Examining Mirai's Battle over the Internet of Things

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

Using hundreds of thousands of compromised IoT devices, the Mirai botnet emerged in late 2016 as a game changing threat actor, capable of temporarily taking down major Internet service providers and Internet infrastructure. Since then, dozens of variants of IoT-based botnets have sprung up, and in today's Internet distributed denial-of-service attacks from IoT devices have become a major attack vector. This proliferation was significantly driven by the public distribution of the Mirai source code, which other actors used to create their own, customized version of the original Mirai botnet.

In this paper we provide a comprehensive view into the ongoing battle over the Internet of Things fought by Mirai and its many siblings. Using 7,500 IoT honeypots, we show that we can use 300,000,000 compromisation attempts from infected IoT devices as well as a design flaw in Mirai's random number generator to obtain insights into Mirai infections worldwide. We find that networks and the particular malware strains that plague them are tightly connected, and malware authors over time take over strategies from their competitors. The most surprising finding is that epidemiologically, IoT botnets are not self-sustaining: were it not for continuous pushes from bootstrapping, Mirai and its variants would die out.

CCS CONCEPTS

• **Security and privacy** → *Network security*; *Malware and its mitigation.*

KEYWORDS

Mirai, botnet, IoT, cyber threat intelligence

ACM Reference Format:
Harm Griffioen and Christian Doerr. 2020. Examining Mirai's Battle over the Internet of Things. In *2020 ACM SIGSAC Conference on Computer and Communications Security (CCS '20), November 9–13, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3372297.3417277

1 INTRODUCTION

The emergence of the Mirai botnet in late 2016 fundamentally changed the Internet threat landscape. Although the risk of insufficiently protected Internet-of-Things (IoT) devices was long known, the problem made the front pages when its initial distributed denial-of-service (DDoS) attacks exceeded volumes of 600 Gbps, crippling major Internet infrastructure and service providers such as OVH [2] or Dyn [1]. The first attack by an IoT botnet doubled previous attack volumes from the get go, and in the coming weeks would continue to increase to break the 1 Tbps threshold [8].

This did not only capture the attention of the general public and defenders, but also the imagination of perpetrators. Soon after, the source code of Mirai was published online [5], leading copycats to release clones of Mirai-based IoT malware. Named "MIORI", "JOSHO", or "MASUTA", these variants are directly derived from the Mirai source code, copying the core framework and essential routines for scanning, infection and communication, but also feature actor-specific modifications to passwords, the way the malware identifies itself, and the C&C servers bots report back to.

Trivial access of the source and the abundance of vulnerable devices has led to a plethora of Mirai variants, which are fighting over control over the millions of IoT devices deployed worldwide. In this paper, we are examining this battle between Mirai and its siblings that have emerged since. To do this, we leverage the fact that the Mirai botnet behaves like a worm, using an infected IoT device to scan the Internet for other devices it could potentially compromise. With an installation of 7,500 IoT honeypots, we are collecting these infection attempts, and can turn 300,000,000 received compromisation attempts into a real-time view which devices around the world are infected at a given moment with a particular strain.

One of the core features across Mirai and all its variants is how it searches for victims. To avoid sequential port scanning that is trivially detected, the malware selects targets based on randomly generated destination IP addresses, and for which the authors designed their own random number generator (RNG). As we will show, this design has severe flaws, allowing us to break the seed based on information from a *single* incoming packet. This insight can then be used to learn about the lifetime of the infection on the device itself. With this work, we are making the following contributions:

- We are the first to provide a comprehensive view into the infection and reinfection behavior of IoT devices. We demonstrate that it is possible to exploit structural weaknesses in Mirai's RNG to extract the precise moment of compromisation, which we use to understand the lifetime of infections.
- We show that IoT devices are under intense attack by a plethora of actors. We follow the activities of 39 variants and analyze how they infect and retain control over some 200,000 unsecured devices. We find that compromised hosts frequently change "ownership" through reboots and reinfections, but also based on hostile takeovers by other actors.
- We show that IoT malware crashes comparatively quickly on routers, and there are clear differences between particular malware strains and the network devices are placed in.

CCS '20, November 9–13, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7089-9/20/11…$15.00
https://doi.org/10.1145/3372297.3417277

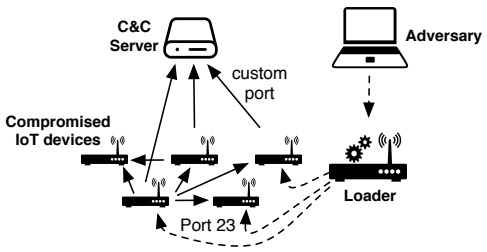


Figure 1: Adversaries bootstrap using loaders, targeting a predefined list of devices. Compromised hosts report to a C&C and randomly target IP addresses to proliferate.

- We provide the first epidemiological quantification of Mirai, and show that epidemiologically speaking, Mirai is not a self-sustaining worm and would die out if it would not be for the bootstrapping infrastructure that constantly spreads reinfections to vulnerable devices.
- We find evidence of specialization of malware authors towards specific victim groups. Strains use information about devices in particular networks to gain a competitive advantage, yet malware owners observe their competition and adapt their password lists with information from their rivals.
- We demonstrate that adversaries are making deliberate efforts to evade network ranges and thus avoid detection by common measurement initiatives.

2 THE MIRAI BOTNET

While attacks on IoT devices have become common place, the advent of Mirai as the first major IoT malware was a milestone in Internet security. Japanese for "future", the IoT malware became front-page news when its attacks severely impacted major Internet infrastructure and service providers in DDoS attacks. To understand how Mirai operates, we begin with an overview of the system infrastructure and the way it targets victims.

2.1 System Infrastructure

The ecosystem of the Mirai botnet consists of three main components as shown in figure 1: a loader to bootstrap the botnet, the compromised routers themselves, and the command-and-control (C&C) server to which bots report back and obtain instructions from. Connections to the C&C are made using a plain TCP socket, on an address and port hardcoded in the source. The control protocol is very simple, with instructions in binary status codes. Core of the botnet are the compromised IoT devices, which are responsible for spreading the infection further. To this end, they independently scan the Internet for devices responding on specific ports. When configured to search for telnet, they target ports 23 and 2323.

As adversaries would avoid to do the first initial scan and compromisation activities from their own IP address, the Mirai toolchain includes a loader mechanism, which is provided with a list of devices, for example powerful hosts with a high-bandwidth network connection. These devices are then used to start the scanning and infection activities, thereby bootstrapping the Mirai installation.

2.2 Mirai Behavior

Certain implementation choices of Mirai create unique behavioral characteristics that can be efficiently fingerprinted. At a high level, the Mirai IoT malware pursues three different objectives after the infection of a device: first, it closes the entry point (typically telnet on TCP port 23, which is also the focus of this paper) in order to gain exclusivity over the compromised device. Second, the malware will start a continuous scan for other devices and if it identifies an open telnet port it begins to brute force based on a built-in credential list. And third, it establishes a connection with the C&C server hardcoded in the source and executes commands it receives. Figure 2 shows an excerpt of the Mirai source, which we will discuss in the following, with these three tasks spawned in `main()` (see marker ① in figure 2).

When targeting telnet, Mirai will attempt to bind to port 23. If this is not possible, it tries to shut down the other telnet process through which other logins could take place. In `killer_init()`, it identifies and kills the process that has bound to this port ②, thereby locking the router down for the duration of the infection. Mirai and its descendants are however non-persistent, so after a reboot the device is restored to its pre-infected, vulnerable state. It will spin up its original telnet process, and is ready for reinfection.

As one of its first actions, the Mirai malware forks off a separate thread that performs an Internet-wide scan for other IoT devices. As shown in `scanner_init()`, it begins by re-initializing the random number generator (RNG) ③ and choosing a random source port in the range of [1024, 65535] ④. It then prepares the IP and TCP header it will use for subsequent scan packets. In an infinite loop, it chooses random IP addresses as target (`get_random_ip()`) ⑤, sets a random IP ID for the packet and chooses port 23 or 2323 as port ⑥. Every IP is probed 10 times before continuing. In case of a response, Mirai will start to brute force the login using a local dictionary.

Target Selection and Randomness Generation. A number of packet-specific information in the TCP SYN scans sent by Mirai are populated based on output from the RNG, most importantly the destination IP address, source port, and IP ID. This RNG is seeded at startup and again at the fork of the thread with the current time in epochs, the process IDs of the `main()` and the scanning thread, and the number of clock ticks since the current process has been started (which resets in the scanning thread due to the fork). These functions return 32, 15, 15, and 32 bit values respectively, and are used to initialize register variables x, y, z and w used in each iteration. Although this would naively imply 94 bits of entropy in the RNG seed, in practice most bits are entirely predictable.

The RNG itself follows a linear shift feedback register design (LFSR), with four XOR operations and three bit shift operations. The inner workings are depicted in figure 3, which shows the state initialization and the generation routine. After the initialization of (x, y, z, w) with environment variables, in each iteration, the left and the right most internal state variables are repeatedly combined with bit-shifts and XOR operations to compute the value of the internal state variable w, which is at the same time the generated random number. Note that the generated randomness is directly taken from the internal state without any output transformation, and thus leaks the state information to the outside. The content of

```
// main.c ①
int main(int argc, char **args) {
   …
   rand_init();
   …
   scanner_init();
   …
   killer_init(); ②
   …
   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();                ③
   fake_time = time(NULL);
   …
   do {
      source_port = rand_next() & 0xffff; ④
   } 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(); ⑤
         iph->check = 0;
         iph->check = checksum_generic((uint16_t *)iph,
            sizeof (struct iphdr));

         if (i % 10 == 0) {
            tcph->dest = htons(2323);
         }else{                        ⑥
            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 2: Selection of the Mirai source code, responsible for the creation of scanning packets [4]

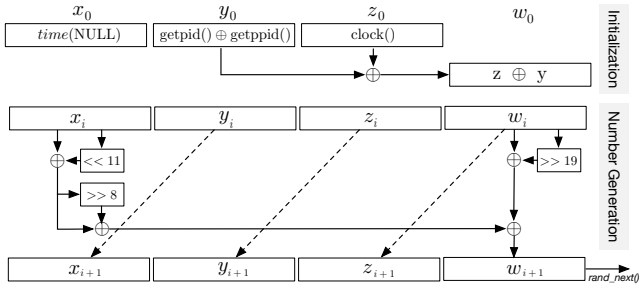


Figure 3: RNG initialization and generation.

the remaining three state variables is shifted by one field to the left, thus the next random number directly depends on the last random number generated as well as the one from three iterations ago.

While `rand_next()` generates a 32-bit value, its output is truncated when used for the source port and IP ID to the lower 16 bits. Mirai also rejects values lower than 1024 when setting the source port to avoid privileged ports, as well as when the generated target is part of specific IP ranges the malware author chose to skip. The excerpt shows that in case of an IP address on the local loopback address (127.0.0.0/8), the 15.0.0.0/8 and 16.0.0.0/8 net blocks a new value is drawn. These lists are hardcoded in the source, figure 2 shows a truncated list for readability.

3 RELATED WORK

Over the past two decades, security researchers have tried to identify, analyze and mitigate botnets. One of the first of its kind was Cooke et al. [16], who demonstrated that the activity of botnets may be captured and analyzed by deploying honeypots. While research has studied botnet identification and mitigation, limited focus has been put into analyzing why some botnets are more successful than others, and how the increase in botnets has prompted botmasters to compete with each other for control of devices. The relative ease to build botnets and the disruptive power they possess rallied researchers to create detection and mitigation methods for this threat. Although many methods have been proposed [11, 22, 27, 32, 41, 42], the continuous evolution of botnets remains a constant challenge for the disruption of these networks [9, 23, 30, 36, 38].

One such evolution is the use of poorly configured Internet-of-Things (IoT) devices to build a botnet, which has since gained a lot of attention of the security community [26, 29, 31, 34, 35, 45]. When the source code of the IoT-based botnet Mirai was publicly shared [5], new actors could bootstrap their own botnet with ease. Antonakakis et al. [8] gave the first comprehensive understanding of this botnet and the effect of the shared source code, describing that new, specialized variants spawned off from Mirai. We build on this large foundation of studies to show the impact of these evolutions, and how botmasters evolve to trump their competition.

While the source code leak of Mirai enabled many actors to join the IoT game, the raised awareness of these new threats prompted device owners to protect their devices [15]. As a result, the share of devices available to actors has decreased. As the scale of a botnet is important for its monetization [13], by for example using the computation power of infected devices for cryptocurrency mining

[10], botmasters have found an increasing number of ways to infect devices. While new infection vectors have been added to also the descendants of Mirai [8], the most commonly probed service is still the telnet port used in the original version of Mirai [24]. The impact of these evolutions on the size and geographic distribution of botnets has yet to be evaluated in the scientific community.

Several works have analyzed the structure and infrastructure behind the original Mirai IoT botnet. [26] show the communication pattern between Mirai bots and loader, and show how Hajime has evolved beyond the original Mirai. [44] performed a forensic analysis of the original Mirai botnet, among others on encoded user credentials. The authors call upon further research to focus on the customizations of Mirai to find these artifacts in them as well. [29] has provided an overview of a handful of Mirai variants, among them Hajime, Persirai and Brickerbot, show how these variants are used and how they have changed their infection behavior from the original Mirai botnet. Furthermore, [17] found there to be many different password combinations in use by various IoT malwares, [19] showed a link between password lists and used tooling during brute forcing. Similar to the other related work, these papers have also not identified the differences in success between botnets.

Other IoT botnets have also been examined by the security community, with [25] analyzing the Hajime botnet in-depth, [40] surveying the BlackIoT botnet and many reports from industry [20, 33, 37, 39]. Similar to the works on the original Mirai botnet, these papers focus on one botnet and do not analyze the effect of multiple botnets competing for the same IoT devices.

In this work, we go beyond the state of the art by answering two major questions that were put up but remained unanswered in previous work. First, what is the interaction and competitive behavior of the plethora of IoT and specifically Mirai malware strains in the wild? And second, which factors determine how successful a particular malware can spread? We extend the work of Antonakakis et al. [8] and in this paper additionally investigate the evolution of Mirai-based botnets and how these adaptations relate to the regions they infect. We do this by introducing a novel way to identify the exact moment a device was compromized, a technique that we also use to track the lifetime of an infection, which allows us to accurately trace the infection characteristics of Mirai and its descendants.

4 DATASET

The analysis in this paper is made possible through the combination of three datasets listed in table 1, each capturing a different aspect of the world-wide phenomenon of an IoT botnet. First, we use a large network telescope to capture the scanning behavior at scale of infected Mirai instances looking for other vulnerable hosts. Second, we operate an installation of 7,500 honeypots to determine which particular strain an IoT device is infected with. And third, we use operator netflows to assess the behavior of the botnet at large. We will describe each of these datasets in more detail in the following.

Telescope. As soon as a device is infected by a Mirai variant, it immediately starts scanning the Internet for other vulnerable hosts. This scanning routine is based on specifically crafted and injected packets, and thus displays some implementation particularities. For instance, the TCP sequence number will always be set to the destination IP, and on startup a Mirai instance picks random values for window size and source port it will use for every packet until the infection is removed. This allows us to link incoming packets to those potentially sent by Mirai.

Dataset	Size (Jan-Mar 18)	Purpose
Telescope	1.2 TB	Infected devices, RNG analysis
Honeypots	213 GB	Variant+behavior identification, credentials, staging servers
Netflows	569 GB	Verification and coverage analysis, blacklisting analysis

Table 1: Datasets used in this study.

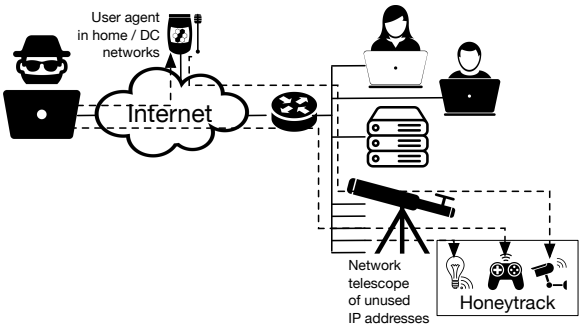


Figure 4: The Honeytrack system routes compromization requests into a separate, virtualized environment.

In order to confidently arrive at an estimate how many and which devices are potentially infected by Mirai, we use a large network telescope consisting of three partially populated /16 networks with ~65,000 IP addresses. As dark IP addresses, these addresses will only receive scans and Internet backscatter, which can be easily separated based on the TCP SYN or SYN+ACK flags [12]. As Mirai's target selection is random, the large size of the telescope – we monitor about one in 65,000 IP addresses on the Internet – will allow us to quantify and track how long infections are active.

The Honeytrack IoT system. While the telescope provides a basis for counting and tracking a particular infection based on header values, it does not reveal which malware strain has currently infected a device. For this, we developed a distributed honeypot system, consisting of an endpoint agent and a backend environment visualized in figure 4. The agent listens on TCP port 23 and transparently tunnels incoming packets to an environment where IoT devices are emulated. Incoming requests are randomly allocated to one of eight system images (based on the most common banners in a Censys telnet survey), exposing a fully functional (high engagement) but virtualized IoT device. Once a connection is made, the same IP will always be be connected to the same device again, which is subsequently serialized to preserve state if the attacker would be coming back.

Over a period of 6 weeks, 7,500 honeypots were deployed in 3 home ISP networks, 2 network ranges of a public cloud, as well as three network ranges of an enterprise network where they were

interleaved on unused IPs between active systems, so that the honeypots would blend in. For this permission of the network owners was sought via the established approval mechanisms. The installation recorded a total of 300,000,000 login attempts from 203,920 unique sources, which during the brute forcing would reveal their hard-coded password list as well as the malware strain. This is possible as the Mirai botnet does not immediately test whether it has successfully logged in with one of the passwords it has sent. Instead, the malware sends a username/password combination followed by a set of commands, the last one being */bin/busybox MIRAI* to the tested device. This has two purposes. If the username/password combination was correct, the command string gets executed. If none of the credentials worked, the busybox command is just another username. If the credentials were correct and the command executed, a router running busybox would respond with "MIRAI: applet not found", while a host without would throw an error. Mirai thus knows from the response whether a credential was correct and the system runs busybox. When later botmasters recycled Mirai's source code to change their own IoT variants, they changed the busybox string. By providing a login and a working shell environment, we can thus track with Honeytrack which variant attacks, and whether it was modified to contain different or new credentials.

To assess the role of Mirai and its variants in the entire spectrum of actors targeting telnet and the IoT, we tracked all source IPs that scanned or attempted login over the entire observation period. 87% of all source IPs exhibited Mirai's special way of crafting packets, thus making this particular IoT malware family the dominant player and most relevant phenomena in the IoT malware arena. For this reason, we have made Mirai and its variants the focus of the discussion in this paper.

Netflows. The above two datasets provide a view which malware strains are currently deployed at which IoT device around the world. To also understand what these devices are actually doing and how the support infrastructure works we also make use of netflows from a Tier 1 network operator. To preserve the privacy of the end users, the source and destination IP addresses are masked using the prefix-preserving anonymization scheme presented in [43], proven to be semantically secure. This AES masking key is only known to the operator, which does provide us with a mapping of IP addresses that attacked us to their anonymized counterparts. The data access procedure was cleared with the corresponding protection authority of the operator. This setup protects the identity of particular clients, but at the same time does allow us to understand in which networks and countries particular variants are raging.

5 ATTACKING A LOW ENTROPY PRNG

Although the PRNG of Mirai is relatively simple, it creates good quality output. The cycle length has a period of $2^{96} - 1$ and the output passes the Dieharder test suite [7, 14]. Still, LFSRs are not equivalent to cryptographic RNGs and inherently vulnerable, and the lack of a suitable output transformation effectively leaks the internal state via the generated randomness. If enough consecutive bits become observable, it allows the prediction of future results.

The LFSR used within Mirai depends on four internal state-keeping variables, which means that if we were to learn four consecutive random values we could predict all future ones. Observing four complete consecutive values is however impossible, since the `rand_next()` function is used not only in the packet generation but also elsewhere, and the 32-bit output is truncated to 16-bits when setting the IP ID or source port. If we were to obtain two Mirai scan packets back-to-back, we would thus learn twice 32 bit RNG output from the destination IP and twice 16 bits from the IP ID. However, as IP addresses are hit randomly, it is unlikely that two consecutive packets hit our telescope or honeypots. Still, as two subsequent calls to `rand_next()` are used in the creation of a single packet, we can efficiently confirm a correct state guess as the likelihood to obtain a 32-bit and a 16-bit match is vanishingly low.

Attacking the seed. Any PRNG is only as good as the entropy it has been seeded with. Even a cryptographically-strong PRNG can be broken when initialized with (largely) predictable data. Such insufficient entropy is another problem in Mirai's home-grown design. Upon startup of the thread, the router fills the internal state variables with four values: (1) the epoch in seconds, (2) the process ID, (3) the parent process ID, and (4) the number of clock ticks since the program was launched. In theory, the combination of these four values could provide 94 bits of entropy on a 32 bit system. In practice however, the entropy of the random numbers is lower due to a number of conceptual and implementation mistakes:

- `time(NULL)` returns the time in seconds since January 1, 1970. As Mirai immediately starts to send packets at a high rate, the startup time will be very close to the time of the first arriving packets, especially in a telescope. Even if we very conservatively assume that the device has been infected for 24h before we see the first packet, this would imply a mere 16 bits of entropy. Our measurements have shown this to be much shorter in practice, with the first attack packets reaching us on average within 15 min after startup.
- Before the RNG is initialized, the process is forked to create a dedicated thread for scanning. As a fork is essentially the start of a new program, the number of clock ticks elapsed starts at 0 for the child process. By the time the RNG is seeded, only four instructions have been executed. This reduces 32 bits of potential entropy to just a handful of bits. In glibc prior to version 2.18 which is frequently used in IoT devices, the resolution of `clock()` was limited to a granularity of 10,000 ticks. Thus, possible values at this stage in the program are either 0 or 10,000, reducing the entropy to just 1 bit.
- The state y is set to the process ID XORed with the parent process ID. While both process IDs are 15 bits, the combination of both also has 15 bits of entropy.

Due to these issues the 94-bit seed has in practice only 32 bits of entropy. We can analytically derive which bits in the four registers actually contain entropy, however given that LFSRs can be efficiently computed, it is trivial to brute force the remaining 32 bits and evaluate the sequence to find a value pair where the 16 bits used in the source port and window size matches the output.

As the source port and window size only provide the last 16 bits generated, multiple seeds in our search space will generate these numbers in correct succession. To verify which of the matching combinations is the actual seed, we can generate the sequence of numbers and verify if this matches packets coming into the telescope. In these sequences we can identify whether an IP ID and

	Variant	Hosts		Variant	Hosts
1	MIORI	75,249	6	MASUTA	5,338
2	MIRAI	62,235	7	NGRLS	5,113
3	JOSHO	23,487	8	SORA	4,631
4	daddyl33t	12,583	9	RBGLZ	4,076
5	Cult	5,621	10	OWARI	2,201

Table 2: Unique hosts for the top 10 advertised botnets.

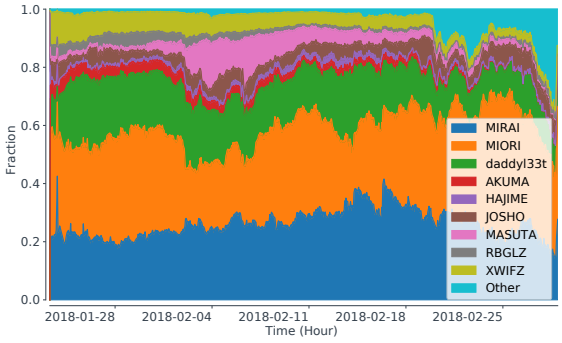


Figure 5: Marketshare of advertised variants.

destination IP address are generated consecutively, and repeat the process until we have only one candidate seed left which is the right seed. In most cases, we only have to generate the sequence until the first IP address in order to find the correct seed. Generating the sequence until the first observed IP address is a process that will terminate for the correct seed, but for incorrect seeds it will run in the worst case indefinitely. The number of steps we have to take depends on the statistical likelihood to be hit by a Mirai infected device, and as our telescope contains more than 2^{16} IP addresses, a fully random scan would target one of our IPs after on average 2^{15} steps. For 99% certainty that a seed is not correct we can stop our verification if it does not generate the first observed combination from the telescope within 2^{25} steps. Program execution can therefore be halted soon without potentially loosing candidate solutions, and yields the seed within 100 milliseconds.

6 MIRAI'S BATTLE OVER THE IOT

The attack on the random number generator and the ability to efficiently compute the seed value allows us to characterize the current state and behavior of Mirai infections worldwide. When receiving scan and brute-forcing attempts into our telescope and honeypots, we can determine the exact moment when a particular IoT device was infected. Since packets have the same random source port and window size, we can relate all subsequent interactions to a particular infection of which we know the variant type from the honeypot interactions. Finally, as soon as no further connection attempts appear within the expected inter-arrival time, we can tag the device as cleaned again, due to a reboot or being patched by the owner. We will use this method to look at the lifetime of infections, first the infection characteristics in general, and second operational aspects such as the regional biases and other modifications that have been introduced into the source code by later botmasters.

In total, we received scanning and credential brute forcing from 203,920 hosts that matched the Mirai fingerprint. This number is significantly lower than the peak size of Mirai in 2017 [8], but we find a similar total number as the 200,000 - 300,000 reported by [8]. While our honeypots would welcome any telnet traffic, we see that with 87% of all telnet compromization attempts, Mirai and its siblings are the dominant player in the IoT malware arena. Table 2 lists the top variants and how many hosts were observed in our study, the largest one infecting more than 75,000 hosts. We have identified 39 variants in total, with only the top 10 being advertised from more than 2,000 hosts during the duration of our study. The full list of variants covered in this paper is in the appendix.

6.1 Infection characteristics

As Mirai spreads as a worm, the growth of the infections is in principle exponential, but is naturally bounded by the total number of vulnerable machines. After the rapid expansion, the infection will therefore reach a steady state or die out depending on the parameters of injection and curing. Indeed, Antonakakis et al. [8] show that Mirai rapidly grew in size at the beginning, but after a while the total number of infected devices fluctuated between 200,000 and 300,000. When we measure Mirai a year later, we still find the ecosystem of Mirai-infected IoT devices in a steady state, except the infected population having significantly decreased.

Epidemiologically speaking, the memory-bound infection follows a mixed SIS (Susceptible-Infected-Susceptible) or SIR (Susceptible-Infected-Recovered) process [28]. When we measure the infection rate, we see that on average 27,182 devices are newly infected each day with a standard deviation of 18, 381. With the curing rate being almost identical – 26, 757 devices are cleaned up every day with 19,153 as standard deviation –, it is clear why the ecosystem is in a steady state. We empirically determine a basic reproduction number of $R_0 = 1.0033$, which is surprisingly low given the aggressive scanning and infection behavior, and barely enough for the worm to sustain itself. As Mirai targets IPs randomly, it is highly unlikely for pockets of undiscovered, vulnerable devices to still exist, hence new infections can only come from newly introduced vulnerable devices or from taking over devices from someone else.

Indeed, we find that the battle over the Internet of Things is largely a zero-sum game, with significantly fluctuating market shares between variants. Figure 5 shows the distribution of infected devices over the largest botnets, which surprisingly shows that the original Mirai is today only one player among many, and variants that evolved from it have gained significant market share. Surprisingly so, this competition between Mirai variants has received only little attention to date. When we count the number of devices infected by a particular strain at any given moment in time, we see momentary explosions in activity, where suddenly a lot of hosts get newly infected by a variant. As shown in figure 6, this is especially true for smaller strains such as Cult, MASUTA or OWARI, but even the main players in the ecosystem experience frequent jolts in installation size. These spikes are counterintuitive, as brute forcing from thousands of infected hosts towards random IPs should lead

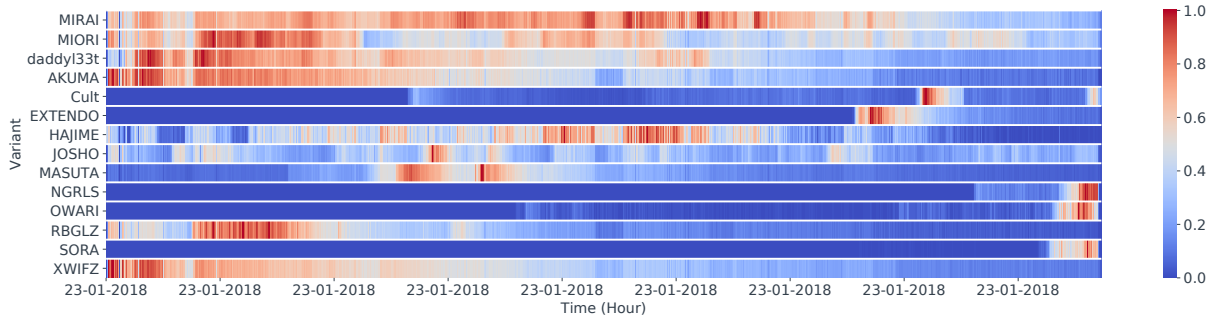


Figure 6: Lifetime distribution of variants over time, normalized per variant. Red shows peak per hour, blue shows little hosts being infected by the variant.

to a continuous influx of new infections, and can be explained with the loaders and reboots as we show later.

6.1.1 Transitions between botnets. While new IoT devices are constantly being added to the Internet, awareness and improved security (such as omitting open admin interfaces or default credentials) should over time reduce the attack surface. This would mean that in order to increase their botnet, botmasters could in the long run only grow by compromising existing devices, potentially already under the control of another botmaster. Indeed, we already observe this, with infections rarely occurring on "fresh" devices and 88.5% of all new infections on systems that have been exploited previously.

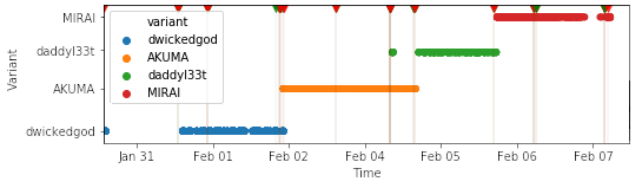
Evidently, botmasters do not want to share an infected device with other botnets, so when a botnet infects a device, one of its first actions is to kill processes such as telnet and ssh that can be used to infect this device. While this ensures that the device cannot be accessed after infection, once the infection is removed from the device – typically after a reboot – it will be vulnerable again. Such reinfections happen comparatively fast, the average duration to be reinfected is 1 day and 9 hours, with a standard deviation of 4 days and 7 hours. We determined the type of device based on daily IPv4 banner grabs from Shodan and Censys. Surprisingly, we see that routers frequently reset in practice, and remain infected only on average 12 hours (standard deviation 89 hours). Still, takeovers can happen if the telnet port was not exposed to TCP port 23 but to port 2323. While Mirai brute forces on both ports, it only eliminates programs listening on port 23. These devices can hence be "stolen" from other adversaries.

Figure 7a shows such takeover behavior for one example device. The device is reset on multiple occasions, indicated by the red triangles, but is almost immediately being reinfected. From the figure we see that this particular device is mostly re-infected with the same malware, but frequently another variant takes over. In the time between reboot and reinfection, IoT devices are up for grabs. Figure 8 shows the transition behavior between infections for all devices after a reboot for four of the major variants encountered. It shows that botnets do at a global scale lose some IoT devices towards other major players, however in most cases the re-infections happen with the same malware strain. This is made possible as successful login attempts are reported to the C&C server, thus providing the botmaster with a list of good credentials and facilitating easy reinfection. This feature was already part of the original Mirai. As the reinfections by the same malware are so successful however, it might be the case that botmasters actually use the alive pings sent to the C&C server to monitor whether a device is still active or has dropped off to infect the device again. By monitoring and reinfecting devices in this manner, they could preempt another botnet from taking over the device. While we do find indications for this hypothesis given that the average time for a device to be re-infected with the same malware is only 1.5 hours as opposed to the average infection time of 1 day and 9 hours, it is impossible to prove this conclusively by merely observing the ecosystem.

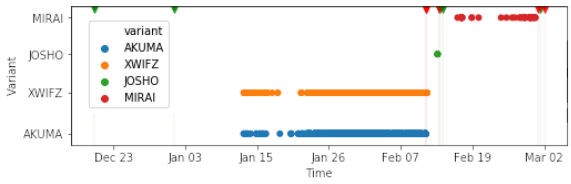
Figure 8 however also shows that a large portion of devices are never infected again after cleanup, denoted by an edge going to the "end" node. As it is unlikely for a device to not be invaded or re-infected, the device is most likely secured by its owner. In total, 175k devices are infected by only one variant over the duration of this study, whereas 28k devices were taken over at least once.

Without information about "infectable" devices, the only way for botnets to locate victims is based on port scanning. As Mirai performs scanning based on randomly-chosen candidate IPs, the largest botnets should be most likely to find these restarted devices to add them to their network, and would therefore need the least time to find and infect a device that has just been cleaned up by another malware. Figure 9 shows the probability density function for different variants, and shows that the larger the variant, the quicker it takes over a device. Size does matter: we find a strong negative correlation between the original size of a botnet and the time it takes to accumulate new devices and grow, that with a Pearson coefficient of -0.501 ($p < 0.01$) is able to explain half of the difference in infection behavior between strains.

6.1.2 Concurrent infections. To gain exclusivity over a device, the Mirai malware shuts down all processes running on port 22, 23 and 80, and binds itself to these ports to prevent other processes from doing so. While this is an effective way of keeping others out, it does not fully eliminate the possibility of other bots attacking certain devices. As telnet is a protocol that receives much unsolicited traffic [18], network operators sometimes bind their telnet not to port 23, but to port 2323. The original Mirai takes this into account by scanning port 2323 in every 10th scanning packet. While the scanning part takes this into account, the port killing part of the



(a) Devices get cleaned up and get reinfected by the same malware variant, until another variant takes over on the restart of a device.



(b) Concurrent infections on 1 IP. The first started 10 days before the second, in mid Jan our setup launched and registered both variants.

Figure 7: Infections over time against IP addresses, every plot is a distinct IP address. Green triangles denote a new infection (based on source port/ window size), red triangles mark the end of an infection.

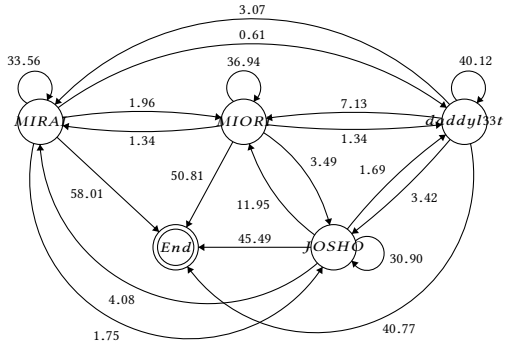


Figure 8: Device transitions between botnets in percentage of out-degree over the entire data collection period.

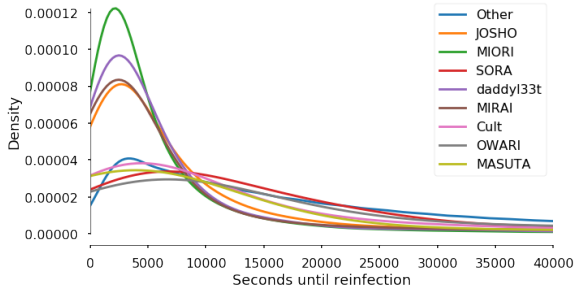


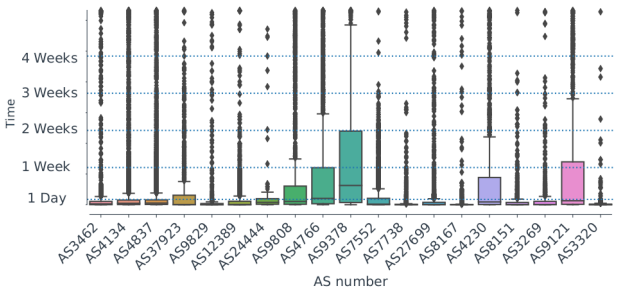
Figure 9: Probability density function of the time for a variant to take over a device previously "owned" by another.

source code does not. Therefore, devices using port 2323 as their telnet port will continue to present an open port even after infection. Figure 7b shows a host being infected with two variants at the same time, XWIFZ and AKUMA, before restarting and getting infected afterwards with other variants. In total, we observe 8.9% of the total infected devices being infected by multiple variants at one time. Note that this behavior is the result of a co-infection and not of two devices behind a NAT, when the packet generation of both stop simultaneously. In [21], we further elaborate on the issue of NATs and show how the faulty RNG can be used to identify NATs, and thus can be used to quantify IP churn across the Internet.

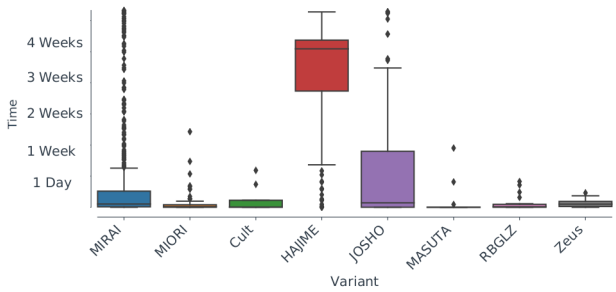
While this oversight in the Mirai source code transfers to most of its descendants, several malware authors put in effort to understand the inner workings and have removed this oversight from their code base. When looking at the variants present on concurrently infected devices, we can identify several variants that do succeed in killing their competitors and locking them out. While MIORI and JOSHO are present on 5680 and 3726 concurrently infected devices, we find no rivals when MASUTA and SORA are running, showing them to be more effective in stopping competition.

6.1.3 Compatibility of Malware. While botnet size has a significant influence on its ability to infect new victims, we also see major differences in infection characteristics between different countries and autonomous systems (ASes). Figure 10a shows the duration a particular strain has control over a given device until it reboots, for a selection of ASes which more than 1,000 compromised devices. While the average duration of an infection is relatively short, there are major outliers where a large share of IoT malware can hold on to devices for up to a week or longer. As can be seen on the box plots, average values of infection times are not a good measure to understand the behavior of IoT malware, given the sheer number of these outliers and their deviation from the mean. Not only are the distributions highly heterogeneous, but we see also the emergence of clusters, indicating that there are groups with similar behavior.

We can clarify this behavior better if we look at infection characteristics not only from the perspective of the victim device, but in symbiosis with a particular malware strain. Figure 10b breaks down the infection duration for one of the ASes with large outliers. Here, we see that the time a particular malware strain holds a device captive is vastly different. In case of AS9121, MIORI and MASUTA are largely unsuccessful in maintaining a foothold and devices reboot on average after 106 and 32 minutes. The fact that in 99% of cases neither malware runs longer than 239 minutes on vulnerable IoT devices within this AS before the device resets and is later infected by something else, suggests some incompatibility of this malware or the way it is used with a particular type of host predominantly used in this network. We can rule out that this behavior is the result of coordinated cleanup activities or some centralized reboots due to a power outage, as devices eventually and at different times fall victim again. As we see from the graph, it appears that JOSHO is a more suitable IoT malware for this AS, while the best performance is achieved by Hajime with an average infection duration 9 times larger than the average for this AS, and even the shorter Hajime infections outperforming the best performers of



(a) Infection times per AS with more than 1,000 infections, showing large differences between the infection times of different ASes.



(b) Infection times of different variants on AS9121, showing large differences of variants within ASes.

Figure 10: Infection times of different ASes and variants.

the other IoT malware by far. We find similar heterogeneity for many ASes and malware types. It seems that malware frequently causes issues, and reboots of IoT devices can be attributed not just to external influences (user reboots, power outages, updates, etc.) but to a significant degree to the malware itself.

6.2 Customization and Evolution

Different types of malware seem to work better or worse depending on the AS devices are located in. Once an IoT device reboots, it is just a matter of time before it is re-discovered and reinfected. As shown previously, the larger a botnet is, the faster it can locate routers to potentially infect. This makes it hard for new botnets to enter the "market" and compete successfully for vulnerable devices. In response to this, it would make sense for botmasters to customize their strategy where to look for victims, and localize productive niches to exploit [10]. Such customization could on the one hand be done by only targeting the address space of particular ASes, or on the other hand be done based on curating credential lists. Using open-source intelligence, actors could identify default username/password combinations for devices that occur frequently but are not so mainstream that other botmasters might target them.

6.2.1 Regional biases. One way of creating a niche in which a botnet can thrive is by targeting devices not in demand by others. In the case of Mirai, which sole attack vector is the use of weak credentials, the way to target new devices would be the addition of new username and password combinations which no other variant uses. An effective approach would be to include credentials that are used by a significant number of devices as their default. With devices such as routers deployed in bulk by telecom operators to their customers, the location of these devices can be heavily biased to a region. When a variant targets such a device, it would therefore grow its influence in a specific geographic location. In figure 11, the distribution per country is plotted for different Mirai variants. The biggest botnets, MIORI and Mirai, occur widely. Others however, such as AKUMA and MASUTA, are heavily biased to Japan and Vietnam respectively. This imbalance stems from the credentials: AKUMA includes 17 unique username and password combinations not seen in other variants, among which the combination "admin,oelinux123". This combination is targeted at the

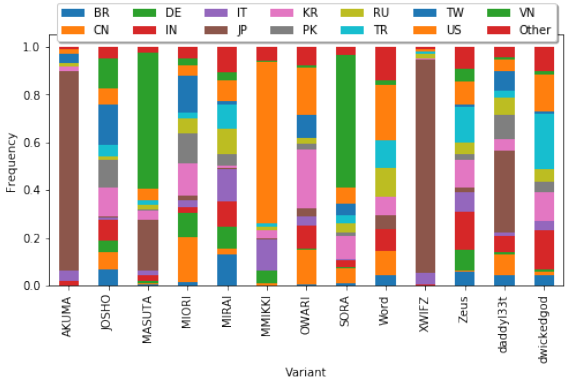


Figure 11: Regional infections per variant, normalized over the total number of seen IoT devices in a country. Several show heavy biases, mainly due to targeted password lists.

EE 4GEE HH70 ROUTER, which is a mobile WiFi router especially popular in Japan. Masuta on the other hand has only one unique combination: "root,00000", the default credentials used to login to an old DVR. After adding this, the MASUTA botnet took over 2645 hosts in Vietnam in one single day, growing their botnet by a factor 4 from 810 infected to 3455 hosts.

6.2.2 Increasing your market share. Specializing on ASes pays off when one is able to find networks with a large number of vulnerable devices. After the initial foothold, variants seem to be effective in maintaining the majority share of an AS, making the initial foothold even more important. Botmasters can optimize their botnet for certain ASes by changing the credential list, and removing those that will not be successful as bots execute a limited number of login attempts. By doing so, the effectiveness in other ASes might be impacted due to the deletion of important credentials. For MIORI, Mirai and JOSHO, we have observed infected devices belonging to the same variant from several ASes brute forcing our honeypots using different password lists, showing that malware authors diversify and launch different versions to maximize effectiveness.

Name	R	p
MASUTA	-0.064	<0.1
Cult	-0.086	<0.05
OWARI	-0.120	<0.001
daddyl33t	-0.124	<0.001
XWIFZ	-0.140	<0.001
dwickedgod	-0.170	<0.001
MIORI	-0.172	<0.001
MIRAI	-0.179	<0.001
HAJIME	-0.188	<0.001
JOSHO	-0.206	<0.001
OBJPRN	-0.663	<0.001

Table 3: Correlation between botnet size and its growth.

AS	R	p
Frontier Communications of America Inc.	-0.519	< 0.001
asn for Heilongjiang Provincial Net of CT	-0.511	< 0.001
Ratt Internet Kapacitet i Sverige AB	-0.501	< 0.001
Bredband2 AB	-0.484	< 0.001
Bredbandsson AB	-0.475	< 0.001
Viettel Group	-0.129	< 0.001
OPTAGE Inc.	-0.127	< 0.001
Jupiter Telecommunications Co. Ltd.	-0.123	< 0.001
Jupiter Telecommunication Co. Ltd	-0.119	< 0.001
NTT Communications Corporation	-0.104	< 0.01

Table 4: Correlation between the size of an AS and its growth, ordered by coefficient for the top and bottom 5 ASes.

Another way to grow a botnet is by better targeting the scanning activity towards vulnerable devices. While Mirai chose to send packets to either port 23, or with 10% probability to 2323, we observe botmasters removing either one of the two to make the scan more focused towards one port. Removing 2323 does in theory make sense, as Shodan reports there are more than 20 times as many devices on port 23 than on 2323. We classify a host actively avoiding a port when we have observed 44 packets but none on the considered port, as this gives us a 99% chance the port should have been observed, and find that mainly variants of Mirai and JOSHO avoiding port 2323 with 7484 and 4558 hosts only scanning 23. On the other hand, we find 105 hosts belonging to MASUTA only targeting port 2323. For hosts avoiding port 2323, we find no difference in distribution over ASes.

6.2.3 Watching and learning from your competitors. If such strategies are fruitful and actors are aware of the success, we would assume those with a competitive advantage to proliferate. As unique passwords are one success factor, we tracked how Mirai variants introduced new credentials during our study, and whether others would adopt these combinations to their own code base.

Figure 12 shows password adoption behavior of the largest Mirai variants, and the smaller but particularly noteworthy strains MM and OBJORN. The size of the connections shows the amount of passwords that get transferred from one to the other, with the color of the edge denoting the variant that pioneered the password. What immediately springs out is that all variants except Mirai engage in password adoption, but that the two largest botnets predominantly operate on credentials they have pioneered themselves. Mirai does not introduce external credentials at all and the other large player MIORI borrows a minor share only from Mirai.

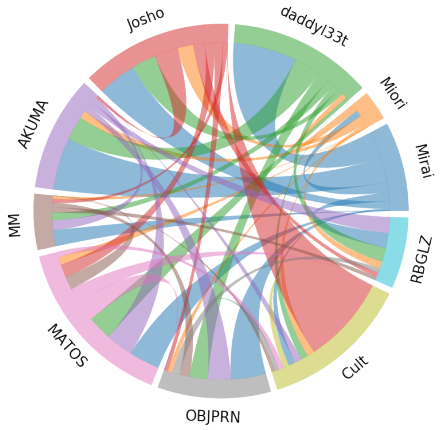


Figure 12: Passwords adoption between variants. Outgoing edges denote a credential adoption, edges are colored in the color of the variant they came from.

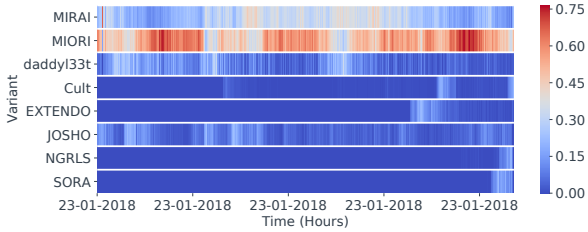


Figure 13: Infection distribution for AS4134 per hour, in percentage of infected devices per variant. The AS is highly dominated by MIORI, others only hold a small percentage.

Overall, smaller botnets are generally much more likely to pick up passwords (MATOS/MM/OBJPRN) rather than being innovative on their own. Even more surprisingly, we see that copying behavior follows some form of hierarchy. The large botnets only feed from other large botnets but not smaller ones. There are also clear preferred relationships, for instance passwords used in the variant Cult predominantly originate from JOSHO.

6.2.4 Keeping the net alive. If a large botnet is amassed, the worm-like structure of the botnet should ideally be able to sustain the loss of devices by growing at least as fast as they decay. The authors in [8] refer to this as the stable state and find that at the end of their measurement period, the overall size of the Mirai botnet was shrinking. When looking at the different strains, we also observe all variants to shrink over time, and there is an overall negative correlation between the size of a botnet and its growth ($r = -0.16$, $p < 0.01$). Table 3 shows these correlations per variant and shows large differences between the variants, where large ones such as Mirai, MIORI and JOSHO are below average, but others such as OBJPRN are much less likely to survive. The survival rate of a botnet seems to be dependent on the time it takes for bots to find a victim to brute force, giving a correlation of $r = -0.516$ with $p = .02$ between the

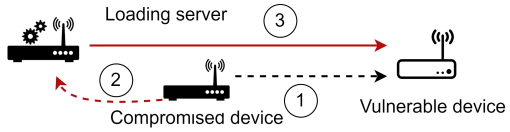


Figure 14: In Mirai's loading infrastructure, bots identify devices where they can log in (1). These are then reported to a loading server (2), which then performs the infection (3).

correlation scores in Table 3 and the average time between brute force attempts on our honeypots from a single infected host.

Table 4 shows the growth rates of the top and bottom performing ASes in our study. We confirm the findings by [8], who showed a decline in the number of bots on certain ASes. We have not observed ASes with a positive correlation between the amount of infected devices and the growth of this number. This shows that the botnets are slowly losing their grip on the IoT devices.

6.2.5 Loading infrastructure. Devices infected by Mirai are used to scan the Internet to help infect new devices by brute forcing passwords. After the correct credentials are found and the device has logged in, the credentials are sent to a server responsible for the infection of new devices [8, 26]. Figure 14 shows this process. By making a centralized server responsible for the infection, it is easier to update the software loaded onto new devices, or customize infections if new devices are discovered.

Identifying these loading infrastructures is trivial, as they always immediately provide the correct credentials when they log in to our honeypots, and never input false credentials. If infections would occur from the infected devices themselves, these would need at least several tries to guess the correct combination before loading the malware onto the device. By identifying the bot that has successfully brute forced the password that is later sent by the loading infrastructure, we are able to identify which variant connects to which loading IP address. Figure 15 shows IP addresses used for loading the malware on devices and the bot variant responsible for letting the server know the correct password. Each colored circle represents a loading server, where the color indicates the variant it spreads and the size the number of installations it does respectively. Individual dots mark the IP address of an infected device, the line to a circle which loading server it provided the credentials to. The color of the line represents the malware variant the infected IP had at that moment. We find that XWIFZ and AKUMA both use a large number of loading servers relative to other malware strains. Even more surprisingly, we see that a large portion of this infrastructure base is shared, as either a bot of AKUMA or XWIFZ has brute forced a password correctly before one of these loaders logged in. For MIRAI we can observe a few small clusters, using the variant name MIRAI as its loader, while in the original source code [4] the loader for MIRAI identified itself with the ECCHI string, which means that actors also adapt the software in the backend.

Almost all variants use centralized loading servers, and are therefore prone to takedowns and IP-based blocking of enabling infrastructure. To limit the risk of a takedown or block, botmasters might opt to regularly change their loading infrastructure by using for example DNS. In practice however, we do not see this behavior in any

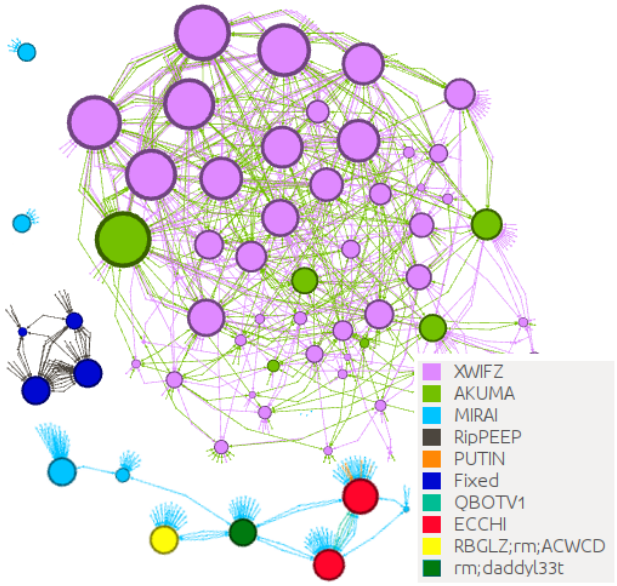


Figure 15: Identified loading servers for several variants. Edges denote successful brute force attempts and are colored by the variant of the brute-forcing bot.

centralized variant, as identified loading servers have been active over the entire period our honeypots were active. Only two variants have significantly changed their loading infrastructure, and use the bots not only to scan but also to perform the infection, making these botnets significantly harder to block. One of these botnets, identified by eight random characters as variant name, spreads its loading servers throughout the network and cycles them around, making it hard to identify and block these loading servers.

Mirai's source code included the original loader used by Mirai, which consequently was copied by its descendants. As with the password lists, the loading code has been altered by some of the variants to remove indicators from the issued commands [3]. Alterations include the original ECCHI variant string, file names created on the system, but also the entire loading procedure. These alterations extend into the commands issued by the bots after a compromisation, where some variants already request more data from the device such as the "/proc/mounts" file. Table 5 shows the changes made by different variant authors in these criteria, and shows that full customization is rare. Additionally, the last two loader services shown in figure 15 change their command structure over time. We first identify the server "RBGLZ;rm;ACWCD" as variant ACWCD, but over time the server changes the loader code and its identifier. The loader server for "rm;daddyl33t" behaves the same, first identifying with daddyl33t, and later changing the commands sent from the infrastructure. Curiously, both modified the identification string to include a command.

6.2.6 Honeypot evasion. In our discussion of the source code, we have pointed out that Mirai skipped over IP addresses it randomly generated if they would fall within a particular range. The original Mirai blocked probes to multicast ranges or invalid addresses, such

Name	Loader name	Load via bots	Loader commands	Bot shell entries
XWIFZ	✓			
RipPEEP	✓		✓	✓
PUTIN				✓
HAJIME	✓	✓	✓	✓
daddyl33t				✓
8-characters	✓	✓	✓	

Table 5: Changes in bot and loader code across variants from the original Mirai source code.

	Network	Akiru	AKUMA	JOSHO	MIRAI	MIORI	OWARI	RBGLZ	SORA
Enter.	authors1 /16		✓	✓	✓	✓			
	authors2 /16		✓	✓	✓	✓			
	authors3 /16		✓	✓	✓	✓	✓	✓	✓
Cloud	165.227/16	✓	✓	✓	✓	✓	✓	✓	✓
	167.99/16	✓	✓	✓	✓	✓	✓	✓	✓
	172.31/16	✓	✓	✓	✓	✓	✓		✓
Home	80.114/16		✓	✓	✓	✓	✓		✓
	83.172/16	✓	✓	✓	✓	✓	✓		✓

Table 6: Honeypot locations hit by variants of Mirai.

as `0.0.0.0/8`, as well as selected large organizational networks such as the Department of Defense and the US Postal Service.

In order to remain undetected, and in an effort to speed up the scanning process by skipping known IP addresses that are not of interest, adversaries can update the blacklist to include these IP ranges. To quantify to which extent malware variants update and utilize blacklisting, we distributed our honeypot agents in 8 different /16 networks assigned to Internet Service Providers, public clouds and one enterprise network. By comparing incoming probes across the different ranges coming from the same source during the same infection period, we can quantify which Mirai variants update and customize their blacklists. Table 6 lists a selection of variants and the ranges we have observed them at. We clearly see that the largest malware MIRAI, MIORI or JOSHO show no discriminative behavior, but that smaller variants such as OWARI selectively exclude the enterprise range. Cloud providers are attractive across all variants, even though one would not expect to find many IoT devices in these networks. The ranges of DigitalOcean (165.227/16 and 169.99/16) are for example not excluded by any single variant.

This blacklisting behavior indicates that adversaries are conscious about where to find victims or where their activities might be monitored. Multiple variants perform evasion, and the ranges they evade are similar, which could mean that all of these actors either research where infrastructure is located, or that locations are shared among actors. While the first reason is hard to verify, we have found evidence of these lists being shared online [6], labeled with the exact institutions or organizations these ranges belong to. Additionally, a plethora of blocklists that can be readily put into source code exist online, which block major ranges of the Internet.

Verifying Blocklists. We verify whether a variant blacklists additional IP addresses in two ways: First, as Mirai's target selection merely skips generated IP addresses if the target appears on the list, we can use this feature and our ability to efficiently bruteforce the seed to selectively test which blacklist an actor is using. Whenever IP addresses are skipped the state of the RNG is advanced, and we can now test whether we can break the seed and reproduce the sequence of probing packets given one of the lists we could locate online. Second, we use netflows from a Tier-1 operator to verify that the bots do indeed never probe the particular IP ranges.

We find that some of the botmasters go the extra mile of updating this part of the Mirai source code. Akiru, OWARI, RBGLZ and SORA have adopted customized blacklists, whereas MIORI, AKUMA and JOSHO all run with the original Mirai algorithm. Especially larger variants are not homogeneous, and there exist not just one version that identifies itself as Mirai. 93% of all hosts infected by Mirai are running the original source, but 7% have adapted lists. Even within these, we find differences in behavior, with some of them blacklisting some of the ranges we monitor but not others, while some versions block more extensively.

7 LIMITATIONS

In order to be detected by us, infected devices had to connect to either our telescope or one of our 7,500 honeypots. As discussed above, Mirai and its variants employ a blacklist, and it is possible that there are variants that do not connect to any of the eight /16 networks we were present in. To assess potential blind spots, the Tier1 operator collected the set of IP addresses that showed brute forcing behavior on telnet on a given day, which we compared against the list of hosts that brute forced us that day. This revealed 2.9% additional IP addresses that were targeting host on the Internet via telnet but not present in our analysis, thus our study provides an almost complete picture of telnet brute forcing, and with Mirai accounting for 87% also a solid assessment of the Mirai ecosystem.

For this study, we define a malware as Mirai-based given distinct features in the packet generation process, such as a TCP sequence number - IP match, as well as the fixed window size and IP ID fixed for the session. While later botmasters copied the original Mirai source code, they also introduced modifications. These were functional in nature, to introduce new features or avoid certain ranges. It is possible that some modified the packet generation itself, for example randomizing every header value with a RNG not related to the original one. These variants are not part of our analysis as they would not stand out based on structural features. Given that the netflows reveal only very few IP addresses going after telnet but not in our honeypot dataset, indicates that such deep modification would occur rarely if at all.

8 CONCLUSION

After the source code of the Mirai botnet was shared, many new actors have sprung up to take advantage of misconfigured IoT devices. In this work, we exploited a flaw in the design and entropy of Mirai's RNG, allowing us to track the exact infection time of a host as well as track how infections are evolving. We observe a continuous battle over the Internet of Things among different strains. We find that these IoT botnets on their own are not self-sustaining, and that the success of malware variants depends on their installation size, but also in how well they seem adapted to occupy specific niches of vulnerable devices.

REFERENCES

[1] [n. d.]. DDoS attack on Dyn DNS. ([n. d.]). https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/.

[2] [n. d.]. DDoS attack on OVH. ([n. d.]). https://www.ovh.com/world/news/articles/a2367.the-ddos-that-didnt-break-the-camels-vac.

[3] [n. d.]. Hajime botnet analysis. ([n. d.]). https://www.infopoint-security.de/media/Botnet_Hajime_Radware_Analyse.pdf.

[4] [n. d.]. Mirai scanner source code. ([n. d.]). https://github.com/jgamblin/Mirai-Source-Code/blob/3273043e1ef9c0bb41bd9fcdc5317f7b797a2a94/mirai/bot/scanner.c.

[5] [n. d.]. Mirai source code shared. ([n. d.]). https://krebsonsecurity.com/2016/10/source-code-for-iot-botnet-mirai-released/.

[6] [n. d.]. Shared network ranges to blacklist. ([n. d.]). https://sciencesmaths.skyrock.com/323627548-very-famous-ip-s-try-2-hack-them-&-they-will-f-u.html.

[7] [n. d.]. Tracking Mirai: An In-depth Analysis of an IoT Botnet. ([n. d.]). Master thesis, Pennsylvania State University, 2017.

[8] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. 2017. Understanding the mirai botnet. In *26th USENIX Security Symposium*. 1093–1110.

[9] Manos Antonakakis, Roberto Perdisci, Yacin Nadji, Nikolaos Vasiloglou, Saeed Abu-Nimeh, Wenke Lee, and David Dagon. 2012. From throw-away traffic to bots: detecting the rise of DGA-based malware. In *Presented as part of the 21st USENIX Security Symposium*. 491–506.

[10] Hugo LJ Bijmans, Tim M Booij, and Christian Doerr. 2019. Just the Tip of the Iceberg: Internet-Scale Exploitation of Routers for Cryptojacking. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*.

[11] James R Binkley and Suresh Singh. 2006. An algorithm for anomaly-based botnet detection. *SRUTI* 6 (2006), 7–7.

[12] Norbert Blenn, Vincent Ghiette, and Christian Doerr. 2017. Quantifying the Spectrum of Denial-of-Service Attacks through Internet Backscatter. In *International Conference on Availability, Reliability and Security (ARES)*.

[13] Giovanni Bottazzi and Gianluigi Me. 2014. The botnet revenue model. In *Proceedings of the 7th International Conference on Security of Information and Networks*.

[14] Robert G Brown, Dirk Eddelbuettel, and David Bauer. 2013. Dieharder: A random number test suite. *Open Source software library, under development, URL http://www. phy. duke. edu/˜ rgb/General/dieharder. php* (2013).

[15] Orçun Çetin, Carlos Ganán, Lisette Altena, Takahiro Kasama, Daisuke Inoue, Kazuki Tamiya, Ying Tie, Katsunari Yoshioka, and Michel van Eeten. 2019. Cleaning Up the Internet of Evil Things: Real-World Evidence on ISP and Consumer Efforts to Remove Mirai. In *NDSS*.

[16] Evan Cooke, Farnam Jahanian, and Danny McPherson. 2005. The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets. *SRUTI* 5 (2005), 6–6.

[17] Andrei Costin and Jonas Zaddach. 2018. Iot malware: Comprehensive survey, analysis framework and case studies. *BlackHat USA* (2018).

[18] Zakir Durumeric, Michael Bailey, and J Alex Halderman. 2014. An internet-wide view of internet-wide scanning. In *23rd USENIX Security Symposium*. 65–78.

[19] Vincent Ghiette, Harm Griffioen, and Christian Doerr. 2019. Fingerprinting Tooling used for SSH Compromisation Attempts. In *International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*.

[20] Dan Goodin. 2017. BrickerBot, the permanent denial-of-service botnet, is back with a vengeance. *Ars Technica* (2017).

[21] Harm Griffioen and Christian Doerr. 2020. Quantifying Autonomous System IP Churn using Attack Traffic of Botnets. In *International Conference on Availability, Reliability and Security (ARES)*.

[22] Guofei Gu, Roberto Perdisci, Junjie Zhang, and Wenke Lee. 2008. Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection. (2008).

[23] Fariba Haddadi and A Nur Zincir-Heywood. 2015. Botnet detection system analysis on the effect of botnet evolution and feature representation. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*. 893–900.

[24] Hwanjo Heo and Seungwon Shin. 2018. Who is knocking on the telnet port: A large-scale empirical study of network scanning. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. 625–636.

[25] Stephen Herwig, Katura Harvey, George Hughey, Richard Roberts, and Dave Levin. 2019. Measurement and Analysis of Hajime, a Peer-to-peer IoT Botnet. In *NDSS*.

[26] Georgios Kambourakis, Constantinos Kolias, and Angelos Stavrou. 2017. The mirai botnet and the iot zombie armies. In *IEEE Military Communications Conference (MILCOM)*.

[27] Anestis Karasaridis, Brian Rexroad, David A Hoeflin, et al. 2007. Wide-Scale Botnet Detection and Characterization. *HotBots* 7 (2007), 7–7.

[28] William Ogilvy Kermack and A. G. McKendrick. 1927. A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society A* 115, 772 (1927).

[29] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Jeffrey Voas. 2017. DDoS in the IoT: Mirai and other botnets. *Computer* 50, 7 (2017), 80–84.

[30] Chao Li, Wei Jiang, and Xin Zou. 2009. Botnet: Survey and case study. In *Fourth International Conference on Innovative Computing, Information and Control (ICICIC)*.

[31] Joel Margolis, Tae Tom Oh, Suyash Jadhav, Young Ho Kim, and Jeong Neyo Kim. 2017. An In-Depth Analysis of the Mirai Botnet. In *International Conference on Software Security and Assurance (ICSSA)*.

[32] Claudio Mazzariello. 2008. IRC traffic analysis for botnet detection. In *The Fourth International Conference on Information Assurance and Security*.

[33] Trend Micro. 2017. Persirai: New internet of things (IoT) botnet targets ip cameras. *URL: https://blog. trendmicro. com/trendlabs-security-intelligence/persirai-new-internetthings-iot-botnet-targets-ip-cameras* (2017).

[34] Yin Minn Pa Pa, Shogo Suzuki, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, and Christian Rossow. 2015. IoTPOT: analysing the rise of IoT compromises. In *9th USENIX Workshop on Offensive Technologies (WOOT 15)*.

[35] Yin Minn Pa Pa, Shogo Suzuki, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, and Christian Rossow. 2016. Iotpot: A novel honeypot for revealing current iot threats. *Journal of Information Processing* 24, 3 (2016), 522–533.

[36] Stijn Pletinckx, Cyril Trap, and Christian Doerr. 2018. Malware coordination using the blockchain: An analysis of the cerber ransomware. In *2018 IEEE Conference on Communications and Network Security (CNS)*.

[37] CP Reasearch. [n. d.]. Iotroop botnet: The full investigation, Oct. 29 2017. ([n. d.]).

[38] Mohammd Reza Rostami, Meisam Eslahi, Bharanidharan Shanmugam, and Zuraini Ismail. 2014. Botnet evolution: Network traffic indicators. In *International Symposium on Biometrics and Security Technologies (ISBAST)*.

[39] T Seals. [n. d.]. Bricker bot follows mirai tactics to permanently dos iot devices, Apr. 7 2017. ([n. d.]).

[40] Saleh Soltan, Prateek Mittal, and H Vincent Poor. 2018. BlackIoT: IoT botnet of high wattage devices can disrupt the power grid. In *27th USENIX Security Symposium*. 15–32.

[41] W Timothy Strayer, David Lapsely, Robert Walsh, and Carl Livadas. 2008. Botnet detection based on network behavior. In *Botnet detection*. Springer, 1–24.

[42] Jing Wang and Ioannis Ch Paschalidis. 2016. Botnet detection based on anomaly and community detection. *IEEE Transactions on Control of Network Systems* 4, 2 (2016), 392–404.

[43] Jun Xu, Jinliang Fan, Mostafa Ammar, and Sue B. Moon. 2001. On the Design and Performance of Prefix-preserving IP Traffic Trace Anonymization. In *ACM SIGCOMM Workshop on Internet Measurement*.

[44] Xiaolu Zhang, Oren Upton, Nicole Lang Beebe, and Kim-Kwang Raymond Choo. [n. d.]. IoT Botnet Forensics: A Comprehensive Digital Forensic Case Study on Mirai Botnet Servers. ([n. d.]).

[45] Zhi-Kai Zhang, Michael Cheng Yi Cho, Chia-Wei Wang, Chia-Wei Hsu, Chong-Kuan Chen, and Shiuhpyng Shieh. 2014. IoT security: ongoing challenges and research opportunities. In *IEEE 7th international conference on service-oriented computing and applications*.

APPENDIX

List of Mirai variants covered in this paper

'MIORI', 'MIRAI', 'OWARI', 'OOMGA', 'MATOS', 'PUTIN', 'AKUMA', 'NGRLS', 'dwickedgod', 'EXTENDO', 'MMIKKI', 'MASUTA', 'RipPEEP', 'JOSHO', 'sunless', 'TSUYOI', 'RBGLZ', 'chmod', 'ASUNA', 'OBJPRN', 'XWIFZ', 'QBOTV1', 'satori', 'daddyl33t', 'FREEPEIN', 'HHHH', 'MEMES', 'MM', 'HAJIME', variant that identifies itself with '8 Randomized characters', 'WICKED', 'KATRINA', 'kkuuaassaa', 'AKIRU', 'SORA', 'Cult', 'Zeus', 'Word', 'REKAI'