Exploring The State of Rate Limiting in IPv6

PIETER VAN HEIJNINGEN, University of Twente, The Netherlands

The rapid adoption of IPv6 and the availability of large address blocks pose new challenges for rate limiting in computer networking. Traditionally, rate limiting has been implemented on a per-IP basis in IPv4 networks. However because of the extensive address space in IPv6, users are assigned large ranges of address, rendering per-IP rate limiting inadequate. This paper aims to investigate the current state of rate limiting in IPv6 by addressing several research questions. Firstly, the implementation of IPv6 rate limiting in open-source software will be examined. Secondly, the approaches taken by various cloud providers in implementing IPv6 rate limiting will be analyzed. Additionally, the paper explores the policies of organizations involved in address assignment and their rationale behind the size of address blocks assigned to users. The methodology involves examining documentation and conducting local stress tests. The paper provides new insights into the current state of IPv6 rate limiting, given the scarcity of research in this area, although concretely addressing all research questions may be challenging.

1 INTRODUCTION

In computer networking, rate limiting is the act of limiting the amount of requests that a given user can perform in a predefined span of time. This is done to prevent a single attacker from overloading the server and/or to prevent web scraping.
Traditionally, with IPv4, rate limiting is implemented on a per-IP basis. For the most part, this works fine, due to the limited amount of IPv4 addresses available. Which makes getting access to a large quantity to use is hard and/or expensive.
However with IPv6, this changes. For example RIPE NCC recommends that end users are assigned multiple /64 blocks of IPv6 addresses [36]. To elaborate further on why this causes an issue: A /64 block of IPv6 addresses gives the user access to 2^{64} distinct addresses. So theoretically, if an attacker would cycle through his entire /64 block for every request, he could send half a million requests every microsecond for a year before the server would see the same IPv6 address again. Clearly per-IP rate limiting for IPv6 is not sufficient.
IPv6 adoption is steadily growing, this can for example be seen by Google's user statistics. Which shows an increase in the percentage of users that access their services over IPv6. With a current adoption rate of 45% [19].
Therefore, with more and more of the Internet supporting IPv6 this issue only becomes more important.

2 BACKGROUND

In this section relevant background information in order to understand this paper will be provided.

TScIT 39, July 7, 2023, Enschede, The Netherlands

2.1 CIDR notation

When talking about ranges of IP addresses, CIDR notation is used. This is a slash, followed by the number of bits in the address that are fixed.
For example the University of Twente owns the 2001:067c:2564::/48 IPv6 address range. As IPv6 addresses are 128 bits, this gives them access to 2^{128-48} distinct addresses. All with the same fixed beginning.

2.2 Tunnel brokers

IPv4 to IPv6 tunnel broker services as defined by RFC 3053 can provide the service of IPv6 connectivity for users who otherwise do not have access to IPv6 [16].
The existence of these services are of importance to this paper as it makes it trivial for an attacker to have access to a large IPv6 address range. For example Hurricane Electric offers /48 tunnels all over the world for free [15].

2.3 Address assignment

Who gets to own and control which IPv6 address range is decided by a hierarchical structure of organisations. At the top is IANA, who assigns address space to 5 regional internet registries (RIR), who in turn assign their address space to local internet registries (LIR), then to internet service providers (ISP). Who finally assign to end users [22, 23].
In practice, the distinction between LIRs and ISPs is a bit blurred. Mainly because they both receive an Autonomous System Number (ASN) from their overarching RIR. This AS number is used to assign, and own IPv4 and IPv6 address ranges.

2.4 DoS attacks

One interesting aspect of this topic is its usage in Denial-of-service attacks. Which usually involves one or many (DDoS) computers sending requests to a server in an attempt to make the service inaccessible for regular users. Currently, most large scale DDoS attacks are done over IPv4, and make use of IP spoofing [31]. However the large ranges of IP addresses available under IPv6 could remove the need for IP spoofing. Making DDoS attacks harder to detect and more sophisticated.

2.5 Web servers, WAFs and reverse proxies

Various software products that implement rate limiting will be tested, and these software products fall under different categories, which will be explained in this section.
First, we have web servers. Web servers are primarily used to handle incoming requests and deliver web content to users. Most web servers also implement rate limiting mechanisms.
Quite often a reverse proxy is deployed between the web server and the open internet. Reverse proxies are commonly used for load balancing purposes or to add an additional layer of security, such as with rate limiting. It's worth noting that some web servers, like

NGINX and Apache, can also act as reverse proxies.
To enhance the security of web applications, Web Application Firewalls (WAFs) are often employed. WAFs serve as an extra layer of defense against common vulnerabilities such as SQL injection and cross-site scripting (XSS). Additionally, WAFs may also implement distributed denial-of-service (DDoS) prevention mechanisms, which could be considered rate limiting. However, by design often exact implementation details can be hard to determine.

3 PREVIOUS WORK

One interesting paper is one from Frank Li and David Freeman [30]. In the paper they research Facebook's users who use IPv6 vs who use IPv4. For us the interesting part is the research they did on rate limiting. Where, through statistical analysis, they found that typically a single IPv4 address behaves most similar to an entire /48 or /56 IPv6 subnet when looking at a user level. In other words, if a user only uses one IPv4 address he behaves most similar to a user using a large IPv6 subnet. So they suggest rate limiting at this level. A recent blog post from Adam Pritchard further emphasises the important of this issue [42]. He suggests as a solution to rate limit per /64 block instead of per-IP or even to disable IPv6 all together. However he notes both of these solutions are far from perfect.

Finally a paper from Maximilian Golla et al [32]. Here they explore the state of rate limiting in general, specifically focused at brute-forcing passwords. In this paper it is made clear that exclusively focusing on IP-based rate limiting in their case is not sufficient, but other factors like cookies or session tokens should also be considered.

4 PROBLEM STATEMENTS

As is evident by the scarcity of academic papers about this topic and Adam's blog post [42]. This subject is under-researched. Therefore the focus of this paper will be quite broad. With the following research question:

What is the current state of HTTP rate limiting on the web when utilising IPv6?.

This can be answered with the following sub-questions:

- **RQ1:** How do various open-source web servers, WAFs and proxies implement IPv6 rate limiting?
- **RQ2:** How do various cloud providers implement IPv6 rate limiting?
- **RQ3:** At the various organisations handling address assignment, what are the policies on the size of block that gets assigned to users? And what is the reasoning behind them?
- **RQ4:** How to effectively rate limit IPv6 requests?

During the research, all of these will be considered. But during the research it will be decided which research questions are looked at more in-depth.

5 METHODOLOGY

5.1 Answering RQ1

For various open-source web servers, Web Application Firewalls and proxies we will either look at their code base to see how rate limiting is implemented. Or alternatively perform local stress tests on our own machine to determine this.

When relevant, also the memory usage of the software will be analysed. The theory is that when attacking a server with lots different IPv6 addresses, the server's memory could be exhausted by trying to keep track of all addresses for rate limiting purposes. For example storing every address in a /64 IPv6 block would take $2^{64} * 128$ bits of memory/storage, or 295.1 exabytes.

5.1.1 Performing requests. To perform these tests, the following test machine will be used:
A desktop computer running Ubuntu server 22.04 as its operating system, which has the piece of software to be tested installed. The specifications of this machine are as follows: 12 core Ryzen 5 3600 CPU (max 4.2 GHz), 16 GB of 3200 Mhz DDR4 RAM, and a 500GB Samsung 960 EVO SATA SSD.
On the same machine, local requests to the software to be tested will be performed using a bash command utilising freebind [11] as described in Listing 1

Listing 1. Script to bluk send requests

```
ip -6 route add local fd66:1234:5678:90ab
    ::/64 dev lo
time yes "url=localhost" | head --lines
    REQ_NUM | freebind --random fd66
    :1234:5678:90ab::/64 -- curl --http1.1 -6
     -H "Connection: close" --parallel --
    parallel-immediate --parallel-max 0 --
    config -
```

First, an iptables route is set up to assign an arbitrary /64 block of the unique local IPv6 unicast address [20]. Effectively simulating a usable /64 block, which can only access localhost.
For the second command, everything is ran under *time*, which measures the time it took to execute all the requests. The command first generates an infinite list of lines containing `"url=localhost"` using *yes*. Then this is piped to *head*, which limits the number of lines by REQ_NUM, thus in turn limiting the number of requests to be made. Then this is piped to *freebind*, which makes use of the entire previously assigned prefix to randomise which /128 address of the /64 block is used for each request using `--random`. Finally requests are all sent in parallel using *curl* with the following flags: `--parallel`, to enable requests to be performed in parallel instead of in series, `--parallel-immediate`, to always make it open a connection, instead of waiting for another connection to become available, `--parallel-max 0`, to allow unlimited parallel connections.
Then finally `--http1.1 -H "Connection: close"` to send the Connection: close header with each request, to allow a new socket connection to be opened every time. Thus allowing for a new IPv6 address, for this header to work the HTTP/1.1 standard is required

One downside of this command is that the long list of URLs generated by *yes* are stored in memory, which occupies multiple gigabytes of system memory for large REQ_NUM.

5.1.2 Monitoring system performance. To monitor the impact on system performance the rate limiting implementation has, the *pidstat* program from the *sysstat* package will be used [45]. Which will be used to log CPU and memory usage at an interval of 1 second using a bash script [49].

5.2 Answering RQ2

We will look at their publicly available documentation for various cloud providers to see how they have implemented it.

5.3 Answering RQ3

We will research the various organisations who are responsible for addressing and look at their publicly available policies and documentation. For technical reasoning we will investigate further into the IPv6 RFCs.
Finally we will verify the real-world implementation of these policies by statistically analysing a IPv6 hitlist, which is a list that contains all currently active IPv6 addresses.

5.3.1 Attacker model. When answering this research question, various types of attackers will be taken into account. The first type will be an attacker with access to a normal internet connection, like his home network. Secondly, an attacker with access to an AS number, which for many RIRs is quite trivial to get access to, including RIPE NCC. Where any organisation or person can obtain an AS number through some verification and payment [39].
Finally nation-state attackers will be considered.

5.4 Answering RQ4

We will use the information from the previous research questions to determine if it is possible to have an effective rate limiting IPv6 algorithm.

Once all these questions have been answered to some extend, it should give a satisfactory overview of the current state of IPv6 rate limiting on the web.

6 RESULTS

6.1 RQ1

We decided to research the most widely-used open-source web servers, as measured by % websites that use them world wide according to w3techs [50]. All web servers will be tested on two aspects, first on what rules are implemented and secondly how their implementation would handle keeping track of many IPv6 addresses in memory. A summary of the results can be found in table 1.

6.1.1 NGINX (34.4% of the web). NGINX, according to w3techs is currently the most popular publicly facing piece of web software. Note NGINX can function as either a web server directly or as a reverse proxy.
As can be publicly found in the NGINX documentation, default rate limiting behaviour on IPv6 is per-IP [41]. However the *key* variable in *limit_req_zone* can be modified to allow blocking for /64, or larger subnets [29].

Software	Importance	Implementation
NGINX	34.4% market share	Per-IP by default
Apache	32.1% market share	Third-party, Per-IP
Microsoft-IIS	5.5% market share	Per-IP
Node.js	2% market share	Third-party, Per-IP
Caddy	47.8k GitHub stars	No defaults given
HAProxy	3.8k GitHub stars	Per-IP by default
Traefik	43.5k GitHub stars	Per-IP by default

Table 1. RQ1 summary

Performance impact. The impact on performance of the rate limiting solution was analysed. As a baseline, first a test was run with the default NGINX configuration, which had no rate limiting enabled. This test had a REQ_NUM of 50 million, which it completed in 56 minutes and 21 seconds. Processing requests at a consistent rate of 887 thousand a minute.
Then, rate limiting was enabled using the *ngx_http_limit_req_module* [41]. utilising the configuration described in Listing 2

Listing 2. NGINX config snippet

```
limit_req_zone $binary_remote_addr zone=
    mylimit:1000m rate=1r/m;
server {
        listen 80 default_server;
        listen [::]:80 default_server;

        limit_req zone=mylimit;
```

To show the biggest effect, the strictest possible rate limiting was implemented, only allowing 1 request per minute per IP. Furthermore, inside of *limit_req_zone* the zone size can be specified, which is the maximum memory size used for rate limiting purposes. According to the official documentation 128 bytes is used to store a single rate limiting state [41]. Knowing that, plus the previously measured consistent rate of 887 thousand requests a minute. We can expect about 113.5 megabytes of added memory usage from the rate limiting solution. The zone size set at 1000 megabytes should therefore be sufficient.
This configuration was applied, the access.log file was cleared, and NGINX was restarted. And rate limiting was confirmed to be working. Then the same test, of 50 million requests was performed again, which completed in 56 minutes and 54 seconds. The method of evading rate limiting rules utilising freebind did indeed succeed, as all 50 million requests completed successfully. While only 60 requests should have been allowed from the same source.
The impact the the rate limiting solution had can be seen in the found in figure 1 and 2

Code and raw data that was used to generate these plots can be found in the accompanying repository [49].
A couple of things can be seen from the graphs:
First of all in the CPU graph, CPU usage keeps increasing over time, and will likely keep increasing. But given that this also occurs with

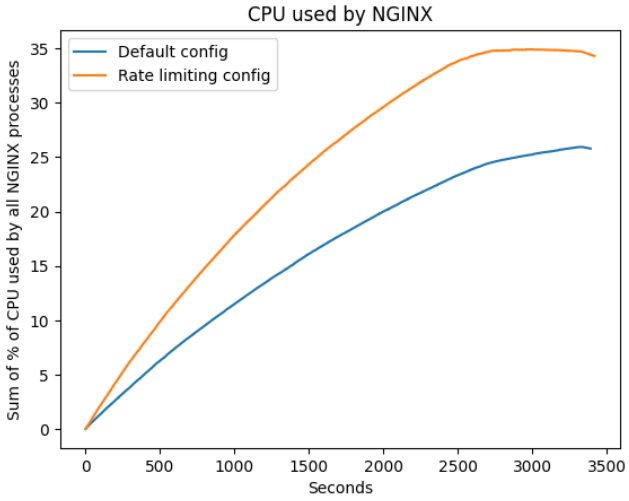

Fig. 1. NGINX CPU graph

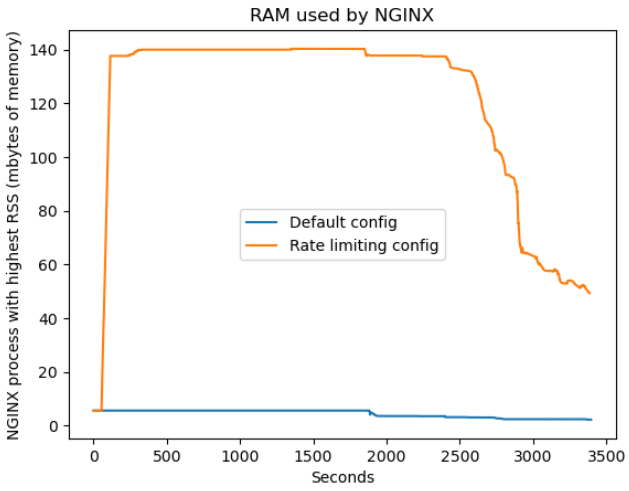

Fig. 2. NGINX memory graph

the default configuration, although at a lesser extreme. While interesting, due to this reason, we consider it out of scope for further investigation into this rate limiting topic.
Secondly, the predicted 113.5 megabyte increase can be clearly seen in the memory usage graph. Confirming what was stated in the documentation. However the steep drop off at the 2500 second mark with the rate limiting configuration is strange, this also coincides with a slow down of CPU usage in the other graph at the same time. To confirm that this was not a fluke, multiple test were run, which produced similar results.

In conclusion, it seems, given a reasonable amount of server system memory, that it not possible to saturate all a said memory with a poorly configured NGINX rate limiting set up by a single attacker. This is mainly due to the maximum time interval that can be set up is one minute, and the attacker, even if he operates on localhost, cannot send that many requests in that timeframe to even saturate more than 140 mbytes of memory.

6.1.2 Apache (32.1% of the web). Same as with NGINX, Apache is a popular alternative supporting many of the same features, thus it can function as a web server or a reverse proxy.
While Apache does not have official support for rate limiting, various third-party modules where researched. Which practically all do per-IP rate limiting. Testing the memory aspect did not seem relevant as most third-party modules are either outdated or deprecated.

6.1.3 Microsoft-IIS (5.5% of the web). Microsoft-IIS is Microsoft's webserver, which can also act as a reverse proxy.
Publicly available documentation for Microsoft's IIS server states rate limits are implemented on a per-IP basis [33].

6.1.4 Node.js (2% of the web). Same case as with Apache, there are various third-party packages that allow for rate limiting. But virtually none look at more than per-IP for IPv6.

6.1.5 Caddy. No usage statistics could be found on Caddy, but it is the most popular project on github with the tag web-server at 47.8 thousand stars, therefore it was relevant to discuss [18]. In their documentation it can be found that there's two modules you can use to implement rate limiting [12]. Both of them have easy support for blocking per address range instead of per-IP, but worryingly no recommendation on how to handle IPv6 is given in the documentation. Plus the RussellLuo/caddy-ext module has a default zone_size, as in the number of addresses that are stored in memory simultaneously, of 10 000. Which is insufficient in preventing IPv6 attacks.

6.1.6 HAProxy. Is a popular reverse proxy server solution used by many high-traffic websites [21]. With currently 3.8 thousand stars on GitHub. An official blog post that gives four examples on how to implement rate limiting makes no mention of blocking per IP range nor of IPv6 [43]. Therefore we think it is reasonable to assume that no proper IPv6 rate limiting is supported by default.

6.1.7 Traefik. Is another popular reverse proxy server solution used by many high-traffic websites [46] with 43.5 thousand stars on GitHub. Their public documentation has an extensive page on how to configure rate limiting, but no mention of IPv6 or IP ranges. Therefore per-ip rate limiting for IPv6 can be assumed [47].

6.1.8 WAFs. Investigation into various popular, open source WAFs was done, including ModSecurity, Naxsi and IronBee. But it was found that rate limiting seems to generally not be in the scope of WAFs responsibilities. Instead it is often implemented at the reverse proxy or web server level. Part of their solutions does include DDoS protection, but for obvious reasons exact implementation details are not shared.

To conclude, in the open-source space clearly very little is known about the impact that IPv6 can have on rate limiting. As is shown by us not being able to find a single open-source web server that does anything more than per-IP rate limiting for IPv6 by default.

6.2 RQ2

6.2.1 Cloudflare. Today one of the largest cloud providers is Cloudflare, with around 80% of websites utilising a CDN network using them and/or their Web Application Firewall (WAF) [26]. How they handle IPv6 rate limiting is pretty clear, they block it per /64 range as stated by their official documentation [25].

6.2.2 AWS. Another big cloud provider is Amazon's AWS. As is also mentioned in Adam Pritchard's blog post, they don't seem to mention anything regarding how they specifically handle IPv6. However it is indeed concerning that their WAF can only block up to 10 000 IP addresses [10, 42].

6.2.3 Others. To keep things concise, various other cloud providers where researched, which all make no mention of how they handle IPv6 in their public documentation of their WAFs. This list includes; Azure, IBM Cloud, Google Cloud and Alibaba cloud [4, 14, 24, 34]. Exact implementation details are therefore anyone's guess.

6.3 RQ3

6.3.1 Recommendations from RFC and RIRs. RFC6177 states some best practices regarding how RIRs should make recommendations on the IPv6 address block size are assigned to end users [44]. As a result of that, RIPE NCC, LACNIC, APNIC and AFRINIC have more or less the same policy on IPv6 address assignment. In that they assign one or more /64 blocks to end users [1, 5, 27, 36]. Interestingly enough, ARIN on the other hand recommends end users receive a /48 block [7]. From a rate limiting perspective this could be interesting, as this means that users using ARINs IPv6 space generally have access to larger blocks.

The reasoning behind assigning seemingly large IPv6 blocks to send users is simple. RFC4862 introduces SLAAC, which is a mechanism that allows for address assignment without a DHCP server in IPv6. For this to work properly a /64 block is required. [35]. Assigning multiple /64 blocks allows the end user to partition its network into seperate subnets.

Furthermore, RFC6177 states that they want to make it as effortless as possible for businesses to get a /48 block to avoid businesses having to resort to IPv6-to-IPv6 Network Address Translation once they run out of /64 subnets [44].

6.3.2 Implementation of ISP. To get a better idea on how ISP implement these recommendations we compiled a list from various large ISP in the table which can be found in Appendix A. Due to the language barrier, only English speaking countries we're researched, plus The Netherlands, spanning over 3 different RIRs. Only ISPs for which some kind of source could be found where included.
From the results it becomes clear that ISP operating under a certain region do not necessarily follow their RIRs recommendations, and in general are more generous in the size of the block that gets assigned to end users. While on the other hand, for the US-based ISPs, users are assigned less than is recommended by ARIN.

6.3.3 IPv6 hitlist. Finally to get an even better understanding of how ISPs implement these policies one source of data we can look at are IPv6 hitlists. One of the most complete currently available IPv6 hitlists was chosen [17, 51].

This data of all 6.7 million responsive IPv6 addresses was analysed using a Python script [49]. With as output the number of active addresses under each active /48 block. This was chosen as a /48 block, given some small exceptions, should be the the largest assigned prefix to end users, as stated by RIR and RFC policies discussed in the previous section. Finally this data was clustered by known IPv6 address ranges of each RIR according to IANA [23].

Using data from 2023-05-27, this script produced results which can be found in table 2

RIR	# Active /48's	Avg. active count /48	1k+ active /48's
LACNIC	17131	8.56	12 (0.07%)
RIPE NCC	148959	29.0	784 (0.53%)
ARIN	29937	36.34	125 (0.41%)
APNIC	46774	18.56	69 (0.14%)
AFRINIC	1101	11.16	0

Table 2. /48 block data per RIR region

Given the idea behind IPv6 addresses, where each device on a network has its own, open to the internet public IP address. We can get some idea on the prefix sizes assigned to end users from this table for each different region. For example for RIPE NCCs region, there are on average less IPv6 addresses behind an active /48 block than for ARIN. Which likely means end users get assigned larger address ranges under RIPE NCC than under ARIN. Interestingly enough this conflicts with the ISP policy's stated by the respective RIR regions, but as was also discussed, ISPs do not always tend to follow these recommendations.

To get an even better understanding of this data, we also looked at the distribution of active address counts per /48 block. This way we can get some insight into how many /48 blocks are heavily populated per RIR region. These results can be found in figure 3 and figure 4

One thing that becomes more obvious when looking at figure 4, is that mainly RIPE NCCs and ARINs distributions looks very similar. While for example AFRINIC clearly assigns larger sizes. But this could also be due to the scarcity of internet connectivity in Africa.

To give some more insight on the interesting case of /48 blocks with lots of active addresses, in table 2 we provide the exact counts of the number of /48 blocks with 1000 or more active addresses.

A reasonable assumption to make, is that a /48 block with more than 1000 active address can't possibly be a single end user, and is likely an ISP that smaller assigns parts of a /48 block to end users. To reaffirm this, consider that the average household won't have more

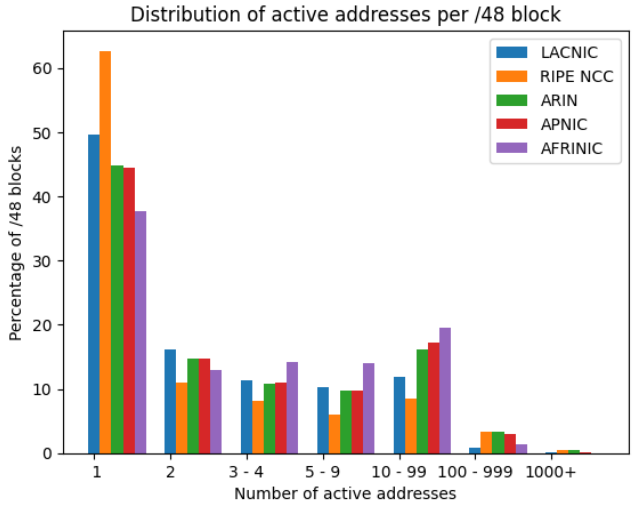

Fig. 3. /48 distribution per RIR

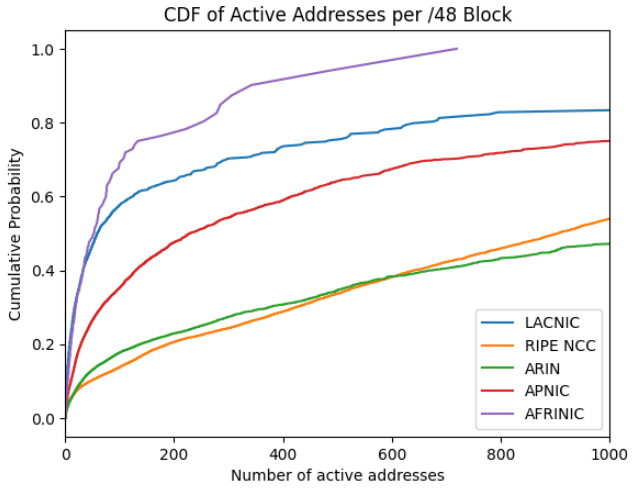

Fig. 4. CDF chart of /48 distribution per RIR (excluding 1k+)

than 1000 devices connected to the internet. With the assumption and the above data, we can make some interesting observations: First of all, in contrary to conclusions made previously in this subsection, and in the "implementation of ISP's." subsection. The table above does, slightly seem to reflect the policies of the RIR regions, most notably when comparing RIPE NCC and ARIN like before. Under ARIN, 0.41% of all /48 blocks have more than 1000 active addresses. While under RIPE, this is 0.53%. Thus, possibly showing that end users under ARIN get larger prefix sizes, but this different is not too large. A strong conclusion can't be made, because not all addresses are from end users, some might be routers or servers operated by organisations.

One thing this data does show, is that rate-limiting by /48 block for IPv6 indeed might cause issues for normal end users. As a single /48 block could easily represent more than a thousand devices.

6.3.4 Tunnel brokers. Like mentioned before, IPv4 to IPv6 tunnel brokers exist. Which can give any attacker who has an internet connection access to a /48 block. These services are generally free to use and require no personal information [15, 16]. Therefore it can be assumed that any knowledgeable attacker can have access to a /48 block.

6.3.5 Attackers with access to AS number(s). For attackers with an ASN under RIPE NCC, one could request a IPv6 address range. To do this, you need to be an RIPE NCC member, which costs 1000 EUR, plus an annual membership fee of around 1500 EUR [40]. However you can then request a /29 IPv6 block without any additional justification [37]. It is also worth nothing that a user can obtain a provider independent (PI) /48 assignment, which does not require a membership, but requires a sponsoring LIR with a RIPE membership [38].
Similarly for ASNs under ARIN, a /32 IPv6 block can be obtained with relative ease and little justification [9]. Costs are a bit lower with ARIN, with a 800 USD sign up fee, and a 1150 USD annual fee [8].
For APNIC, some more justification for address space seems to be required. As their "Proof of Internet number resource needs". To get a similarly sized block (/32) at APNIC they ask for a 500 AUD sign up fee and a 2025 AUD annual membership fee [6].
For LACNIC, with some small justification a /32 could be obtained. At the cost of a 2100 USD initial fee plus that same amount annually [28]. Finally, for AFRINIC they have quite strict policies on proving that you are an internet provider [2]. But they will provide a /32 at the cost of a one time payment of 1650 USD and an annual fee of 2500 USD [3].
Finally, table 3 summarises these findings. With these findings, we

RIR	Justification	Prefix	Cost in USD
LACNIC	Little	/32	$2100 and $2100 annually
RIPE NCC	None	/29	$1091 and $1637 annually
ARIN	Little	/32	$800 and $1150 annually
APNIC	Strict	/32	$342 and $1386 annually
AFRINIC	Strict	/32	$1650 and $2500 annually

Table 3. Cost for an ASN + prefix

can conclude that a motivated, well funded, attacker residing in LACNIC, RIPE NCC or ARINs regions can relatively easily get access to an entire /32 IPv6 block.
As for nation-state attackers, as they could have access to multiple ASNs from multiple ISPs as demonstrated in a nation-state DDoS attack on Google in 2017 [13]. Therefore access to multiple /32 IPv6 blocks can be assumed.

6.4 RQ4

Combining the information gathered from the previous sub-research questions, we can give some advise on how IPv6 rate limiting should be approached.

First of all, rate should at least be done per /64 block, this is how arguably the industry leader, Cloudflare implements it. This is reasonable, as per RFC spec a minimum of a /64 block should be assigned per end user [44]. Making this limit higher, like /48 might cause some issues for real users, as in RQ3 was found that many of the same /48 blocks are used hundreds if not thousands of end users. On top of that, while per-/48 blocking is too strict, it is also useless against an attacker who is motivated enough to acquire a ASN number. Which gives them access to a /32 or more in many RIR regions. Effectively giving them 2^{16} usable addresses in per-/48 rate limiting.

Given difficulty to obtain a new IPv6 prefix to use, whether that is through an ISP or through a RIR via an ASN. We think the best approach to prevent abuse is to monitor for either /56, /48 or /32 blocks that suddenly have a large spike in requests, and then permanently block that range. Thus requiring the attacker to obtain a new IPv6 address range. System administrators can decide the whitelist of block the range based on based on previous traffic data, associated ASN, request rate and types of requests to determine if this is an attack or benign user traffic/growth.
This should stop most attackers who does not have access to a botnet, that, combined with blocking per-/64 seems to be most reasonable. This approach does have the flaw of if an attacker knows about this implementation, he can just simply scale up his DDoS attack slowly, thus not showing any suspicion. A pseudo code implementation of this idea can be found in Listing 3, which modifies the standard leaky-bucket design of implementing rate limiting [48] slightly
Furthermore the reasoning behind the partially manual approach is the infrequency of IPv6 based attacks being performed at this point in time.

Listing 3. Pseudo code IPv6 rate limiting implementation

```
def on_ipv6_request(request):
    # take the first 64 bits
    small_prefix = request.ip.bits()[:64]
    # take the first 48 bits
    large_prefix = request.ip.bits()[:48]
    if bucket_is_full(small_prefix):
        block_request()

    if bucket_is_full(large_prefix) and not
        whitelisted(large_prefix):
        notify_sysadmin()
        pass_request()

    pass_request()
```

7 CONCLUSION

In conclusion, this paper has highlighted the poor state of IPv6 rate limiting support across the entire industry. Both in the enterprise and open source solutions. This can be seen by the fact that, with very few exceptions, per-IP IPv6 rate limiting is enabled by default. Which as shown, is completely useless against almost all users who have an IPv6-enabled internet connection.
A watertight, automated solution is difficult imagine. However it is important that the internet community at least becomes aware of this issue.
Finally, for future work. More work can be done in researching the effect of memory usage of a large-scale IPv6 DDoS attack on more software implementations could be performed. Which in turn might find some insights for a more proper solution for IPv6 based rate limiting, one which would require less manual work.

REFERENCES

[1] AFRINIC. 2013. IPv6 Address Allocation and Assignment Policy | AFPUB-2013-v6-001. https://afrinic.net/ipv6-address-allocation-and-assignment-policy-afpub-2013-v6-001
[2] AFRINIC. 2023. How to become an AFRINIC Resource Member. https://afrinic.net/become-member#eligibility
[3] AFRINIC. 2023. Membership Fee and Payment Facilities. https://afrinic.net/membership/cost#calculator
[4] Alibaba. 2022. Configuring rate limiting. https://www.alibabacloud.com/help/en/alibaba-cloud-cdn/latest/configure-rate-limiting
[5] APNIC. 2013. APNIC guidelines for IPv6 allocation and assignment requests. https://www.lacnic.net/684/2/lacnic/4-ipv6-address-allocation-and-assignment-policies
[6] APNIC. 2023. Get IP: Before You Begin. https://www.apnic.net/get-ip/
[7] ARIN. 2004. IPv6 Address Allocation and Assignment Policy. https://www.arin.net/vault/policy/archive/ipv6_policy.html
[8] ARIN. 2023. Fee Schedule. https://www.arin.net/resources/fees/fee_schedule/#internet-service-providers-isps
[9] ARIN. 2023. Requesting IP Addresses or ASNs. https://www.arin.net/resources/guide/request/
[10] AWS. 2023. Rate-based rule statement. https://docs.aws.amazon.com/waf/latest/developerguide/waf-rule-statement-type-rate-based.html
[11] B. Blechschmidt. 2023. Freebind. https://github.com/blechschmidt/freebind.
[12] caddy documentation. 2023. Module http.handlers.rate_limit. https://caddyserver.com/docs/modules/http.handlers.rate_limit
[13] Catalin Cimpanu. 2020. Google says it mitigated a 2.54 Tbps DDoS attack in 2017, largest known to date. https://www.zdnet.com/article/google-says-it-mitigated-a-2-54-tbps-ddos-attack-in-2017-largest-known-to-date/
[14] Google Cloud. 2023. Rate limiting overview. https://cloud.google.com/armor/docs/rate-limiting-overview
[15] Hurricane Electric. 2023. Hurricane Electric Free IPv6 Tunnel Broker. https://www.tunnelbroker.net/
[16] Dr. Paolo Fasano, Dr. Ivano Guardini, Alain Durand, and Domenico Lento. 2001. IPv6 Tunnel Broker. RFC 3053. https://doi.org/10.17487/RFC3053
[17] Oliver Gasser, Quirin Scheitle, Pawel Foremski, Qasim Lone, Maciej Korczynski, Stephen D. Strowes, Luuk Hendriks, and Georg Carle. 2018. Clusters in the Expanse: Understanding and Unbiasing IPv6 Hitlists. In *Proceedings of the 2018 Internet Measurement Conference* (Boston, MA, USA). ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3278532.3278564
[18] Github. 2023. web-server. https://github.com/topics/web-server
[19] Google. 2023. IPv6 Adoption. https://www.google.com/intl/en/ipv6/statistics.html#tab=ipv6-adoption
[20] Brian Haberman and Bob Hinden. 2005. Unique Local IPv6 Unicast Addresses. RFC 4193. https://doi.org/10.17487/RFC4193
[21] HAProxy. 2023. They use it ! https://www.haproxy.org/they-use-it.html
[22] Russ Housley, John Curran, Geoff Huston, and David R. Conrad. 2013. The Internet Numbers Registry System. RFC 7020. https://doi.org/10.17487/RFC7020
[23] IANA. 2019. IPv6 Global Unicast Address Assignments. https://www.iana.org/assignments/ipv6-unicast-address-assignments/ipv6-unicast-address-assignments.xhtml
[24] IBM. 2020. Configuring rate limiting. https://cloud.ibm.com/docs/cis?topic=cis-cis-rate-limiting
[25] Cloudflare Inc. 2023. Configuring Cloudflare Rate Limiting (previous version). https://developers.cloudflare.com/support/firewall/tools/configuring-cloudflare-rate-limiting/
[26] kinsta. 2023. kinsta. https://kinsta.com/cloudflare-market-share/
[27] LACNIC. 2023. IPv6 ADDRESS ALLOCATION AND ASSIGNMENT POLICIES. https://www.lacnic.net/684/2/lacnic/4-ipv6-address-allocation-and-assignment-policies
[28] LACNIC. 2023. ISP IPv6 Fees. https://www.lacnic.net/5450/2/lacnic/isp-ipv6-fees
[29] ldrg. 2022. Reply: How to protect a web application from IPv6 bots? https://serverfault.com/a/1102191

[30] Frank Li and David Freeman. 2020. Towards A User-Level Understanding of IPv6 Behavior. In *Proceedings of the ACM Internet Measurement Conference* (Virtual Event, USA) *(IMC '20)*. Association for Computing Machinery, New York, NY, USA, 428–442. https://doi.org/10.1145/3419394.3423618

[31] Cloudflare Marek Majkowski. 2018. The real cause of large DDoS - IP Spoofing. https://blog.cloudflare.com/the-root-cause-of-large-ddos-ip-spoofing/

[32] Theodor Schnitzler Maximilian Golla and Markus Dürmuth. 2018. "Will Any Password Do?" Exploring Rate-Limiting on the Web. https://wayworkshop.org/2018/papers/way2018-golla.pdf

[33] Microsoft. 2022. Deny by Request Rate <denyByRequestRate>. https://learn.microsoft.com/en-us/iis/configuration/system.webserver/security/dynamicipsecurity/denybyrequestrate

[34] Microsoft. 2023. Configure a Web Application Firewall rate limit rule. https://learn.microsoft.com/en-us/azure/web-application-firewall/afds/waf-front-door-rate-limit-configure?pivots=portal

[35] Dr. Thomas Narten, Tatsuya Jinmei, and Dr. Susan Thomson. 2007. IPv6 Stateless Address Autoconfiguration. RFC 4862. https://doi.org/10.17487/RFC4862

[36] RIPE NCC. 2020. IPv6 Address Allocation and Assignment Policy. https://www.ripe.net/publications/docs/ripe-738#assignment

[37] RIPE NCC. 2021. How to Request an IPv6 Allocation. https://www.ripe.net/manage-ips-and-asns/ipv6/request-ipv6/how-to-request-an-ipv6-allocation

[38] RIPE NCC. 2021. How to Request an IPv6 PI Assignment. https://www.ripe.net/manage-ips-and-asns/ipv6/request-ipv6/how-to-request-an-ipv6-pi-assignment

[39] RIPE NCC. 2023. Autonomous System Numbers. https://www.ripe.net/manage-ips-and-asns/as-numbers/request-an-as-number

[40] RIPE NCC. 2023. Billing, Payment and Fees. https://www.ripe.net/participate/member-support/payment

[41] nginx. 2023. Module ngx_http_limit_req_module. https://nginx.org/en/docs/http/ngx_http_limit_req_module.html#limit_req_zone

[42] Adam Pritchard. 2022. The scary state of IPv6 rate-limiting. https://adam-p.ca/blog/2022/02/ipv6-rate-limiting/

[43] Nick Ramirez. 2019. HAProxy Rate Limiting: Four Examples. https://www.haproxy.com/blog/four-examples-of-haproxy-rate-limiting

[44] Rosalea Roberts, Geoff Huston, and Dr. Thomas Narten. 2011. IPv6 Address Assignment to End Sites. RFC 6177. https://doi.org/10.17487/RFC6177

[45] sysstat. 2023. sysstat - System performance tools for the Linux operating system. https://github.com/sysstat/sysstat.

[46] Traefik. 2023. Success stories. https://traefik.io/success-stories/

[47] Traefiklabs. 2023. RateLimit. https://doc.traefik.io/traefik/middlewares/http/ratelimit/

[48] J. Turner. 1986. New directions in communications (or which way to the information age?). *IEEE Communications Magazine* 24, 10 (1986), 8–15. https://doi.org/10.1109/MCOM.1986.1092946

[49] Pieter van Heijningen. 2023. Script to generate useful data from IPv6 hitlists. https://gitlab.utwente.nl/s2614359/ipv6-ratelimiting

[50] W3techs. 2023. Usage statistics of web servers. https://w3techs.com/technologies/overview/web_server

[51] Johannes Zirngibl, Lion Steger, Patrick Sattler, Oliver Gasser, and Georg Carle. 2022. Rusty Clusters? Dusting an IPv6 Research Foundation. In *Proceedings of the 2022 Internet Measurement Conference* (Nice, France). ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3517745.3561440

APPENDIX A

Table 4. ISP IPv6 prefix size assignments to end users

ISP	Country	Prefix size	Source
RIPE NCC			
KPN	NL	/48	https://id.nl/huis-en-entertainment/computer-en-gaming/netwerk/ip-adressen-raken-op-alles-over-ipv6
Ziggo	NL	/56	https://id.nl/huis-en-entertainment/computer-en-gaming/netwerk/ip-adressen-raken-op-alles-over-ipv6
Freedom Internet	NL	/48	https://helpdesk.freedom.nl/algemene-instellingen-eigen-modem
T-Mobile Netherlands	NL	no support	https://community.t-mobile.nl/bekabeld-internet-492/een-provider-zonder-ipv6-hoe-is-dat-350182
Sky	UK	/56	https://www.ispreview.co.uk/index.php/2016/09/uk-isp-sky-broadband-officially-finish-roll-ipv6.html
BT Broadband	UK	/56	https://www.ispreview.co.uk/index.php/2016/11/bt-broadband-lines-now-support-ipv6-internet-addresses.html
Virgin Media	UK	no support	https://www.ispreview.co.uk/index.php/2021/11/update-on-ipv6-plans-for-virgin-media-talktalk-plusnet-and-vodafone.html
Vodafone UK	UK	no support	https://www.ispreview.co.uk/index.php/2021/11/update-on-ipv6-plans-for-virgin-media-talktalk-plusnet-and-vodafone.html
Plusnet	UK	no support	https://www.ispreview.co.uk/index.php/2021/11/update-on-ipv6-plans-for-virgin-media-talktalk-plusnet-and-vodafone.html
TalkTalk	UK	no support	https://www.ispreview.co.uk/index.php/2021/11/update-on-ipv6-plans-for-virgin-media-talktalk-plusnet-and-vodafone.html
ARIN			
Comcast Xfinity	US	/64	https://forums.xfinity.com/conversations/your-home-network/ipv6-prefix-delegation-size/602daf7cc5375f08cd0d948c
Verizon	US	/56	https://www.reddit.com/r/Fios/comments/xoq9s6/hi_sorry_to_bother_but_what_prefix_delegation/
Google Fiber	US	/64	https://support.google.com/fiber/thread/808240/google-fiber-is-now-handing-out-a-64-for-ipv6-instead-of-an-56-ipv6-prefix-delegation-like-earlier
APNIC			
Internode	AUS	/56	https://www.internode.on.net/support/guides/internet_access/ipv6/faq/
Aussie Broadband	AUS	/48	https://www.aussiebroadband.com.au/help-centre/internet/does-aussie-broadband-support-ipv6/