP churn and NATs on measuring the Mirai botnet used in this study.

We have observed over 4 million churns over the 9 months of data collection, which would imply an overestimation of 20% when counting IP addresses infected by Mirai. As we have shown in this paper, ASes have different policies for when their network is churned, and thus how much we overestimate the size of a botnet in that AS. The largest overestimation we observe is in AS48738, which churns every 6 hours leading to an overestimation of 409%.

[1]There are corner cases to this rule, as there exist a Mirai descendant that bind not to telnet at the default port. However, these cases are rare and the effect negligible when doing quantifications across ASes.

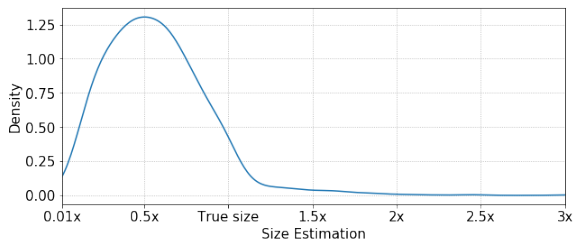

Figure 9: PDF of the estimation of botnet size per AS when using IP addresses as unique bot count.

NATs lead to an underestimation of the total Mirai spread when solely counting IP addresses, as multiple infections are located behind the same publicly routable IP address. In our data, only looking at IP addresses will lead to an estimation of only 55% of the total number of infections. As smaller networks are more inclined to use NATs in their network, the underestimation in small ASes tend to be larger than in ASes having access to many IP addresses. The largest underestimation is for AS29914, a small AS with 15,616 IP addresses, and all the infections located behind a single NAT. So we would only count this as 1 infection, while in reality we observe 287 different infections behind this NAT, meaning we largely underestimate the number infections.

Figure 9 shows the PDF of infection size estimations of 2200 ASes divided by the actual size of the infection on the AS. We see that most infections are underestimated, due to the presence of a NAT, and that only few are overestimated due to churn. We see that NATs have a larger effect on the estimation than churns, as NATs can easily host a large number of devices while the churn rate is not high enough to compensate for this underestimation. Only 27 ASes were correctly estimated, of which 21 were void of churns and NATs, showing that IP addresses cannot be used as an effective surrogate to quantify behavior and characteristics of hosts.

8 CONCLUSION

This paper presents a method to quantify IP allocation behavior by leveraging the spread of a large botnet. The proposed method is able to distinguish in detail different subnets used by ASes for different purposes. While previous work has relied on active measurements to measure churn rates on large scale, we are able to passively measure over 12,000 ASes.

We show we are able to measure churn rates in different networks, by using only passive data collected in a network telescope. Using these measurements, we show that network operators segment their network in different ways, where some operators choose to divide their network based on BGP prefixes, other operators segment also within these prefixes. Additionally, we show that we can identify addresses used in (carrier-grade) network address translation, and are able to quantify the differences in over- and underestimation for IP-based measurements in different ASes.

REFERENCES

[1] Norbert Blenn, Vincent Ghiëtte, and Christian Doerr. 2017. Quantifying the Spectrum of Denial-of-Service Attacks through Internet Backscatter. In *Proceedings of the 12th International Conference on Availability, Reliability and Security.*

[2] Xue Cai and John Heidemann. 2010. Understanding block-level address usage in the visible internet. In *ACM SIGCOMM*.

[3] Martin Casado and Michael J Freedman. 2007. Peering through the shroud: The effect of edge opacity on IP-based client identification. In *NSDI*.

[4] David Dagon, Cliff Changchun Zou, and Wenke Lee. 2006. Modeling Botnet Propagation Using Time Zones. In *Network and Distributed System Security Symposium*.

[5] Michael Dooley and Timothy Rooney. 2013. *IPv6 Deployment and Management*. Vol. 22. John Wiley & Sons.

[6] Vincent Ghiette and Christian Doerr. 2018. How Media Reports Trigger Copycats: An Analysis of the Brewing of the Largest Packet Storm to Date. In *ACM SIGCOMM Workshop on Traffic Measurements for Cybersecurity (WTMC)*.

[7] John Heidemann, Yuri Pradkin, Ramesh Govindan, Christos Papadopoulos, Genevieve Bartlett, and Joseph Bannister. 2008. Census and survey of the visible internet. In *ACM SIGCOMM Conference on Internet Measurement*.

[8] Dominik Herrmann. 2016. *Beobachtungsmöglichkeiten im Domain Name System. Angriffe auf die Privatsphäre und Techniken zum Selbstdatenschutz*. Springer.

[9] Stephen Herwig, Katura Harvey, George Hughey, Richard Roberts, and Dave Levin. 2019. Measurement and Analysis of Hajime, a Peer-to-peer IoT Botnet.. In *Network and Distributed System Security Symposium*.

[10] Manas Khadilkar, Nick Feamster, Matt Sanders, and Russ Clark. 2007. Usage-based DHCP lease time optimization. In *ACM SIGCOMM conference on Internet measurement*.

[11] Marc Kührer, Thomas Hupperich, Jonas Bushart, Christian Rossow, and Thorsten Holz. 2015. Going wild: Large-scale classification of open DNS resolvers. In *ACM SIGCOMM Conference on Internet Measurement*.

[12] Gregor Maier, Anja Feldmann, Vern Paxson, and Mark Allman. 2009. On dominant characteristics of residential broadband internet traffic. In *ACM SIGCOMM Conference on Internet Measurement*.

[13] Giovane CM Moura, Carlos Ganán, Qasim Lone, Payam Poursaied, Hadi Asghari, and Michel van Eeten. 2015. How dynamic is the isps address space? towards internet-wide dhcp churn estimation. In *IFIP Networking*.

[14] William B. Norton. 2014. *The Internet Peering Handbook*. DrPeering Press.

[15] Ramakrishna Padmanabhan, Amogh Dhamdhere, Emile Aben, kc claffy, and Neil Spring. 2016. Reasons dynamic addresses change. In *ACM SIGCOMM Conference on Internet Measurement*.

[16] Ioannis Papapanagiotou, Erich M. Nahum, and Vasileios Pappas. 2012. Configuring DHCP leases in the smartphone era. In *ACM SIGCOMM Conference on Internet Measurement*.

[17] Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis. 2007. My botnet is bigger than yours (maybe, better than yours): why size estimates remain challenging. In *USENIX Workshop on Hot Topics in Understanding Botnets*.

[18] 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*.

[19] Long Vu, Deepak Turaga, and Srinivasan Parthasarathy. 2014. Impact of DHCP churn on network characterization. *ACM SIGMETRICS Performance Evaluation Review* 42, 1 (2014).

[20] Yinglian Xie, Fang Yu, Kannan Achan, Eliot Gillum, Moises Goldszmidt, and Ted Wobber. 2007. How dynamic are IP addresses?. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*.