From Mirai to Gorilla: Deep Dive into a Long-Lasting DDoS-for-Hire Botnet

Maarten Weyns
Delft University of Technology
m.b.m.weyns@tudelft.nl

Dario Ferrero
Delft University of Technology
d.ferrero@tudelft.nl

Stefan Op de Beek
Delft University of Technology
s.opdebeek@tudelft.nl

Daniel Wagner
MPI-INF / DE-CIX
daniel.wagner@de-cix.net

Georgios Smaragdakis
Delft University of Technology
g.smaragdakis@tudelft.nl

Harm Griffioen
Delft University of Technology
h.j.griffioen@tudelft.nl

Abstract

In 2016, the Mirai botnet swept the Internet, ushering in a new era of DDoS attacks. Over the following decade, spinoffs of the Mirai botnet transitioned from simple attack tools into commercial platforms, offering Distributed Denial of Service (DDoS) attacks for Hire. Such platforms enable users to launch large-scale DDoS attacks with minimal technical expertise. One notable example is the Gorilla Botnet, which was operational between Fall 2024 and Summer 2025, an unusually long lifetime compared to similar Mirai-based Botnets.

In this paper, we reverse-engineer the Mirai-based Gorilla Botnet and aim to understand its design, engineering decisions, and marketing strategies to enhance its resilience and success. We investigate its operational characteristics, including the types of attacks it supports, its underlying infrastructure, and the behavior of its bots. We find that Gorilla's longevity stems from targeted improvements, including two software development phases and learning from previous releases, setting it apart from typical Mirai-based botnets. In the process, we analyze the firepower and attack vectors of the Gorilla botnet and characterize the business types of its targets.

1 Introduction

Botnets remain a cornerstone of large-scale cyber threats [15], leveraging distributed networks of compromised devices to launch attacks ranging from Distributed Denial of Service (DDoS) to credential theft and spam campaigns. Early botnets such as Mirai [5], infamous for exploiting weak credentials on Internet of Things (IoT) devices, demonstrated the potential of commodity malware when its source code was publicly leaked in 2016 [21]. In the years since, adversaries have evolved their tactics and tooling, giving rise to a new generation of botnets that combine enhanced evasion techniques with DDoS-for-Hire, a business model in which attackers rent out access to botnet-driven DDoS capabilities, transforming Mirai-based tools to full-service attack platforms.

However, DDoS-for-Hire platforms based on Mirai, e.g., Bashlite and Tsunami [59] struggle to maintain stability, reliability, and a long-lasting presence with the average lifespan range from weeks to a couple of months [8]. A noticeable exception is the Gorilla Botnet, whose operation was first reported in the fall of 2024 and was taken down by law enforcement through coordinated efforts in the Summer of 2025. The "Gorilla" botnet (often referred to as *GorillaBot*) is a Mirai-based DDoS-for-Hire platform that was flagged by national CERTs following a DNS amplification campaign against critical infrastructure [45] in October 2024. Before that, Gorilla has already conducted over 300,000 attacks across more than 100 countries, targeting a wide range of services including gaming platforms, financial institutions, and media outlets [48].

The Gorilla Botnet offers a range of services with varying price levels, catering to different types of attack demands and budgets. The Gorilla Botnet also employs marketing strategies on social media and other channels to demonstrate the capabilities of its platform, highlighting high-profile attacks to showcase its efficacy. The botnet is provided as a *DDoS-for-Hire* service, which enables customers to launch attacks without needing to manage their own botnet infrastructure and without revealing their real identity. While attacks are typically short-lived, lasting only a few minutes, they can be highly disruptive, with some campaigns succeeding in hampering high-profile organizations. For example, the Swedish Public Broadcasting service SVT [11].

In this paper, we reverse-engineer the Gorilla Botnet to understand the design, engineering decisions, and marketing strategies employed to enhance its resilience and success, thereby making it stand out among other Mirai-based botnets. We investigate the Gorilla botnet to understand why it was so successful. We leverage a dedicated Command-and-Control (C2) "milker", where we emulate a bot and are effectively "part of the botnet" from an attacker's point of view. We combine attack data obtained from our milker with network telescope data, netflow records, and Gorilla malware binaries to provide the first in-depth characterization of the Gorilla Botnet architecture, the modern DDoS-for-Hire economy, and DDoS attack economics. Our contributions are multifold:

- We present a systematic dissection of the Gorilla infection lifecycle, its infrastructure, its encrypted C2 protocols, and functionalities in Section 4. We notice that despite inheriting heavily from the original Mirai code, Gorilla underwent two major software releases, learning from previous experiences to make it more resilient.
- We perform a detailed study of the target profiles and the attack vectors that were weaponized for the attacks in Section 5, and show trends in the attack traffic and high-profile targets that are successfully attacked.
- We empirically evaluate Gorilla's firepower and attack efficacy in Section 6, showing that the network relies on centralized infrastructure next to the botnet for attacks.

2 Background

Denial of Service for Hire: Denial of Service (DoS) attacks aim to render a service unavailable to legitimate users. This is typically achieved by resource exhaustion, where the attacker consumes the target's resources (such as bandwidth, memory, or processing power) to the point that it can no longer respond to legitimate requests. In a Distributed Denial of Service (DDoS) attack, multiple compromised systems (often part of a botnet) are used to flood the target with traffic.

To launch a DDoS attack, an attacker typically needs to control a large number of compromised devices, and direct them to send a massive amount of traffic to the target, overwhelming its resources and causing service disruption. In recent years, the emergence of Denial of Service for Hire has made it easier for even non-technical individuals to launch DDoS attacks [62]. This model allows attackers to rent access to botnets or DDoS attack tools, allowing users to launch attacks with minimal technical knowledge.

Mirai Botnet: The Mirai botnet is one of the most notorious examples of a DDoS attack tool that has been used to launch large-scale attacks. Mirai scans the internet for devices that are using default usernames and passwords, which are common in many IoT devices. Once it finds a vulnerable device, it infects it and adds it to the botnet. The infected devices can then be commanded to launch coordinated attacks against specified targets, overwhelming them with traffic. The source code of Mirai was released in 2016, leading to a proliferation of similar botnets and DDoS attack tools [5]. The ease with which Mirai can be deployed has made it a popular choice for attackers, contributing to the rise of DDoS-for-Hire offerings. Many other botnets (such as Gorilla) have since been developed using similar techniques, lowering the barrier to entry for launching DDoS attacks even further.

The common setup of a Mirai-based botnet is shown in Figure 1. The botnet consists of a command and control (C2) server that issues commands to the infected devices, which are referred to as bots. The bots, once infected, continuously scan for new vulnerable devices to infect, expanding the botnet. If a device is found, an 'infector' is used to exploit it and download the payload from a 'hosting server'.

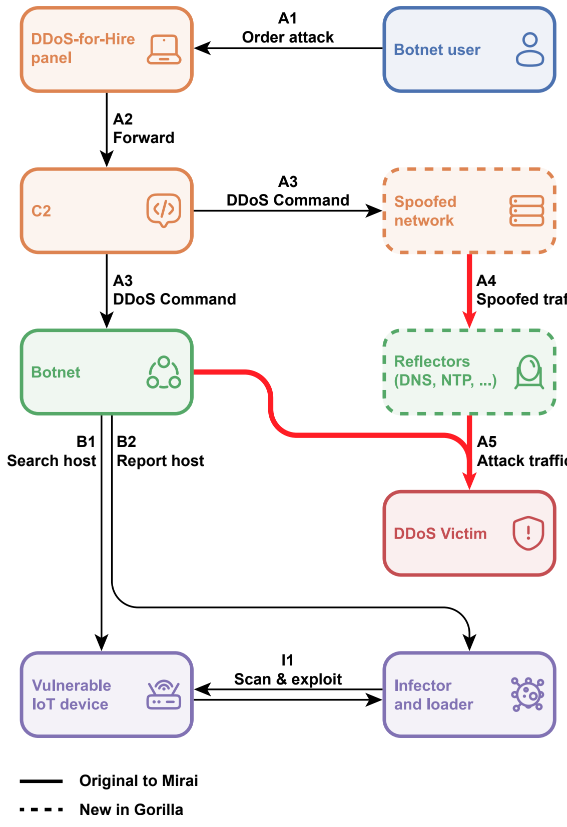


Figure 1: Common setup of a Mirai-based DDoS platform.

3 Methodology

The goal of this work is threefold: (1) to understand the Gorilla platform and its DDoS-for-Hire (stresser) business model, (2) to follow and characterize the attacks it performs, and (3) to map the ecosystem of the platform. To achieve this, we employ a combination of reverse-engineering, network traffic analysis, and data collection techniques.

In this section, we outline the methodologies used in our research, detailing how we collect and analyze data from the Gorilla platform. We also discuss the tools and techniques employed to reverse-engineer the malware, monitor its C2 communications, and analyze network traffic. A high-level summary is presented in Table 1.

Dataset	**Method**	**Start**	**End**	**Description**	**Section**
Malware samples	Reverse engineering	2024-09-05	2025-07-05	Inner workings of the botnet.	4, 5, 6
Attack commands	C2 milker	2024-10-07	2025-06-28	Continuous polling of C2 commands.	4, 5, 6
Scan traffic	Network telescope	2024-09-05	2025-07-14	Scanning traffic towards "dark" IP space.	4
Major IXPs + ISPs	Netflow analysis	2025-03-13	2025-06-04	Flow records for volumetric analysis.	5, 6
Telegram groups	Chat monitoring	2025-04-19	2025-07-14	Monitoring of the Gorilla Telegram group.	4
C2 panel	Web scraping	2025-04-22	2025-06-19	Regular scraping of the control panel.	4, 6
DStat dashboards	Web scraping	2025-04-29	2025-06-28	Scrapes every 10 seconds for attack size.	6
Attack traffic	Self-Attacks	2025-06-20	2025-06-26	12 attacks on our own infrastructure.	6

Table 1: Summary of Datasets, Methodologies, Extracted Information, and which Section they are used.

3.1 Reverse Engineering

We begin by performing static analysis of malware samples associated with the Gorilla platform. This analysis enables us to enumerate the implemented attack vectors, observe how the bots connect to and communicate with the Command-and-Control (C2) server, and identify additional functionalities.

During this process, we focus on Gorilla binaries that are compiled for the ARMv7 architecture. Opposed to the other binary types (e.g., ARM, x86), these binaries are not stripped. This implies that, for example, function names and system calls are visible, which allows us to analyze the code without the complexity that arises from stripped binaries. Not stripping the ARMv7 binaries is often observed in botnet repositories based on Mirai source code [12, 57], which is likely due to a particular bug in the cross-compiler that makes it unable to compile a stripped binary. For this reason, many Mirai forks include a special case in the Makefile to build an ARMv7 sample without stripping it. Additional information on the reverse engineering is listed in Appendix A.

3.1.1 Obtaining Samples

We analyze Gorilla botnet samples obtained from various sources. We start by identifying samples in MalwareBazaar, a repository of malware samples [1], with the first sample identified on September 5th, 2024. We have classified the samples as part of the Gorilla botnet based on the presence of the string "`gorilla botnet is on the device ur not a cat go away`" in the binary, and the C2 server that is contacted by the malware, which are unique identifiers for Gorilla samples. From this initial set, we create a Yara rule, a "filter" that searches for such unique identifiers in malware samples. We deployed this Yara rule on YARAify [2] (Appendix A.5) to be notified of new Gorilla samples. This allows us to continuously monitor new samples as they become available.

Next to proactively identifying new samples, we also perform retro-hunting, a process to search through a database of historical malware samples on existing repositories such as Hybrid Analysis [29] and VirusTotal [61]. This allows us to expand our dataset and analyze the evolution of the Gorilla botnet over time. The process of identifying malware samples based on a Yara rule relies on these samples being present in the repositories. However, some samples may simply not have been uploaded to these repositories. To address this, we also download samples directly from the botnet. We start this automated download process on April 18, 2025, and continue to download samples hourly to ensure we have the latest versions of the Gorilla botnet malware. This is possible because we can identify the malware download link through our network telescope, explained in Section 3.3. Overall, we obtain 134 unique samples.

3.2 C2 Milking

To monitor the Gorilla botnet's activities, we connect and communicate directly to its Command-and-Control (C2) server. The information obtained from reverse-engineering the Gorilla samples is used to implement a "C2 milker", which is a tool that emulates a bot and "joins the network". This tool is designed to extract commands sent by the C2 server, allowing us to monitor the botnet's activities and understand its operational patterns. Throughout our experiments, the C2 server was updated multiple times, shown in Table 2.

We implement the C2 milker in Python, using the exact connection protocols and encryption methods observed in the samples. This ensures that we can communicate with the C2 server in a way that does not involve any attack traffic from us, thus avoiding any potential legal issues or ethical concerns.

We are interested in how the botnet operates, and how bots are chosen to execute attacks. To understand whether the botmasters use a specific selection mechanism of bots for attacks (e.g., based on geolocation), we run multiple C2 milkers with different IP addresses and geolocations through the use of proxies. We run a total of five C2 milkers distributed across different continents. We find that the botmasters do not use a specific selection mechanism but indiscriminately send commands to all active bots.

Ref	Start Date	IP Address	DL	AS
S1	2024-09-06	45.202.35.64	✓	AS6079
S2	2024-10-04	154.216.19.139	–	AS17561
S3	2024-10-14	87.120.84.248	–	AS215439
S4	2024-10-20	193.143.1.70	✓	AS198953
S5	2024-12-01	94.156.227.234	–	AS214943
S6	2025-02-25	193.143.1.72	✓	AS198953
S7	2025-02-26	176.65.134.15	✓	AS214717
S8	2025-06-06	196.251.84.41	✓	AS401120
S9	2025-06-10	86.54.42.125	✓	AS42624

Table 2: Overview of C2 servers. New C2s replace the old. **Ref** refers to the labels on Figure 2. **DL** indicates that the IP also hosted the malware samples.

3.2.1 Aggregating C2 Commands to Attacks

One single "attack" event can consist of multiple commands from the C2 server: successive commands can be sent for a longer attack, and duplicated commands can occur to instruct higher-power attacks (see Section 4.2.4). To turn a log of C2 commands into distinct attacks, we use a simple grouping algorithm based on the configuration of the command. For every unique (*duration*, *vector*, *target*, *options*) tuple, we group successive commands within a 30 second time window. In other words, if a new command is issued with the exact same configuration as a previous command expired less than 30 seconds ago, we count both as part of the same attack event. We do this since, from the victim's perspective, it looks like one continuous attack. We empirically verify that continuous attacks typically occur within this 30-second window.

3.3 Network Telescope

The Gorilla botnet code is largely based on the Mirai botnet, which is known to indiscriminately scan the Internet for vulnerable devices. To capture the scanning traffic of the Gorilla botnet, we utilize a network telescope consisting of 80,000 IP addresses that are not assigned to any legitimate services. This telescope is designed to passively observe incoming traffic, allowing us to identify scanning activities.

A passive telescope does not actively send traffic. As such, it does not allow us to observe any application-layer payloads after the initial TCP SYN scanning packet from the botnet. To observe in more detail the specific exploits that are used by the Gorilla botnet, we also deploy a reactive telescope [26], which responds to the scanning traffic with a TCP SYN-ACK packet completing the TCP handshake. This allows us to capture the subsequent traffic from the botnet, including any exploit attempts. Our reactive telescope consists of 2,000 IP addresses and interacts on the entire TCP port range.

To identify which hosts are part of the Gorilla platform, we determine which hosts aim to download the Gorilla malware binary from the C2 server as part of their exploit. After identifying the Gorilla scanning infrastructure (as shown in Section 4.2.3), we can track the exploit attempts of Gorilla over time. The Gorilla platform leverages multiple exploits, including an ADB exploit, which is used to infect Android devices. The ADB exploit is particularly interesting because it contains a `wget` command that downloads the Gorilla client from the C2 server. By observing these commands, we can identify the newest download server and get the most recent botnet sample. This provides us with a continuous view of Gorilla's C2 infrastructure. An example of the ADB exploit is shown in Appendix A.6.

3.4 Netflow Analysis

In addition to the network telescope, we analyze netflow records from IXPs and ISPs to characterize the Gorilla botnet's attack activities. While netflow records can be used to estimate the attack power of the botnet, they are not completely accurate due to traffic sampling and aggregation. Additionally, providers can only observe the traffic passing their own network. However, netflow data still provides valuable information about the botnet's attack patterns and the volume of traffic generated. To ensure we obtain the closest possible estimation on the attack volume, we analyze netflow records on attacks against victims that are located within the netflow provider's network. This ensures that all attack flows have passed through the network. Because of this, we refrain from making claims about the botnet's full potential through this dataset, and only use it for validation.

3.5 Miscellaneous Data Sources

In addition to the primary methodologies outlined above, we also leverage various other data sources to enrich our understanding of the Gorilla botnet. These sources include:
Telegram Channels: We monitor Telegram channels to gain insights into the botnet's operations and marketing.
Web Scraping: We scrape the Gorilla botnet's control panel to collect information about the botnet's services and pricing, and the self-reported attack capabilities.
DDoS Power Dashboard: Public DDoS power dashboards, such as `dstat.love` [14] and `vedbex` [60], are used by users to test attack capabilities against a public endpoint. We scrape these DStat dashboards every 10 seconds and correlate the observed traffic to Gorilla botnet attack commands, allowing us to estimate the attack power of Gorilla.
Self-Attacks: We perform self-attacks against our own infrastructure to test the attack capabilities and botnet footprint of Gorilla.
Victim Testimony: We talked to victims of the Gorilla botnet to gather qualitative insights into their experiences with the botnet, including the impact of the attacks on their services and any mitigation strategies they employed.

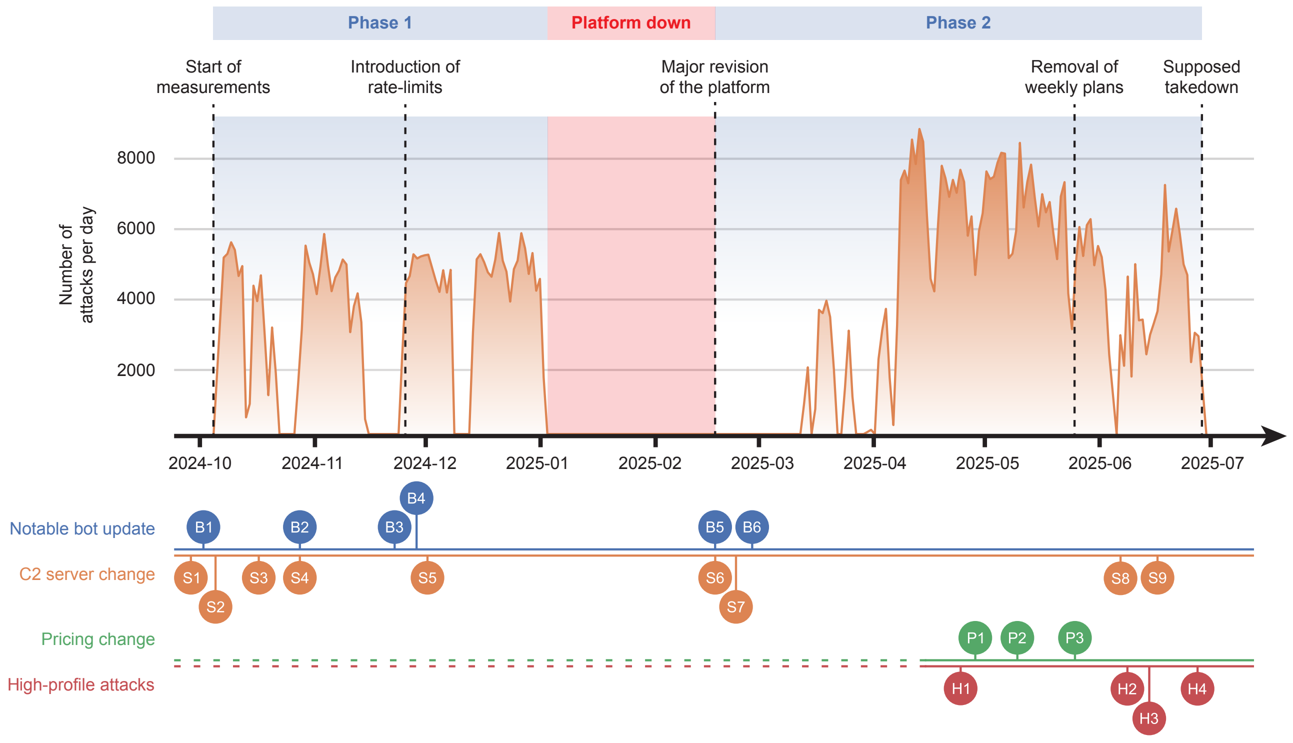


Figure 2: General timeline of activity within the Gorilla Platform.

4 The Gorilla Platform

This section provides an in-depth analysis of the Gorilla Platform by dissecting its technical and organizational aspects. Figure 2 provides a timeline of activity of the platform.

4.1 Gorilla Marketplace

The Gorilla platform, marketed as the "Gorilla Stresser" was available online via `gorillastress.st`. Here, users can buy subscriptions and start DDoS attacks. Additionally, this website is used by the maintainers to provide statistics: *attacks today*, *running attacks*, *total amount of users*, *users currently online*, and the *load per network cluster*. We find that the reporting is accurate by comparing the data from our C2 milkers to these statistics. This shows that the actors do not exaggerate their statistics for marketing purposes. In addition, the operators expose an "uptime monitor" where the status of every component of the platform can be checked. From this uptime monitor, we learn that the Gorilla platform not only relies on a botnet, but also uses centralized infrastructure to complement their DDoS attacks. This centralized infrastructure is covered in more detail in Section 6.

The control panel shows 5,355 registered users. Since e-mail addresses are never confirmed, and an account can exist without a subscription, this statistic does not represent the number of paying customers. Through the control panel, users can contact the botmasters to ask for support. Additionally, a Telegram channel is available for updates. While the channel is often removed, we record `@catmeowatyou` to have a peak of 525 subscribers. The Telegram channel is mainly used to share updates for the network, such as new methods or improved performance. Regular promotion campaigns are also published, e.g., temporary discounts. The stresser service is promoted to keep itself known among potential users. To show off attacks and their potential impact, a YouTube channel demonstrates how to perform an attack. The channel promotes mainly gaming-related attacks, showing online sessions stuttering and crashing after launching an attack through the service. Furthermore, many DStat sites include advertisement banners for stresser services, including Gorilla. Screenshots of the marketing materials are included in Appendix A.8. On May 9th, 2025, the network administrators posted a message on Telegram expressing their need for new advertisers, preferably on YouTube. We did not observe an increase in attack traffic after this request.

4.2 Evolution of the Gorilla Client

4.2.1 C2 Connection

The Gorilla botnet client includes a method called `lolxd`, used to connect to the Gorilla C2 server. Initial versions of the client included four IPs in the binary, encrypted with a cipher based on the Tiny Encryption Algorithm [63]. This encryption complicated the extraction of C2 IPs from the malware samples. Clients would decrypt and cycle through these IP addresses, until a successful connection was established with one of them over TCP on port 38242, which is hardcoded in the binary. Throughout the entire campaign, the port used for the C2 server did not change.

On October 20th, 2024 ("B2" in Figure 2), the client code changed and only a single IP address was included in the code, still encrypted with the same cipher. On November 20th ("B3"), the IP address changed in the binary to plaintext.

4.2.2 C2 Communication

The original Mirai botnet used a simple communication protocol with its C2 servers, relying on a fixed set of commands and a straightforward encoding scheme [5]. This communication scheme has been fingerprinted on a network-level, allowing defenders to proactively identify and block Mirai traffic [33]. The Gorilla botnet builds upon this foundation but, starting October 1st, 2024, introduces obfuscation to its C2 traffic.

On October 1st of 2024 ("B1" in Figure 2), the client introduced a SHA-256 hash and a byte-wise Caesar cipher for the C2 communication. The Caesar cipher is used on the original Mirai packet with an offset of 3, after which a SHA-256 hash is calculated and prepended to the message: [sha256, command]. A visual representation of this communication can be found in Appendix A.3. The goal of the SHA-256 hash is unclear. While it is used by the bots as a checksum to identify that the command is valid, it provides little extra protection against a hostile takeover, as anyone who seizes control of the C2 could craft these commands.

In addition, to make it more difficult to eavesdrop on the Gorilla C2 server, the botnet introduced a client-server authentication mechanism on the same date. To authenticate, the bot sends `01` as a ping, after which it receives a four-byte challenge. Using a hardcoded key, it decrypts another hardcoded TEA-encrypted buffer. Then, it computes the SHA-256 of the [challenge, buffer] pair. This hash is sent back as a response. If correct, the client starts to receive commands from the C2 server. An overview of the authentication handshake can be found in Appendix A.4.

While the actual attack command in the first phase was obfuscated, the underlying structure remained similar to Mirai, making it possible to write a simple detection rule. Starting with the second phase, indicated by "B5" in Figure 2, actual encryption is used to prevent detection and to obfuscate the meaning of the commands even further. The communication protocol was overhauled to be more complex, incorporating a custom Feistel cipher [36] to encrypt the commands. The full payload contains decryption parameters, SHA-256, and decryption keys, making it much harder to detect and analyze the traffic without reverse-engineering the binary. The command structure changed to: [decryption parameters, sha256, key, command], which decrypts to a regular Mirai command. This change significantly increases the complexity of the botnet's communication, disallowing anyone from identifying the actual command without extra information on the specific encryption protocol. However, if the protocol is known, only a single C2 message is needed to decrypt the command.

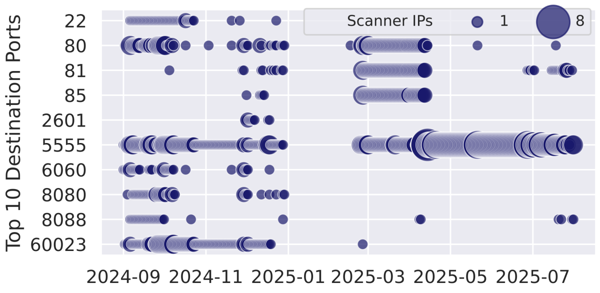


Figure 3: Scanning activity on the 10 most scanned ports.

On February 28, 2024 ("B6" in Figure 2), the client introduced a new *handle_utility_command* method. This method allows the C2 server to kill all running attacks or update the client. When a client update is desired, the C2 server can specify the download URL of the new binary. During our measurements, we have not observed this method being used.

4.2.3 Scanning

In the first phase, the individual bots scan the Internet for vulnerable Telnet devices, propagating as an Internet worm in the same way as Mirai [5, 58]. While this spreads the botnet exponentially, it also makes the bots more detectable as the scanning will be noticed by defenders. Additionally, the operators relied on centralized servers to spread the malware.

In the second phase of the Gorilla platform, shown in Figure 2, the scanning routine in the botnet clients was completely removed in favor of this centralized infrastructure. Using our telescope datasets introduced in Section 3, we are able to identify hosts aiming to exploit devices and download samples of Gorilla malware. As the reactive part of our telescope collects Application-layer information on many protocols, we can extract exploit attempts where the installed malware binary is part of Gorilla. From this analysis, we identify 29 IPs that are used to spread Gorilla infections.

Figure 3 shows the top 10 ports targeted by the scanning hosts that are part of the Gorilla platform, and the amount of hosts participating in the scans on a daily basis. There is a clear cut between the first and second phase of the Gorilla platform, with the first phase focusing on many different ports, and the second phase being much more specific towards a smaller set of ports. The malware emphatically targets ADB (TCP/5555), trying to infect vulnerable devices such as set-top boxes or routers running Android. The cut in the scanning activity shows that the Gorilla platform was not scanning for new devices to infect during the downtime in January 2025 and was thus completely offline.

4.2.4 Attack Rates and Limits

Whenever an attack command is received by a Gorilla client, it creates a child process responsible for executing the attack. When we first observed the Gorilla client, it did not contain any rate-limiting code in the clients and the bots simply generate packets as fast as possible. This means that any concurrent attack effectively reduces the attack volume as bots are splitting their attack capabilities over multiple targets, and the operators had no control over the bots' attack power.

On November 24th, 2024 (indicated by the "B4" event in Figure 2), we see an updated version of the Gorilla client that does in fact include rate-limiting. The included rate-limit is a throttle mechanism that limits the amount of packets per second (pps) sent to a victim to 3,000. Later, in April of 2025, we see an updated version of the botnet in which the rate-limit halved and attacks are capped at 1,500 pps. Interestingly, a month later, the C2 server starts sending duplicated commands. This reverted the earlier halving of the packet rate limit, putting it back at 3,000 pps. Now, to increase the attack volume, users can ask for multiple "concurrents". In essence, "concurrents" are just multiple attacks running at the same time, instructed by the C2 server simply sending the same command multiple times. This launches multiple instances of an attack, each capped at 3,000 pps.

During the entire Gorilla campaign, the C2 does not selectively include bots in an attack, but rather uses all bots in the network. We verify this by comparing the number of attacks seen by our C2 milker to the number of attacks shown in the control panel discussed in Section 4.1 and by running our milkers in different geographical locations as discussed in Section 3.2.

The operators of the platform publish the "load on the network" on their control panel, which is a measure of the number of attacks that are currently running. We initially observe that every attack introduces a 2% load on the network, meaning that the Gorilla platform would be capable of running 50 concurrents at a time. When usage was high, the operators would increase the capacity of the platform to 60 simultaneous attacks. An example Telegram update can be seen in Figure 11 in Appendix A.8. This change also reflects in their reported load statistics, every attack now accounting for 1.67% load on the network. Multiple concurrents also take up multiple attack slots in the network.

4.3 Subscriptions

Gorilla offers two types of subscriptions: "normal" and "API", the only difference being API access. With the normal plan, attacks can only be started through their WebUI. The maximum power and duration of an attack is defined by the subscription. As of May 20, 2025, the cheapest option allows users to launch one "concurrent" for 100 seconds per attack. The most expensive plan allows users to launch 20 concurrent attacks for a duration of 1,500 seconds per attack. Table 3 provides an overview of the available subscriptions. In our measurements, we observe 344 attacks that are instructed with a duration of 1,500 seconds, all requiring at least the Amethyst plan.

Name	Normal	API	Concurrents	Time(s)
Basic	$40	$60	1	100
Pro	$80	$120	2	200
Gold	$120	$180	3	300
Diamond	$160	$240	4	400
Enterprise	$200	$300	5	500
Crystal	$240	$360	6	600
Quartz	$280	$420	7	700
Sapphire	$320	$480	8	800
Emerald	$360	$540	9	900
Palladium	$400	$600	10	1,000
Grauer	$600	$900	15	1,500
Amethyst	$800	$1,200	20	1,500

Table 3: Available Gorilla plans after May 20, 2025.

On April 26, 2025 ("P1" in Figure 2), an "Alexandrite" plan was introduced, allowing users to launch 30 concurrent attacks. At the time of writing, this plan was removed. On May 8, 2025 ("P2"), the operators introduced a free tier. Free attacks were limited in power and were mainly provided as a demo for their stresser panel. A user could perform a free UDP or HTTP attack for maximum of 60 seconds. Interestingly, attacks in the free tier did not use the botnet, but relied solely on the centralized infrastructure. When instructing a free attack, the maximum bitrate we measure is 300 Mbps, which would not cause harm for most Internet connections [49].

All plans were available as daily, weekly, and monthly subscriptions. The daily and weekly options, starting at just $5, were removed on May 20, 2025 ("P3"). On the same day, the monthly plans were revised as well. The price per concurrent dropped, but the removal of the daily and weekly options raised the minimum spending amount significantly.

Subscriptions are paid with cryptocurrency. When purchasing a plan, users can choose to pay with Litecoin, USDT, Solana, or Tron. Upon ordering, a unique wallet address is generated, and we were not able to find any information on the payments on the blockchain. Payments using Bitcoin or Ethereum are only possible when creating a support ticket or contacting one of the admins on Signal directly.

5 Gorilla DDoS Attacks

During our measurement period, we observe a total of 740,351 DDoS attacks executed by the Gorilla Platform, shown in Figure 2, targeting 146,213 unique IP addresses. In this section, we provide insights into Gorilla's attack patterns, target selection, and attack characteristics.

Name	Vec.	V1	V2	Mirai	# attacks
udp_generic	0	✓	–	✓	–
udp_vse	1	✓	✓	✓	32,280
tcp_syn	3	✓	✓	✓	38,968
tcp_ack	4	✓	✓	✓	60,103
tcp_stomp	5	✓	✓	✓	10,227
gre_ip	6	✓	✓	✓	4,883
gre_eth	7	✓	✓	✓	3,713
udp_plain	9	✓	✓	✓	**246,276**
tcp_bypass	10	✓	✓	–	**135,140**
udp_bypass	11	✓	✓	–	26,275
std	12	✓	✓	–	–
udp_openvpn	13	✓	✓	–	5,346
udp_rape	14	✓	–	–	–
wra	15	✓	✓	–	6,749
tcp_ovh	16	✓	✓	–	34,984
tcp_socket	17	✓	✓	–	16,276
udp_discord	18	✓	✓	–	**62,884**
udp_fivem	19	✓	✓	–	14,346
udp_a2s	20	✓	✓	–	54,667
udp_teamspeak	21	–	✓	–	27,140
udp_samp	22	–	✓	–	441
udp_cs16	23	–	✓	–	353
utility	250	–	✓	–	–

Table 4: Attack vectors present in the bots. V1 and V2 refer to the two phases. The **bold** vectors are the most used.

5.1 Types of Attacks

As Gorilla is largely based on Mirai, which is known for its ability to perform a wide range of DDoS attacks [5], it includes all attack vectors that are available in Mirai. Additionally, Gorilla includes a number of new attack vectors. All available vectors are listed in Table 4. The added attack vectors are designed to target specific services or circumvent specific DDoS mitigations. Many of them are gaming related, such as Discord, TeamSpeak, or FiveM (a popular multiplayer modification framework for Grand Theft Auto V [17]).

While many specific vectors are available, the Gorilla platform primarily performs volumetric attacks with the *udp_plain* vector, also shown in Table 4, even when more specific attacks are available. Users can select the vector used in the attack, but the mapping of the vector as presented to the user to the actual vector is not always straightforward. Table 5 provides an overview of the mapping recorded on June 20, 2025 using the data from our self-attacks (Section 6.2). There is a striking mismatch between the selected attack vector and the vector pushed to the botnet. For example, when a user performs a "UDP bypass" attack, the service sends a "UDP plain" command to the bots, but when the user requests a "UDP fivem" attack, the bots are instructed to perform a "UDP bypass" instead. In the background, the Gorilla operators are likely adjusting the attack commands sent to the bots to optimize the attack's effectiveness.

Requested vector	Actual vector
udp_game	udp_fivem
udp_ts3	udp_teamspeak
udp_discord	udp_discord
tcp_bypass	tcp_ack
tcp_spazz	tcp_bypass
tcp_storm	tcp_ovh
network_rand	gre_ip
udp_bypass	udp_plain
udp_fivem	udp_bypass
udp_raknet	*Does not use the botnet*

Table 5: Mapping of attack vectors on June 20, 2025.

When looking at the attack types in Table 4, we can identify that there are attack vectors in use that are not bound to the methods provided to users on June 20, 2025. The mapping at this time is provided in Table 5. When looking at their Telegram updates, we find that user-facing methods change often while the methods in the botnet clients remain fairly stable, further indicating that the operator updates the attack strategies in the backend. In Appendix A.7, the usage of botnet attack vectors is shown over time.

5.2 Attack Characteristics

The Gorilla Botnet contains attack vectors for both UDP and TCP attacks, as well as some lower-level attacks. During our data collection period, we observed 60% of attacks targeting UDP services, 39% of attacks targeting TCP services, and 1% uses attacks on lower layers of the OSI model.

To understand what types of services are under attack by the platform, we look at the top 10 most targeted ports, shown in Table 6. In addition, attacks can be configured to randomize the target ports. When this is set, the C2 command sends port 0 and the bots will generate a random port every packet. We observed this in 2.5% of all attacks.

The data presented in Table 6 indicates a rather uniform distribution of target ports. While popular ports such as 80, 443, 53, and 22 stand out, all other ports are targeted in under 2% of the attacks. Many targeted ports are ephemeral ports not associated with a specific service, such as 32000, 32002, and 32003. This makes it difficult to infer what specific services are targeted by just looking at the ports. However, UDP/6672 and UDP/30120 are assigned to game servers, namely Red Dead Redemption 2 and Grand Theft Auto V respectively. Combined with the attack vectors this indicates that there is interest towards targeting gaming services.

We also consider the time at which attacks are instructed. Figure 4 shows the times during which attacks are started,

Port	Protocol	Count	Percentage (%)
80	UDP	69,744	8.93
443	TCP	43,963	5.63
80	TCP	38,141	4.88
53	UDP	35,368	4.53
22	TCP	22,080	2.83
32003	UDP	14,864	1.90
32002	UDP	14,817	1.90
6672	UDP	14,578	1.87
32000	UDP	14,506	1.86
30120	UDP	13,934	1.78

Table 6: Top 10 most targeted ports, also split by protocol.

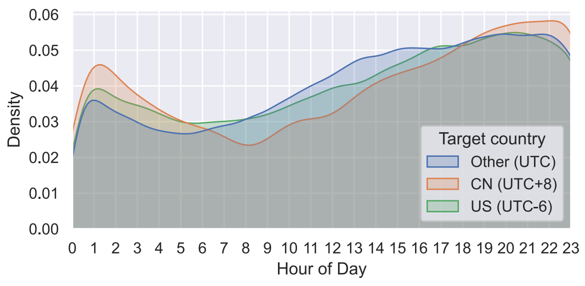


Figure 4: Probability density estimation of attack start times in local time.

grouped in major time zones. Attacks against US-based targets are shown in UTC-6, attacks against China-based targets are shown in UTC+8, and other attacks are shown in UTC. We can see that most attacks are performed in the evening from the victim's perspective, contributing to the hypothesis that many attacks are launched against gaming services in the evening.

5.3 Concurrent and Repeated Attacks

Many attacks performed by the Gorilla platform are short in duration, which is common in DDoS-for-Hire related attacks [9]. To increase the power and the duration of an attack, users have to rely on performing multiple attacks (concurrents) at the same time, and stringing together individual attacks after eachother. From the attacks collected by our C2 milker, we can identify both cases as the C2 server will simply send multiple attack commands at the same time (for running multiple concurrents), or consecutively (to increase the duration). We find that for 58% of attacks, an identical attack is launched within 30 seconds of the original attack expiring. As for running multiple concurrents, we find that 30% of the launched attacks make use of more than one concurrent. We made four observations of attacks with 20 concurrents ("H1" in Fig. 2), which aligns with the most expensive plan

AS Type	# victims	# attacks
Hosting	70,859	574,397
ISP	68,028	175,008
Business	5,684	26,613
Education	368	1,085
Government	307	1,016
Unknown	*967*	*3,232*

Table 7: Number of unique victims and attacks by AS Type.

Victim AS	Total time	AS Name
AS16276	81 days	OVH SAS
AS62041	48 days	Telegram Messenger
AS16509	40 days	Amazon.com
AS4134	37 days	ChinaNet
AS396982	32 days	Google

Table 8: Top 5 most attacked ASes in terms of total attack time rounded down to days (24h).

described in Section 4.3. Coincidentally, these attacks took place within three days without overlap and all targeted hosting providers, indicating that a single actor might be behind these attacks. We observed the longest attack, lasting 24 hours, on June 27th, targeting a hosting provider ("H4").

5.4 Attack Targets

Gorilla is a DDoS-for-Hire platform, which means that it is rented out to users who can select their own targets. When analyzing the 146,213 unique targets, we indeed observe that Gorilla is used to attack a wide range of victims, including gaming, educational, and government. We classify the victims based on the mappings of IPInfo [30], which provides information on the type of Autonomous System (AS) that the victim belongs to. The results are shown in Table 7. Most attacks are targeted at hosting providers (48%) and ISPs (46%), mainly targeting gaming platforms. However, we also observe a significant number of attacks on business (4%), education (0.3%), and government (0.2%) ASes.

Looking at the total attack time per target AS, we see some ASes that are disproportionally targeted compared to others. Table 8 shows the top 5 of ASes that were attacked most in terms of attack time. From this data, we can see that OVHcloud has been attacked for over 81 days during our measurement period. Given that our data spans 265 days, OVHcloud was actively under attack for over 30% of the time.

Looking at the configured attack duration per target AS type in Figure 5, we can see that attacks targeting Government and Education tend to have a higher duration configured compared to attacks at other AS types. This indicates that attacks targeting Government and Education institutions gen-

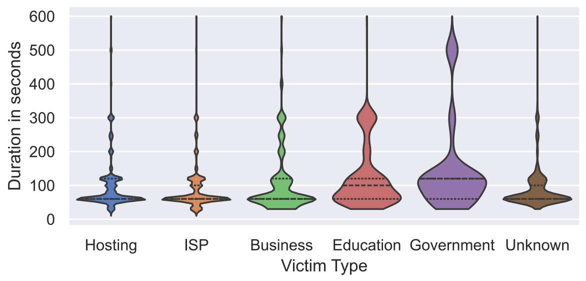


Figure 5: Duration in attack command per AS type.

Country	# attacks
US	164,617
CN	90,200
NL	80,104
DE	70,376
BH	31,932

Table 9: 5 most targeted countries for all AS types

Country	# attacks
US	458
AE	285
RU	264
ID	256
TH	181

Table 10: Top 5 countries (Education and Government)

erally require a more expensive subscription. Table 9 shows the 5 most targeted countries, with the United States being the most targeted country, followed by China and the Netherlands. This distribution is likely influenced by the large number of hosting providers and gaming services in these countries [46]. When removing these categories and focusing on just Government and Education target ASes, we observe a different distribution, as shown in Table 10. The United States is still targeted the most, but we can see that other countries such as the United Arab Emirates and Russia are also heavily targeted.

5.4.1 High Profile Attacks

While most attacks are aimed at gaming services, we also observe a number of successful attacks against high-profile targets. While these attacks are less frequent, they demonstrate the capability of a stresser service to impact a wide range of organizations. In this section, we discuss two high-profile attacks that were performed in June 2025. We correlate press releases with attack commands observed in our C2 milker. We are not able to identify who would be responsible for these attacks, but we can provide some insight based on the data. Targeted organizations experienced significant disruptions to their services as a result of the attacks, showing the impact of DDoS-for-Hire services beyond gaming related attacks [34, 41].

Roularta Media Group ("H2"): On June 10, 2025, the Roularta Media Group was targeted by a series of DDoS attacks [23]. These attacks occurred between 06:21 UTC and 15:45 UTC, and consisted of volumetric UDP attacks targeting ports 80 and 300 on their public-facing firewall. In our C2 milker, we identify multiple "UDP plain" commands aimed at their network. We contacted the organisation, and they shared a recorded bandwidth of 4Gb/s and 3Mpps. This traffic was generated through 3 'concurrents' of 300 seconds per attack, but were repeated throughout the duration of the attack. This attack setup points at the "gold" plan, which would cost a user $120 per month as shown in Section 4.3. The attacks disrupted normal operations and impacted users trying to access Dutch and Belgian news sites and apps. The printing facility was also affected, causing delays in newspaper and magazine deliveries. During the response, the attack was geoblocked, but this was ineffective as the attack appeared to move with every geoblock, likely saturating the uplink and seeing a different part of the attack when blocking out part of the traffic.

SVT (Swedish Public Service) ("H3"): The Swedish Public Service broadcaster SVT was targeted by a series of DDoS attacks in June 2025, lasting on-and-off for more than a week [11]. In this attack, SVT experienced multiple waves of TCP traffic targeting ports 443 and 18264, with the first attack starting on June 8, 2025, and the last command being sent on June 16, 2025. The attacks were characterized by their duration, with some attacks lasting up to 1,000 seconds. For these attacks, a user would have to buy the "palladium" plan of $400, which allows for longer attacks and higher bandwidth.

6 Measuring the Gorilla Infrastructure

To identify why the attacks performed using the Gorilla stresser are so potent, we aim to map the elements participating in a Gorilla DDoS attack. In this section, we identify (1) the attack capabilities by leveraging the attacks of Gorilla against public DStat services, and (2) the elements that contribute to the attacks by attacking our own infrastructre.

6.1 Attack Capabilities

To attract users to the platform, DDoS-for-Hire operators actively market their networks on Telegram. They apply two strategies: (1) attack a specific service after which the generated attack volume is placed in a leaderboard in the channel, or (2) share screenshots of a website going offline due to one of their DDoS attacks. Example screenshots of such Telegram promotion techniques can be seen in Appendix A.8.

We scrape multiple websites that provide servers that can be attacked to measure the attack power of a botnet, as described in Section 3. The Gorilla network also attacks these endpoints regularly, either by the operators or users that want to test the power of the network, and we observe 2,000 Gorilla attacks targeting these services. From this data, we observe the attack power at certain points in time, and observe a maximum firepower of almost 13 Gbps per concurrent. On May 1st, 2025, we observe UDP attacks using three concurrents. From the corresponding DStat result, we find a peak bitrate

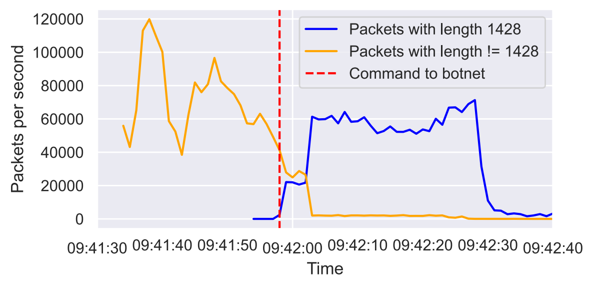


Figure 6: Observed packets based on packet length.

of 41 Gbps. On May 18th, 2025, we observe a UDP attack directed at a different endpoint with five concurrents. The attack caused a peak bandwidth of 24.5 Gbps, which saturated the reported 25 Gbps uplink of the endpoint. Due to this saturation, the actual attack bandwidth is likely higher. The DStat attacks do not reliably indicate the attack power of the Gorilla platform, as it fluctuates often.

6.2 Attacking Controlled Infrastructure

To obtain a ground-truth on what constitutes an attack of the Gorilla platform, we attack our own infrastructure in consultation with our local ISP. This allows us to do a full network dump of the traffic. The data collected in this experiment is not complete as our local ISP performs DDoS scrubbing, limiting the packets we observe in the attacks. We thus cannot use this information for a full picture of the attack, but instead use it to identify a lower-bound on the number of active bots, the characteristics of an attack, and a mapping of attack options in the control panel to specific attack types in the botnet. We perform 12 attacks, covering every attack type available on the platform as of June 20, 2025. All attacks target a different IP address to account for transient effects.

When starting an attack we immediately see traffic coming into our system. However, the command sent to the botnet is delayed, and we only observe the command in our C2 milkers roughly 30 seconds after the start of the attack. Figure 6 plots an attack over time, where we show when the C2 milker first received an attack command by the vertical red line. From the botnet client code, we know that bots send packets of exactly 1,428 Bytes. This allows us to identify the traffic originating from the botnet. The Figure shows that the bots indeed attack for the specified duration of the attack (30 seconds) after receiving the command from the C2 server. However, before the attack command is sent to the bots, other traffic is generated for the duration of the attack (30 seconds). When inspecting this traffic, we find a mix of DNS amplification, accounting for 35% of these packets, and generic UDP packets with spoofed source IPs. Looking into the payloads of these packets, we observe DNS responses to an `ANY` query for the `collectd.org` domain name. This returns around 1,200 bytes of data. The UDP packets with randomized sources contain small amounts of random data in the payload. We hypothesize that this traffic originates from centralized servers rather than the bots, as it requires IPs to be located in a network that allows spoofing [7, 16]. Additionally, the Gorilla uptime monitor states the status of a so-called "spoofed" network.

The Gorilla platform thus appears to have a more complex attack infrastructure than Mirai, likely because the botnet does not contain many hosts and is not capable of generating the attack power required to make the platform effective. During the attacks on our own infrastructure, we only observe 220 bots participating in the attack, which would generate 7 Gbps of attack traffic, as every bot would theoretically generate 33.6 Mbps of traffic per concurrent (due to rate limits and payload sizes). However, the attacks observed on the DStat endpoints show that the platform is capable of generating 10+ Gbps of traffic per concurrent, requiring more than just the botnet. As we observe all commands in all of our geographically dispersed C2 milkers, it is unlikely that we only observe a portion of the botnet. This shows that, while the botnet is not as large as others, the network is very effective.

To identify whether the bots that we observe in the attacks are indeed generating 3,000 packets per second, and not being throttled by the devices that they are running on, we analyze the packet payloads of the bots. To perform a UDP attack, the bot generates random payloads using the same RNG as Mirai. We rely on a similar methodology as Griffioen et al. [21], where the random data generated by the RNG is identifiable. When four consecutive random values are observed from the RNG, all next values can be predicted. This means that we can identify exactly which packets were dropped by our upstream provider. A detailed explanation of this methodology can be found in Appendix C. When a packet is received, we can analyze its payload to identify the offset in the RNG stream. By tracking these offsets over time, we can infer the packet transmission rate at the sender and compare it to the actual reception rate at the measurement point. A transmission rate of approximately 3,000 packets per second is observed for the bots, which aligns with the previously observed rate-limit in Section 4.2.4.

6.3 Netflow Measurements

The attacks targeting our own infrastructure described in Section 6 revealed the existence of a distributed attack infrastructure employed by the Gorilla Platform, composed of (1) a botnet and (2) centralized infrastructure. While the former generates attack traffic without spoofing its source IP address, the latter performs more stealthy attacks either by directly sending spoofed packets or leveraging amplification attacks. This attack model is crucial when measuring traffic routed through IXPs and ISPs. For the entire duration of June 2025 we correlate the commands milked from the active C2.

We find that netflows can provide an accurate lower bound on the attack traffic, but are not a good proxy for identifying the attack sources. This is due to the operators not only leveraging bots, but also hosts that attack with spoofed source IP addresses. This means that when counting sources participating in an attack, we are both overestimating due to the spoofed sources (attacks record more than 100,000 sources), and undersampling due to the limited sampling rate of netflows (we only observe a fraction of the actual traffic). To identify the common hosts participating in the attacks, we apply an intersection between the sets of source IP addresses targeting separate attack victims over the same time window. This approach allows us to filter out the spoofed sources while keeping the common origin hosts, and at the same time remove legitimate traffic, since it is unlikely that a non-malicious host would connect to multiple DDoS victims at the same time. We choose to group the data over time windows of five minutes to limit measurement errors due to IP churn. This approach results in netflow aggregates originating from both botnet clients and reflection hosts. These two can be further distinguished by looking at the source ports, which will be in the ephemeral range for botnet clients, and limited to well-known service ports for amplification [50].

Applying this methodology allows us to identify an average of roughly 100 unique hosts per time window during the last month of activity of the Gorilla Platform. A time series is visible in Appendix B. While these numbers are lower than what we observe through the self-attacks on June 20th, they provide a reliable baseline. The spoofed component from the centralized attacks is not measurable with this approach, requiring more sophisticated traffic fingerprinting methods.

7 Related Work

DDoS stresser services have been studied in the past, with a focus on their business models, attack capabilities, and the underlying botnet architectures. Previous research has highlighted the evolution of DDoS-for-Hire platforms, their targets, and their potential power [27, 35, 47, 51, 54, 55].

A major focus of previous work has been on the analysis of specific botnets, such as Mirai [5], which has served as a foundation for many subsequent DDoS botnets, including Gorilla. Antonakakis et al. [5] provided an early analysis of the Mirai botnet, examining its architecture, propagation mechanisms, and attack capabilities. Their work laid the groundwork for understanding how modern DDoS botnets operate and how they can be mitigated. In a followup work, Griffioen et al. [21] show how the descendants of Mirai compete for market share. Affinito et al. [3] further explored how the variants of Mirai spread through the Internet, focusing on the infection vectors and the techniques used to compromise IoT devices.

After the leak of the Mirai source code, many botnets have adopted similar architectures and attacks, leading to a proliferation of DDoS services that share common characteristics [3, 19, 39]. This led to botnets implementing similar protocols and attacks [18, 39], and competing for market share [21].

In addition to the analysis of botnet architectures, previous research has also investigated the techniques used by DDoS botnets to evade detection and mitigation [4, 6, 10, 38]. This includes the use of encryption, obfuscation, and sophisticated C2 protocols to avoid detection by security systems.

Other studies have examined stresser platforms and their impact, exploring how these services affect the DDoS landscape [22, 25, 53, 55]. Jonker et al. [32] show that a large part of the Internet has been victimized by a DDoS attack at some point, and that the number of attacks has been increasing over time. They also highlighted the role of stresser platforms in facilitating these attacks, making it easier for adversaries to launch large-scale campaigns without needing to maintain their own botnet infrastructure. Hutchings et al. [28] report that before Mirai and botnets were weaponized, DDoS attacks were often performed using centralized servers. Other works have also reported on DDoS attacks and mitigations without botnets [13, 31, 40, 56]. Kopp et al. [37] identify how effective takedown of DDoS services can be, and how the DDoS-for-Hire business model is resilient to such take-downs, which is further analyzed by Vu et al. [62]. Taking down these services is not trivial [20, 42, 43].

Modern stresser services, such as Gorilla, are less studied in the literature. While based on Mirai, the additional features and attack capabilities of modern stresser services have not been extensively analyzed. Recent works have investigated the descendants of Mirai [3], but these studies focus on the malware's propagation and infection mechanisms rather than the DDoS-for-Hire business model and attack capabilities.

This paper is the next in a line of research that aims to provide a comprehensive understanding of DDoS services [25, 28, 32, 37, 52, 55], by focusing on the Gorilla stresser. By analyzing the Gorilla botnet, we shed light on the current state of DDoS-for-Hire platforms, their attack capabilities, and the market behind their operation. Furthermore, we provide insights into how these networks and their attacks can be effectively monitored and what data is needed, contributing to the broader understanding of DDoS threats.

8 Discussion

Gorilla is a new generation of Mirai-based DDoS-for-Hire platform that has been very successful in launching high-profile attacks. The developers of Gorilla leveraged the knowledge gained over a decade in developing and deploying Mirai-based botnets, while also being quite aggressive in advertising their products and engaging customers of the DDoS platform. Gorilla offered a spectrum of options to their customers in terms of functionality and pricing, and utilized different attack vectors, some off-path, e.g., UDP reflection attacks, and others on-path, e.g., TCP attacks.

Research and Development Cycle of Gorilla: Although the Gorilla software originates from the Mirai software, written a decade ago, small additions and changes to the code have made it more popular and effective. While Mirai boasted eight attack vectors, the latest version of Gorilla offers 25 functionalities. This indicates that the operators of the Gorilla Botnet continuously improve their offerings, causing them to be highly successful. The addition of an update functionality for the bots indicate that the operators do not treat their devices as disposable anymore.

Gorilla implemented two major software updates, adding additional attack vectors (functionalities) while phasing out unused functionalities, making it more robust over time. While in the first Gorilla software release phase, updates were quite frequent, in the second phase, there is only one major update. This indicates that the changes in cryptographic communication during the second phase had a significant impact on the platform's resilience. This also shows that the developers of Gorilla learned from their mistakes that made it easy to reverse engineer the C2-bot communication and commands. While the Gorilla network had a rather small botnet, the addition of centralized infrastructure made its attacks more powerful.

Resilience of Gorilla: While previous networks struggled to maintain their customer base and attack power when switching C2 servers [62], this was not the case for Gorilla. In the first phase, it took only a couple of days for the Gorilla operator to reach the same level of attacks after a C2 switch. In the second phase, there was practically no interruption; the number of switches between C2 servers was minimal. This shows that the Gorilla platform owners have become more selective about where to install and operate their C2 servers.

This complicated a takedown of the Gorilla network, but on June 28, 2025, law enforcement supposedly succeeded. While there has not been an official statement from law enforcement, the Gorilla Telegram channel became inactive, and the supporting infrastructure went offline shortly after.

Marketing Strategies: The updates in pricing in the second phase also contributed to the Gorilla's success, making it possible to attract a diverse range of customers with different budgets. The operators of Gorilla frequently demonstrated its success in high-profile attacks, provided free attack capabilities, and even offered discounts for follow-up attacks. This indicates that the success of Gorilla was not solely a technical achievement. The Gorilla operators not only offered a full-fledged attack platform for non-experts, but were very involved in attracting new users and offering competitive pricing. Users do not have to resubscribe after a C2 update [62], which contributes to the loyalty of existing users and high demand for the DDoS platform.

Limitations: In this study, we focus on one successful DDoS-for-Hire platform, the Gorilla Botnet. Although this is a clear example of successful engineering and marketing in the DDoS-for-Hire market, we still cannot confirm whether such decisions have been or will be adopted by other DDoS-for-Hire platforms and if they will lead to the same success.

Moreover, although we did our best to utilize data from different sources to assess the efficacy of the network, it is possible that we do not have the complete view of its firepower. Our analysis reports a lower bound of its DDoS impact, which nevertheless shows that Gorilla was responsible for high-profile DDoS attacks during its operation.

9 Conclusion

The Mirai botnet architecture continues to evolve, having created a new market for DDoS-for-Hire built on top of its original design and software. In this paper, we reverse-engineer one of the most successful Mirai-based DDoS-for-Hire platforms, the Gorilla Botnet. Our analysis reveals that the operators have learned from the shortcomings of Mirai design, deployment, and operation, offering a resilient and efficient proposition that quickly garnered the trust and attention of DDoS customers. While much of the Gorilla network relies on the one-decade-old original software, changes in C2 communication, the separation of scanning and attack infrastructure, the C2 server hosting selection, and the addition of new functions as a response to users's demand have made Gorilla one of the most successful DDoS-for-Hire platforms to date. We also notice that the pricing schemes and marketing strategies played a significant role, as the operators made efforts to improve engagement among its existing users and attract new ones. As the DDoS-for-Hire ecosystem continues to evolve, we would like to evaluate how the methodology for reverse-engineering the Gorilla platform can be applied to analyze similar platforms in the future and understand which, sometimes subtle, differences in engineering and operation can make them successful and profitable.

10 Acknowledgments

This work was supported by the Dutch Research Council (NWO) under the ADAPTIve project and by the European Commission under the Horizon Europe Programme as part of the project SafeHorizon (Grant Agreement #101168562).

Ethical Considerations

In this paper we describe in detail the inner workings of a botnet. While our research aims to advance the understanding of DDoS-for-Hire services and their impact, we acknowledge the potential ethical implications of disclosing such information. We have taken care to anonymize any sensitive data and to focus on the technical aspects of the botnet's operation, rather than on specific individuals or organizations. In this section, we assess the ethical impact for the stakeholders.

Users of the Gorilla stresser: Since we observe the stresser from the perspective of the botnet, we do not come across

any personally identifiable information of the users of the network.
The Gorilla stresser platform: We have only conducted experiments that did not require any active engagement with the botnet, except from the attacks on our own infrastructure. Other than during these self-attacks, we did not actively engage with the botnet and we did not attempt to disrupt or interfere with their activities. Our goal was to simply observe and analyze the botnet's behavior. To conduct the self-attacks, we have bought the cheapest tier without API access. The botnet analyzed in this paper was operational from 2024 until June 2025, and we have not observed any activity since then. The botnet's C2 server went offline on June 28, 2025, and we believe it is no longer active. Therefore, the operators of this botnet will not learn from the publication of our findings. However, we recognize that the techniques and insights gained from this research could potentially be used by other malicious actors to improve their own botnets. To mitigate this risk, we have focused on the technical aspects of the botnet's operation and have avoided disclosing any sensitive information that could be adopted by malicious actors.
Attack victims: When using netflow data for verification of attacks, the data was collected by network operators for operational purposes. All analysis on the netflow data was performed live and not stored as raw data. All analysis was performed on-premises, and no unanonimized data was shared outside of the network operator. We received only aggregated and anonimized data, e.g.,IP addresses were salted and hashed with a secret key that is not known to us. The shared artifacts of this work contain solely anonimized data, as to not expose victim IP addresses.
Infected devices: While performing attacks on our own infrastructure, many bots send data as part of the attack. To ensure anonymity of these devices and their owners, we anonimized all source IPs in the data with the Crypto-PAN algorithm [64].
Self-attack upstream networks: The DDoS attacks performed on our own infrastructure were done with full knowledge and consent from our upstream Internet Service Provider. We obtained permission from the national police agency. While setting up the attack, we chose the smallest and shortest attack possible as to not potentially overload our upstream provider or any network further up the data path. We targeted our own infrastructure and did not target any third-party services.
National Cyber Security Center: The "C2 milker" we developed in this work was shared with the National Cyber Securiy Center, so they could use the gathered intelligence to attribute attacks targeting local victims.

Open Science

We support the principles of Open Science and have made our research as transparent as possible. This includes sharing our datasets and methodologies to enable reproducibility and further research in the field. We encourage other researchers to build upon our work and to contribute to the collective understanding of DDoS-for-Hire services and botnets. From the datasets listed in Table 1, we share the artifacts of all data sources used except the netflows and web scraping data.

We provide the Gorilla botnet's command logs (from the C2 milker), analyzed malware sample hashes, and data on the scanning infrastructure observed by our reactive telescope, including exploit payloads. We also make available selected Telegram messages displaying the advertisement strategies of the Gorilla operators. To preserve the anonymity of the victims of the attacks, we anonymize all IP addresses collected by the C2 milker through the Crypto-PAN algorithm [64].

In addition, we share the network traffic logs from our self-attacks. To preserve the anonymity of infected devices in the network, we anonimized the source IPs in this data with the Crypto-PAN algorithm as well. To preserve anonymity of our own infrastructure, all destination IPs are redacted.

All shared artifacts of this work are available for reference at https://doi.org/10.5281/zenodo.17886098.

References

[1] Abuse.ch. Malware Bazaar. `https://bazaar.abuse.ch/`, 2025. [Online; accessed 21-August-2025].

[2] Abuse.ch. Yaraify. `https://yaraify.abuse.ch/`, 2025. [Online; accessed 21-August-2025].

[3] Antonia Affinito, Stefania Zinno, Giovanni Stanco, Alessio Botta, and Giorgio Ventre. The evolution of Mirai botnet scans over a six-year period. *Journal of Information Security and Applications*, 79:103629, December 2023.

[4] Dennis Andriesse, Christian Rossow, Brett Stone-Gross, Daniel Plohmann, and Herbert Bos. Highly Resilient Peer-to-Peer Botnets Are Here: An Analysis of Gameover Zeus. In *8th International Conference on Malicious and Unwanted Software*, pages 116–123. IEEE, 2013.

[5] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. Understanding the Mirai Botnet. In *USENIX Security*, pages 1093–1110, 2017.

[6] Manos Antonakakis, Roberto Perdisci, Yacin Nadji, Nikolaos Vasiloglou, Saeed Abu-Nimeh, Wenke Lee, and David Dagon. From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware. In *USENIX Security*, pages 491–506, 2012.

[7] Robert Beverly and Steven Bauer. The Spoofer Project: Inferring the Extent of Source Address Filtering on the Internet. In *USENIX SRUTI*, volume 5, pages 53–59, 2005.

[8] H.L.J. Bijmans and M.S.C. van Leuken. No Time to Choose: Leveraging Internet Scans to Determine IoC Lifetimes. In *2024 IEEE International Conference on Big Data*, pages 2586–2595. IEEE, 2024.

[9] Norbert Blenn, Vincent Ghiëtte, and Christian Doerr. Quantifying the Spectrum of Denial-of-Service Attacks through Internet Backscatter. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, pages 1–10, 2017.

[10] Leon Böck, Emmanouil Vasilomanolakis, Max Mühlhäuser, and Shankar Karuppayah. Next Generation P2P Botnets: Monitoring under Adverse Conditions. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 511–531. Springer, 2018.

[11] European Conservative. Massive Cyberattacks Target Swedish State Broadcaster. https://europeanconservative.com/articles/news-corner/massive-cyberattacks-target-swedish-state-broadcaster/, 2025. [Online; accessed 26-August-2025].

[12] DDOS-project. Cybernet DDoS Project. https://github.com/DANO-AMP/DDOS-project/blob/8fb4aff98c6983a81e1f37229df827f0f042d0a0/Botnet/mirai-source/cybernet/build.sh#L30, 2025. Online; accessed 26-August-2025.

[13] Christos Douligeris and Aikaterini Mitrokotsa. DDoS Attacks and Defense Mechanisms: A Classification. In *Proceedings of the 3rd IEEE International Symposium on Signal Processing and Information Technology*, pages 190–193. IEEE, 2003.

[14] dstat.love. dstat.love. https://dstat.love/, 2025. [Online; accessed 25-August-2025].

[15] ENISA. ENISA Threat Landscape 2024. https://www.enisa.europa.eu/sites/default/files/2024-11/ENISA%20Threat%20Landscape%202024_0.pdf, 2024. [Online; accessed 21-August-2025].

[16] Paul Ferguson and Daniel Senie. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. RFC2827, 1998.

[17] FiveM. FiveM: Multiplayer Modification Framework for Grand Theft Auto V. https://fivem.net/, 2025. [Online; accessed 21-August-2025].

[18] Getoar Gallopeni, Bruno Rodrigues, Muriel Franco, and Burkhard Stiller. A Practical Analysis on Mirai Botnet Traffic. In *IFIP Networking Conference*, pages 667–668. IEEE, 2020.

[19] Metehan Gelgi, Yueting Guan, Sanjay Arunachala, Maddi Samba Siva Rao, and Nicola Dragoni. Systematic Literature Review of IoT Botnet DDOS Attacks and Evaluation of Detection Techniques. *Sensors*, 24(11):3571, 2024.

[20] Dimitrios Georgoulias, Jens Myrup Pedersen, Morten Falch, and Emmanouil Vasilomanolakis. Botnet Business Models, Takedown Attempts, and the Darkweb Market: A Survey. *ACM Computing Surveys*, 55(11):1–39, 2023.

[21] Harm Griffioen and Christian Doerr. Examining Mirai's Battle over the Internet of Things. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pages 743–756, Virtual Event USA, October 2020. ACM.

[22] Harm Griffioen, Kris Oosthoek, Paul van der Knaap, and Christian Doerr. Scan, test, execute: Adversarial tactics in amplification ddos attacks. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pages 940–954, 2021.

[23] Roularta Media Group. Temporary disruptions at Roularta Media Group due to Cyberattack. https://www.roularta.be/en/about-roularta/press-releases/temporary-disruptions-roularta-media-group-due-cyberattack, 2025. [Online; accessed 26-August-2025].

[24] hex rays. IDA Pro: Powerful Disassembler, Decompiler and Debugger. https://hex-rays.com/ida-pro, 2025.

[25] Raphael Hiesgen, Marcin Nawrocki, Marinho Barcellos, Daniel Kopp, Oliver Hohlfeld, Echo Chan, Roland Dobbins, Christian Doerr, Christian Rossow, Daniel R Thomas, et al. The age of DDoScovery: An Empirical Comparison of Industry and Academic DDoS Assessments. In *Proceedings of the 2024 ACM on Internet Measurement Conference*, pages 259–279, 2024.

[26] Raphael Hiesgen, Marcin Nawrocki, Alistair King, Alberto Dainotti, Thomas C. Schmidt, and Matthias Wählisch. Spoki: Unveiling a New Wave of Scanners through a Reactive Network Telescope. In *USENIX Security*, pages 431–448, Boston, MA, August 2022.

[27] Nazrul Hoque, Dhruba K Bhattacharyya, and Jugal K Kalita. Botnet in DDoS Attacks: Trends and Challenges. *IEEE Communications Surveys & Tutorials*, 17(4):2242–2270, 2015.

[28] Alice Hutchings and Richard Clayton. Exploring the Provision of Online Booter Services. *Deviant Behavior*, 37(10):1163–1178, 2016.

[29] Hybrid Analysis. Hybrid Analysis Tool. https://hybrid-analysis.com/, 2025.

[30] IPInfo. IP Enrichment Information. https://ipinfo.io/, 2025. [Online; accessed 25-August-2025].

[31] Ghafar A Jaafar, Shahidan M Abdullah, and Saifuladli Ismail. Review of Recent Detection Methods for HTTP DDoS Attack. *Journal of Computer Networks and Communications*, 2019(1):1283472, 2019.

[32] Mattijs Jonker, Alistair King, Johannes Krupp, Christian Rossow, Anna Sperotto, and Alberto Dainotti. Millions of Targets under Attack: A Macroscopic Characterization of the DoS Ecosystem. In *Proceedings of the 2017 Internet Measurement Conference*, pages 100–113, 2017.

[33] Đorđe D Jovanović and Pavle V Vuletić. Analysis and Characterization of IoT Malware Command and Control Communication. In *2019 27th Telecommunications Forum (TELFOR)*, pages 1–4. IEEE, 2019.

[34] Mohammad Karami and Damon McCoy. Rent to Pwn: Analyzing Xommodity Booter DDoS Services. *Usenix login*, 38(6):20–23, 2013.

[35] Mohammad Karami and Damon McCoy. Understanding the Emerging Threat of DDoS-as-a-Service. In *6th USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2013.

[36] Lars R Knudsen. Practically Secure Feistel Ciphers. In *International Workshop on Fast Software Encryption*, pages 211–221. Springer, 1993.

[37] Daniel Kopp, Matthias Wichtlhuber, Ingmar Poese, Jair Santanna, Oliver Hohlfeld, and Christoph Dietzel. DDoS Hide & Seek: On the Effectiveness of a Booter Services Takedown. In *Proceedings of the Internet Measurement Conference*, pages 65–72, 2019.

[38] Chaz Lever, Platon Kotzias, Davide Balzarotti, Juan Caballero, and Manos Antonakakis. A Lustrum of Malware Network Communication: Evolution and Insights. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 788–804. IEEE, 2017.

[39] Artur Marzano, David Alexander, Osvaldo Fonseca, Elverton Fazzion, Cristine Hoepers, Klaus Steding-Jessen, Marcelo HPC Chaves, Ítalo Cunha, Dorgival Guedes, and Wagner Meira. The Evolution of Bashlite and Mirai IoT Botnets. In *IEEE Symposium on Computers and Communications*. IEEE, 2018.

[40] Jelena Mirkovic and Peter Reiher. A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2):39–53, 2004.

[41] Asier Moneva and Eric Rutger Leukfeldt. Interest in Booter Services and Distributed Denial of Service Attacks: Insight from Google Search Data. *European Journal of Criminology*, 22(4):508–533, 2025.

[42] Yacin Nadji, Manos Antonakakis, Roberto Perdisci, David Dagon, and Wenke Lee. Beheading Hydras: Performing Effective Botnet Takedowns. In *Proceedings of the ACM SIGSAC conference on Computer & communications security*, pages 121–132, 2013.

[43] Yacin Nadji, Roberto Perdisci, and Manos Antonakakis. Still beheading hydras: Botnet takedowns then and now. *IEEE Transactions on Dependable and Secure Computing*, 14(5):535–549, 2015.

[44] National Security Agency and developers. Ghidra - Powerful Open-Source Reverse Engineering Tool. http://ghidra.net/, 2025.

[45] NCSC. GorillaBot: A new DDoS-for-hire service. https://www.ncsc.admin.ch/dam/ncsc/de/dokumente/dokumentation/fachberichte/NCSC-CH-GorillaBot.pdf.download.pdf/NCSC-CH-GorillaBot.pdf, 2025. [Online; accessed 25-August-2025].

[46] Newzoo. Top 10 Countries by Game Revenues. https://newzoo.com/resources/rankings/top-10-countries-by-game-revenues, 2025. [Online; accessed 21-August-2025].

[47] Arman Noroozian, Maciej Korczyński, Carlos Hernandez Gañan, Daisuke Makita, Katsunari Yoshioka, and Michel Van Eeten. Who Gets the Boot? Analyzing Victimization by DDoS-as-a-Service. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 368–389. Springer, 2016.

[48] NSFocus. Over 300,000 GorillaBot: The New King of DDoS Attacks. https://nsfocusglobal.com/over-300000-gorillabot-the-new-king-of-ddos-attacks/, 2025. [Online; accessed 25-August-2025].

[49] World Population Review. Internet Speeds by Country 2025. https://worldpopulationreview.com/country-rankings/internet-speeds-by-country, 2025. [Online; accessed 21-August-2025].

[50] Christian Rossow. Amplification Hell: Revisiting Network Protocols for DDoS Abuse. In *NDSS*, pages 1–15, 2014.

[51] José Jair Santanna, Ricardo O De Schmidt, Daphne Tuncer, Joey De Vries, Lisandro Z Granville, and Aiko Pras. Booter Blacklist: Unveiling DDoS-for-Hire Websites. In *2016 12th International Conference on Network and Service Management (CNSM)*, pages 144–152. IEEE, 2016.

[52] José Jair Santanna, Joey de Vries, Ricardo de O. Schmidt, Daphne Tuncer, Lisandro Z. Granville, and Aiko Pras. Booter List Generation: The Basis for Investigating DDoS-for-Hire Websites. *International journal of network management*, 28(1):e2008, 2018.

[53] José Jair Santanna, Romain Durban, Anna Sperotto, and Aiko Pras. Inside Booters: An Analysis on Operational Databases. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 432–440. IEEE, 2015.

[54] José Jair Santanna and Anna Sperotto. Characterizing and Mitigating the DDoS-as-a-Service Phenomenon. In *IFIP International Conference on Autonomous Infrastructure, Management and Security*, pages 74–78. Springer, 2014.

[55] José Jair Santanna, Roland van Rijswijk-Deij, Rick Hofstede, Anna Sperotto, Mark Wierbosch, Lisandro Zambenedetti Granville, and Aiko Pras. Booters—An Analysis of DDoS-as-a-Service Attacks. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 243–251. IEEE, 2015.

[56] Saeed Shafieian, Mohammad Zulkernine, and Anwar Haque. CloudZombie: Launching and Detecting Slow-read Distributed Denial of Service Attacks from the Cloud. In *IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pages 1733–1740, 2015.

[57] Soerungmakara. Soerungmakara. https://github.com/soreungmakara2/Soerungmakara/blob/9cf36ea9b11c3792210af6c3d18986c4a9678db4/build.sh#L30, 2025. Online; accessed 26-August-2025.

[58] Eugene H. Spafford. The Internet Worm Program: An analysis. *ACM SIGCOMM Computer Communication Review*, 19(1):17–57, 1989.

[59] Rui Tanabe, Tsuyufumi Watanabe, Akira Fujita, Ryoichi Isawa, Carlos Gañán, Michel van Eeten, Katsunari Yoshioka, and Tsutomu Matsumoto. Disposable Botnets: Long-term Analysis of IoT Botnet Infrastructure. *Journal of Information Processing*, 30:577–590, 2022.

[60] Vedbex. Vedbex: Live DSTAT. https://www.vedbex.com/dstat, 2025. [Online; accessed 26-August-2025].

[61] VirusTotal. VirusTotal Tool. https://www.virustotal.com/, 2025.

[62] Anh V Vu, Ben Collier, Daniel R Thomas, John Kristoff, Richard Clayton, and Alice Hutchings. Assessing the Aftermath: the Effects of a Global Takedown against DDoS-for-hire Services. In *Proceedings of USENIX Security*, 2025.

[63] David J Wheeler and Roger M Needham. TEA, a Tiny Encryption Algorithm. In *International Workshop on Fast Software Encryption*, pages 363–366. Springer, 1994.

[64] Jun Xu, Jinliang Fan, M.H. Ammar, and S.B. Moon. Prefix-preserving IP Address Anonymization: Measurement-based Security Evaluation and a New Cryptography-based Scheme. In *10th IEEE International Conference on Network Protocols, 2002. Proceedings.*, pages 280–289, 2002.

A Gorilla Botnet Technical Information

A.1 Reverse Engineering

To understand the inner workings of the Gorilla botnet, we reverse engineered the malware samples in detail. We started by collecting multiple samples of the Gorilla malware from various sources, including honeypots and public repositories. We then used a combination of static and dynamic analysis techniques to dissect the malware. The static reverse-engineering involved disassembling the binaries using Ghidra [44] and IDA Pro [24] to analyze the code structure, functions, and algorithms used. Dynamic analysis was performed in a controlled environment using Docker without Internet.

Throughout the reverse engineering process, we specifically look at the code of the malware that is responsible for the C2 communication, the attack vectors, and the persistence mechanisms. By looking at differences in the binaries and reverse-engineering samples over time, we identified when the malware was updated.

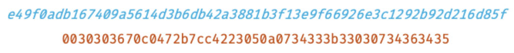
A.2 Gorilla Botnet Persistence Techniques

Once malware is loaded on a victim's device, it aims to establish persistence and make it hard for defenders to clean up the device. These measures were not in place in the original Mirai source code. First, it loops over all processes visible from `/proc` and kills all of them, except for itself. Then, it symlinks the malware binary to `/bin/sh`, effectively replacing the default shell with itself. Additionally, it unlinks binaries like `shutdown`, `reboot`, `poweroff` and `halt`. Until 2024-10-04, the malware also wrote a `systemd` service that downloaded and executed the malware binary upon system startup.

After infection, the following string is appended to `/etc/motd`: "*gorilla botnet is on the device ur not a cat go away*". This message is shown to users when they log in to the device, and it is unclear why the malware authors chose to include this message, as it can be used to indicate that the device is compromised. To hinder automated analysis of the malware, it also contains some simple sandbox detection techniques. First, it checks whether the `/proc` filesystem exists. Afterwards, it checks the presence of a debugger by checking the `TracerPid`. Lastly, it detects a Kubernetes environment by checking the `/proc/1/cgroup` file. If any of these checks fail, the malware refuses to run.

A.3 C2 Messages

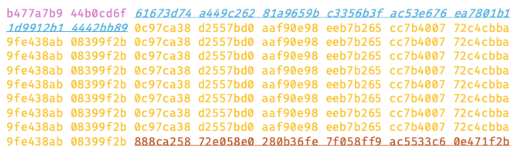
The Gorilla Botnet has two seperate command structures between the two phases. The first command structure is a concatenation between a SHA256 hash and a Mirai command encoded with a ceasar cipher shifting all bytes with an offset of 3 as shown in Figure 7.



```
e49f0adb167409a5614d3b6db42a3881b3f13e9f66926e3c1292b92d216d85f
0030303670c0472b7cc4223050a0734333b33030734363435
```

Figure 7: Command of the botnet in the first phase.

In the second phase, the C2 commands use a Feistel Cipher to encrypt the command, as shown in Figure 8. The command is sent as: [decryption parameters, sha256, key, command].



```
b477a7b9 44b0cd6f 61673d74 a449c262 81a9659b c3356b3f ac53e676 ea7801b1
1d9912b1 4442bb89 0c97ca38 d2557bd0 aaf90e98 eeb7b265 cc7b4007 72c4cbba
9fe438ab 08399f2b 0c97ca38 d2557bd0 aaf90e98 eeb7b265 cc7b4007 72c4cbba
9fe438ab 08399f2b 0c97ca38 d2557bd0 aaf90e98 eeb7b265 cc7b4007 72c4cbba
9fe438ab 08399f2b 0c97ca38 d2557bd0 aaf90e98 eeb7b265 cc7b4007 72c4cbba
9fe438ab 08399f2b 0c97ca38 d2557bd0 aaf90e98 eeb7b265 cc7b4007 72c4cbba
9fe438ab 08399f2b 0c97ca38 d2557bd0 aaf90e98 eeb7b265 cc7b4007 72c4cbba
9fe438ab 08399f2b 0c97ca38 d2557bd0 aaf90e98 eeb7b265 cc7b4007 72c4cbba
9fe438ab 08399f2b 0c97ca38 d2557bd0 aaf90e98 eeb7b265 cc7b4007 72c4cbba
9fe438ab 08399f2b 888ca258 72e058e0 280b36fe 7f058ff9 ac5533c6 0e471f2b
```

Figure 8: Command of the botnet in the second phase.

A.4 C2 Handshake

After successfully establishing a TCP connection with the C2 Server, a challenge-response exchange takes place, illustrated in Figure A.4. From the protocol scheme described in Section 4.2.2, learning both key values while reverse-engineering the malware samples enables the Bot emulation performed through the Milker.

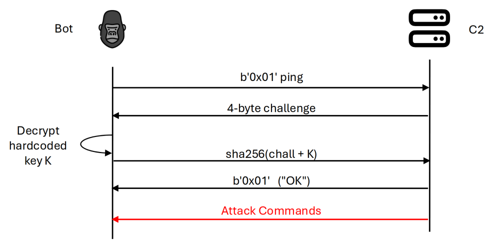


Figure 9: Handshake protocol between Bot and C2 Server.

A.5 YARA Rule

To identify Gorilla malware samples, we created a YARA rule based on a unique string found in the binary. This rule can be used to scan files and identify those that match the characteristics of the Gorilla botnet malware.

```
rule GorillaBotnet {
    strings:
        $a = "gorilla botnet is on the device
            ur not a cat go away"
    condition:
        $a }
```

A.6 Gorilla ADB packet

Sample sequence of TCP-based data payloads targeting ADB:

```
b'CNXN\x00\x00\x00\x01\x00\x00\x04\x00\x1b\x00\x00
     \x00M\n\x00\x00\xbc\xb1\xa7\xb1host::features=
     cmd,shell_v2',
b'OPENX\x01\x00\x00\x00\x00\x00\x006\x04\x00\x00\
     xc2T\x01\x00\xb0\xaf\xba\xb1shell:cd␣/data/
     local/tmp/;␣rm␣*;␣busybox␣wget␣http
     ://176.65.134.15/arm.nn;␣chmod␣+x␣arm.nn;␣./
     arm.nn␣arm.android;␣busybox␣wget␣http
     ://176.65.134.15/arm5.nn;␣chmod␣+x␣arm5.nn;␣./
     arm5.nn␣arm5.android;␣(...)\x00'
```

A.7 Attack Vector Usage

The Gorilla Botnet employs a variety of attack vectors, as detailed in Section 5. The most commonly used vectors include `udp_generic`, `tcp_ack`, and `udp_discord`. These vectors are selected by the user based on the target service and the desired impact of the attack, but mapped differently in the backend. The daily usage of these attack vectors is illustrated in Figure 10, showing the frequency of each vector.

A.8 Gorilla Promotion

The Gorilla operators often post updates in their Telegram channel, explaining changes to their platform. An example of such a message is shown in Figure 11. In this message, the operators share that they increased the attack capacity of the network. Additionally, they introduce a new free tier to their platform, and ask for advertisers on YouTube.

The Gorilla operators maintained a YouTube channel where attacks were showcased. The attacks were often targeted at

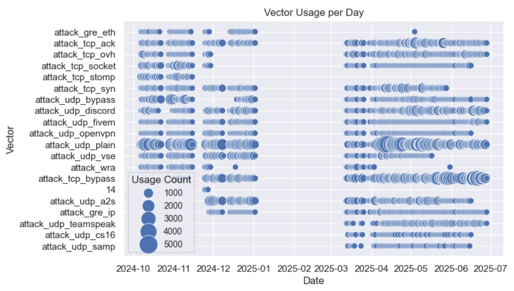
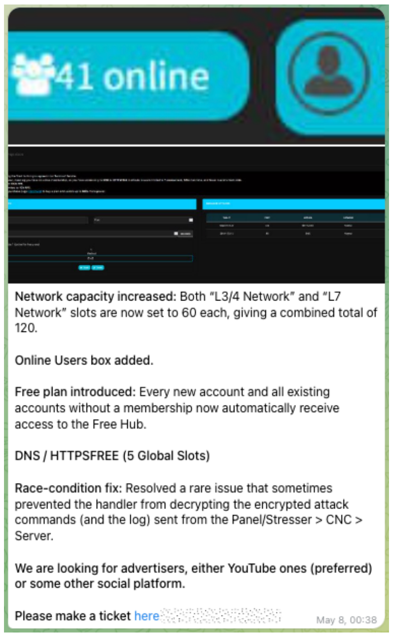


Figure 10: Daily usage of different attack vectors.


Network capacity increased: Both "L3/4 Network" and "L7 Network" slots are now set to 60 each, giving a combined total of 120.

Online Users box added.

Free plan introduced: Every new account and all existing accounts without a membership now automatically receive access to the Free Hub.

DNS / HTTPSFREE (5 Global Slots)

Race-condition fix: Resolved a rare issue that sometimes prevented the handler from decrypting the encrypted attack commands (and the log) sent from the Panel/Stresser > CNC > Server.

We are looking for advertisers, either YouTube ones (preferred) or some other social platform.

Please make a ticket here

May 8, 00:38

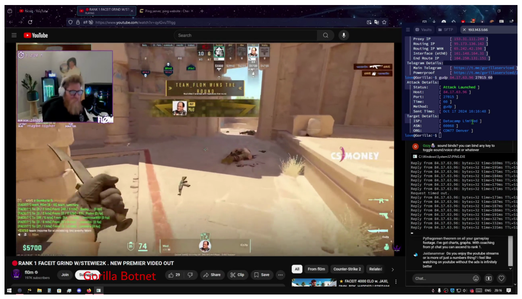
Figure 11: Telegram message promoting more attack slots, new free plan, and inquiring for YouTube advertisers.

gaming sessions that were live-streamed, as can be seen in Figure 12a. Additionally, some attacks were targeting DStat endpoints and showcased the attack power of the network. An example of such a DStat attack can be seen in Figure 12b.

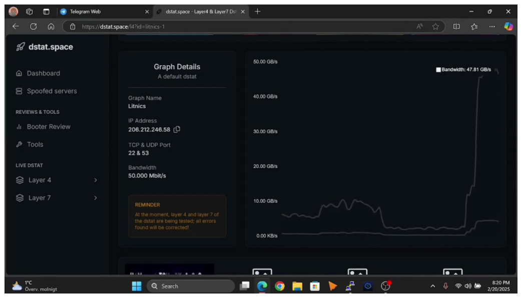
Many DStat websites include advertisements for different stresser services. Gorilla was often shown on these websites, with their own banner, shown in Figure 13.

B Netflow Intersection Timeseries

We correlate attack traffic by intersecting common IPs over two separate attacks within 5-minute windows. This allows us to identify overlapping traffic patterns and shared infrastructure between attacks. The resulting timeseries is shown in Figure 14. While we find that netflows are ineffective for assessing the size of a botnet, they can still provide valuable insights into the behavior and characteristics of the attack traffic.



(a) A screenshot of a YouTube video showing an attack against a live online gaming session.



(b) The DStat panel showing 47 Gbps attack bandwidth.

Figure 12: Screenshots of YouTube videos demonstrating the attack power of the Gorilla platform.

Figure 13: A frame from the Gorilla banner often shown on DStat websites for promotion.

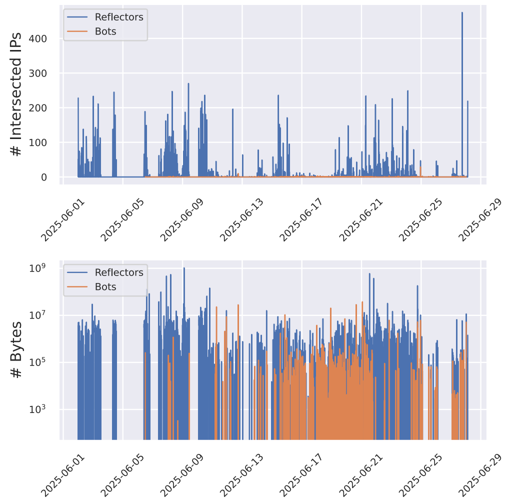


Figure 14: Timeseries of source IP and byte counts inferred by intersecting common IPs over separate attacks within 5-minute windows.

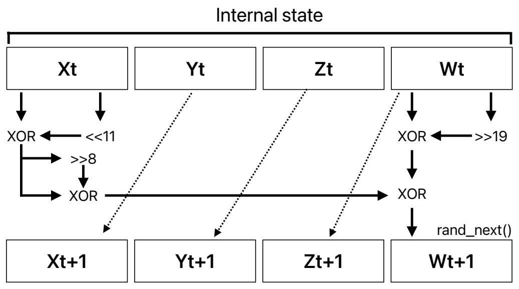


Figure 15: Mirai RNG system.

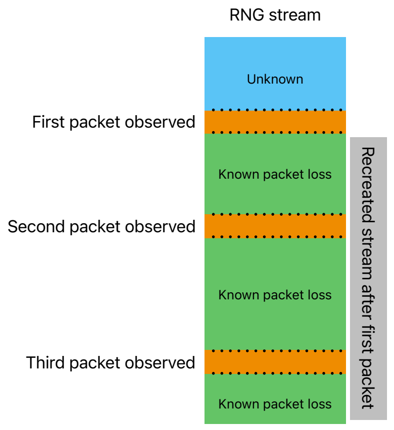


Figure 16: Mapping packets to RNG offsets.

C Identifying Random Number Generator Patterns

To identify the random number generator (RNG) patterns used by the Gorilla botnet, we analyze the random data generated by the bots during attacks. By observing the payloads, we can identify the RNG algorithm used and its characteristics. Gorilla's RNG is similar to Mirai's, which is based on the Linear Congruential Generator (LCG) algorithm. This algorithm is known for its simplicity and speed, but it has predictable patterns. By analyzing the random values generated by the bots, we can identify these patterns and use them to our advantage. A schematic overview of the RNG is shown in Figure 15.

During a `UDP_BYPASS` attack, each bot fills payloads with pseudo-random data generated by the Mirai RNG. A sequence of four consecutive four-byte integers produced by this RNG is sufficient to predict the next bytes. Analysis confirms that the payloads of packets generated by a single bot correspond to a continuous segment of this RNG output stream. As a result, each arriving UDP packet can be uniquely mapped to a specific offset within the RNG stream. This offset effectively reflects the index of the originally transmitted packet. This mapping process is illustrated in Figure 16.

With this process, we can identify which packets are dropped on the network because we can verify which packets have been sent by the bot as the RNG will be advanced to a next state. By tracking the increase in these RNG offsets over time, we can infer the packet transmission rate at the sender and compare it to the actual reception rate at the measurement point. A transmission rate of approximately 1,500 packets per second is observed, which aligns with the hardcoded limit in the malware sample. The ratio of received to transmitted packet rates allows us to estimate the experienced packet loss. This is visualized for a single bot in Figure 17.

When looking at the arrival time of packets together with their corresponding RNG offsets, we observe a staircase pattern, as shown in Figure 18. This pattern arises from the bot's behavior of transmitting up to 1,500 packets at the start of each second, followed by a period of inactivity for the remainder of that second. This bursty transmission behavior is reflected in the plateaus of Figure 18. Bots with constrained upstream bandwidth, which are unable to transmit 1,500 packets within a second, do not exhibit this staircase pattern, as they will not have the period of inactivity. Instead, their packet transmissions are more evenly distributed over time, resulting in a more linear increase in RNG offsets without the distinct plateaus seen in the staircase pattern.

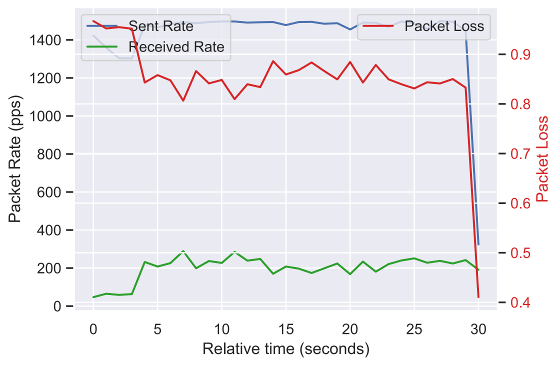


Figure 17: Packet loss based on received and transmitted packet rates.

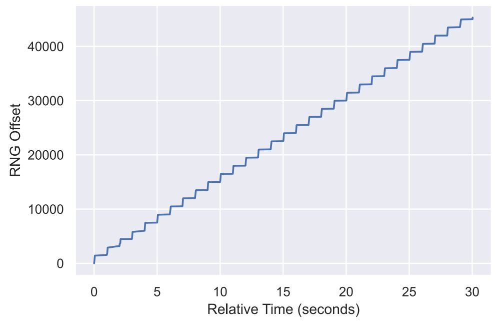


Figure 18: RNG offset producing a staircase pattern based on packet arrival times.