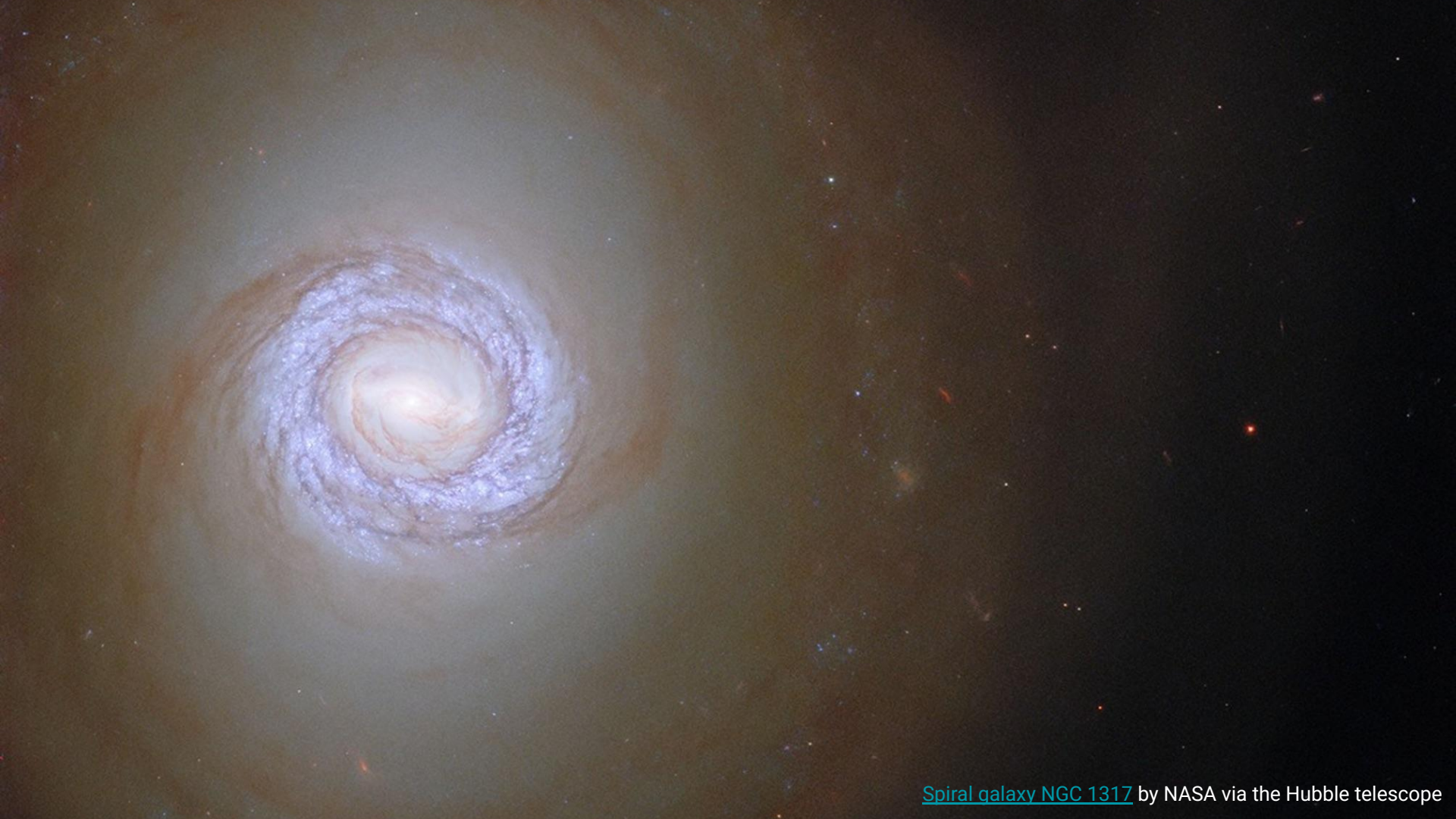
ElixirConf® US

Tyler A. Young

Cat and Mouse: Challenges in Adversarial Web Scraping

@tylerayoung.com @tylerayoung@fosstodon.org



Spiral galaxy NGC 1317 by NASA via the Hubble telescope

```
$ curl "http://tylerayoung.com"
```

```elixir
Req.get!("https://tylerayoung.com")
```

403 Forbidden

nginx/1.28.0

bidden

```elixir
Req.new(
  url: "https://tylerayoung.com",
  headers: %{
    "User-Agent" => [
      "Mozilla/5.0 […] Safari/605.1.15"
    ]
  }
)
|> Req.get!()
```

(We'll come back to the ethics of this!)

```
Req
    ur                                    om",
    he

                                          605.1.15"

    }
)
|> Req                                    e back to the ethics of this!)
```

403 Forbidden

nginx/1.28.0

```
headers = %{
  "User-Agent" => ["Mozilla/5.0 […]"],
  "Accept" => ["text/html,application/xml,[…]"],
  "Accept-Encoding" => ["gzip, deflate"],
  "Connection" => ["keep-alive"],
  "sec-fetch-dest" => ["document"],
  "sec-fetch-mode" => ["navigate"],
  "sec-fetch-side" => ["none"],
  …
}
```

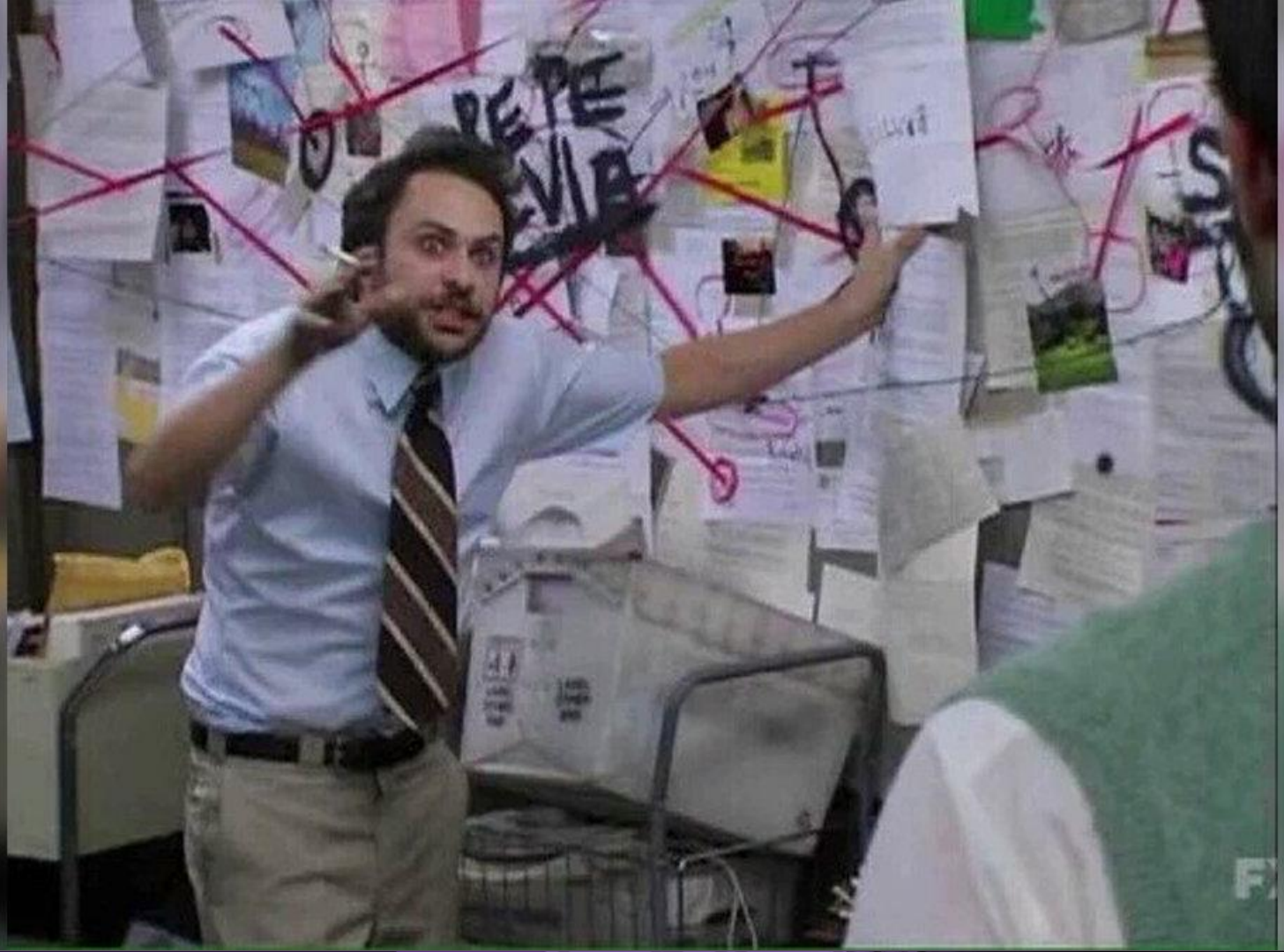
(We'll come back to the ethics of this!)

403 Forbidden

nginx/1.28.0

403 Forbi

nginx/1.28.0



ElixirConf US

Who am I?

- MMO flight sim server for X-Plane in 2019
- Real-time document collab for Felt until 2024
- Took a break to work on indie web site monitoring tool, SleepEasy
- Currently at Jump (AI for financial advisors)
  - We're always hiring!
  - It's a wonderful place to work!

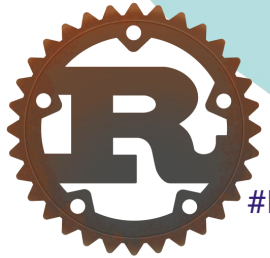
@tylerayoung.com @tylerayoung@fosstodon.org

JA4+ fingerprinting

github.com/FoxIO-LLC/ja4

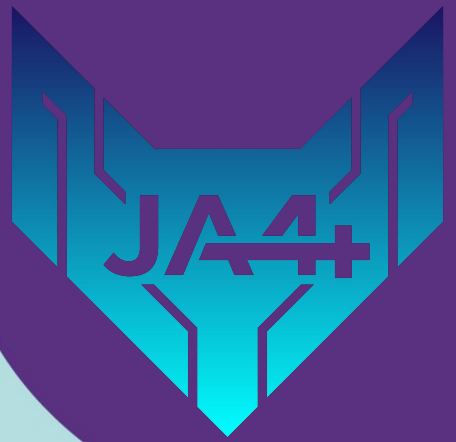
- Looks at the way the client performs a TLS handshake
  - Ciphers, extensions, timings, etc.
- Block the traffic if it doesn't match a known consumer browser





ElixirConf US

JA4+ fingerprinting



HAPROXY

NGINX

Vercel

CLOUDFLARE®

fastly

aws

#ElixirConf

ElixirConf US

#ElixirConf

Just use a headless browser!

wallaby

Playwright

Selenium

Just use a headless browser!

Pros:

⭐ Exactly imitates the browser!

Cons:

🐢 Slow

💥 Unreliable

🐷 Memory hog

🐷 CPU hog

Just use a headless browser!

Pros:

⭐ Exactly imitates th

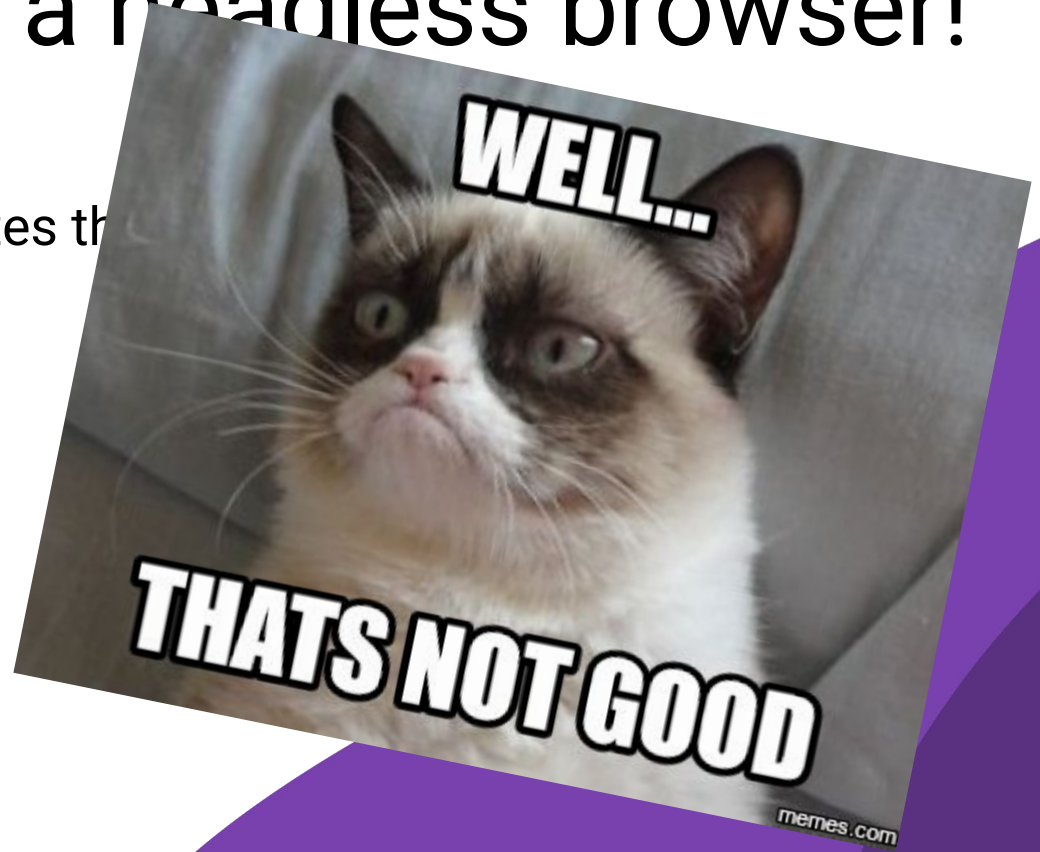
Cons:

🐢 Slow

💥 Unreliable

🐖 Memory hog

🐖 CPU hog



Just pay to make it somebody else's problem!

- Zyte
- Apify
- ScrapingBee
- So, so many others

4¢ to $1+/page

ElixirConf US

What would a better solution look like?

#ElixirConf

Enter curl-impersonate

github.com/lwthiker/curl-impersonate

Exactly* emulates the way Chrome, Safari, Edge, and Firefox perform a request:

- Protocol (HTTP/2)
- TLS handshake
- Headers
- More?

*Results not guaranteed.

Using curl-impersonate from Elixir

- NIF

Using curl-impersonate from Elixir

- NIF
  - How confident are you that it won’t crash the whole system?

Using curl-imp rsonate from Elixir

- NIF
  - How confident
    crash the who

Using curl-impersonate from Elixir

- NIF
  - How confident are you that it won't crash the whole system?
- `System.shell/{1,2}`
  - stderr mixed in with stdout
  - Parse your own results

```
$ curl-impersonate -v https://tylerayoung.com
* Host tylerayoung.com:443 was resolved.
* IPv6: (none)
* IPv4: 18.208.88.157, 98.84.224.111
. . .
* [HTTP/2] [1] [user-agent: curl/8.7.1]
* [HTTP/2] [1] [accept: */*]
> GET / HTTP/2
> Host: tylerayoung.com
> User-Agent: curl/8.7.1
> Accept: */*
>
* Request completely sent off
< HTTP/2 200
< accept-ranges: bytes
< age: 0
< cache-control: public,max-age=0,must-revalidate
. . .
< content-length: 70454
<
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
```

Using curl-impersonate from Elixir

- NIF
  - How confident are you that it won't crash the whole system?
- `System.shell/{1,2}`
  - stderr mixed in with stdout
  - Parse your own results
- Library?

BrowseyHttp

github.com/s3cur3/browsey_http

- Wrapper for curl-impersonate command line
- Loads other page resources (images, JS, CSS) in parallel like a browser
- Preserves cookies across requests
- `:max_response_size_bytes`

~~Defense~~ Offense in depth

1. Try Browsey first when scraping a site
2. If it fails, try a headless browser
3. Only if that fails, fall back to a third-party provider

What does Browsey *not* solve?

~~What does Browsey *not* solve?~~

How I would defeat me

THE GREAT FLOOD

Open source devs say AI crawlers dominate traffic, forcing blocks on entire countries

AI bots hungry for data are taking down FOSS sites by accident, but humans are fighting back.

BENJ EDWARDS – MAR 25, 2025 4:36 PM | 154

Credit: Henrik Sorensen via Getty Images

Software developer Xe Iaso reached a breaking point earlier this year when aggressive AI crawler traffic from Amazon overwhelmed their Git repository service, repeatedly causing instability and downtime. Despite configuring standard defensive measures—adjusting robots.txt, blocking known crawler user-agents, and filtering suspicious traffic—Iaso found that AI crawlers continued evading all attempts to stop them, spoofing user-agents and cycling through residential IP addresses as proxies.

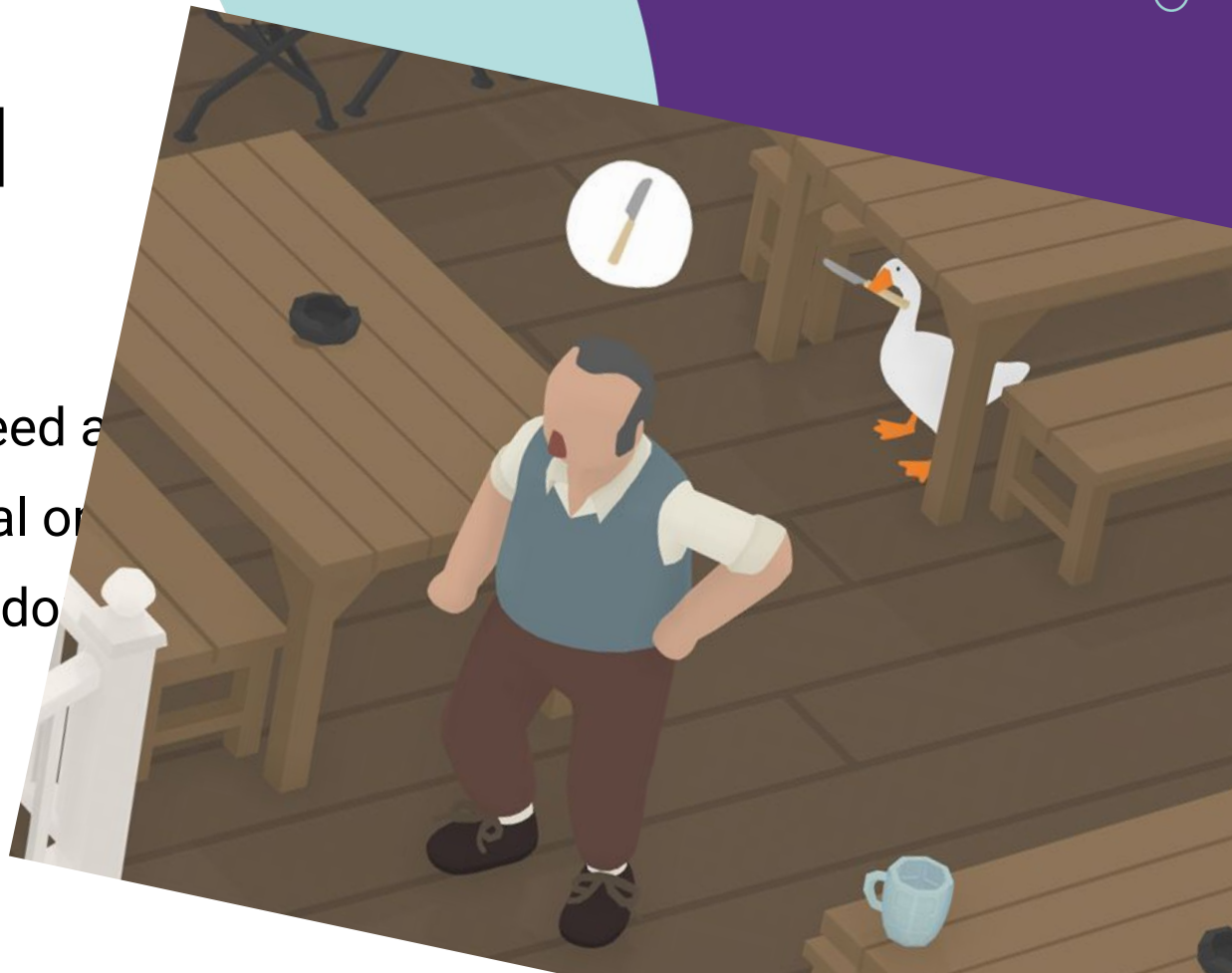
Desperate for a solution, Iaso eventually resorted to moving their server behind a VPN and creating "Anubis," a custom-built proof-of-work challenge system that forces web

How I would defeat me

- Per-IP rate limits (need an IP pool)
- IP source (residential or a data center?)
- “Human factor” (randomness?)
- Require JavaScript
- Authentication

How I would defeat me

- Per-IP rate limits (need a
- IP source (residential or
- “Human factor” (rando
- Require JavaScript
- Authentication



Per-IP rate limiting

- Paraxial.io for Elixir (👋 Michael)
- Cloudflare, AWS WAF, Google Cloud Armor, etc.
- Roll your own via PlugAttack
  github.com/michalmuskala/plug_attack

rate_limit_plug.ex U

lib > prd_web > plugs > rate_limit_plug.ex > ...

```elixir
defmodule MyAppWeb.RateLimit do
  use PlugAttack

  @bypass_rate_limiter? Mix.env() in [:dev, :test]

  rule "allow localhost", conn do
    if @bypass_rate_limiter? do
      allow(true)
    end
  end

  rule "throttle requests by ip", conn do
    throttle({:ip, conn.remote_ip},
      period: to_timeout(second: 60),
      limit: 10,
      storage: {PlugAttack.Storage.Ets, MyAppWeb.RateLimit.Storage}
    )
  end

  @impl PlugAttack
  def block_action(conn, data, _opts) do
    conn
    |> Plug.Conn.send_resp(429, "Too Many Requests\n")
    |> Plug.Conn.halt()
  end
end
```

router.ex M

lib > prd_web > router.ex > { } PrdWeb.Router

You, 1 second ago | 2 authors (You and one other)

```elixir
defmodule PrdWeb.Router do
  use PrdWeb, :router
  use ErrorTracker.Integrations.Plug

  pipeline :browser do
    plug :accepts, ["html"]
    plug MyAppWeb.RateLimit
    plug :fetch_session
    plug :fetch_live_flash
    plug :put_root_layout, {PrdWeb.Layouts, :
    plug :protect_from_forgery

    plug :put_secure_browser_headers, %{
      "content-security-policy" =>
        ContentSecurityPolicy.serialize(
          struct(ContentSecurityPolicy.Policy
        )
    }

    plug :fetch_current_user
    plug :fetch_impersonator_user
    plug :kick_user_if_suspended_or_deleted
    plug PetalFramework.SetLocalePlug, gettex
  end

  pipeline :public_layout do
```

Per-IP rate limiting

- Paraxial.io
- Roll your own via PlugAttack
  github.com/michalmuskala/plug_attack
- Could also limit by ASN* or city (?)
  - github.com/navinpeiris/geoip
  - github.com/g-andrade/locus

*Autonomous System Number.
~100K in use, one per ISP or large org.

Detecting non-residential IP addresses

- All the major cloud providers publish their IP ranges
- Use PlugAttack to block these requests?
- Classifying Data Center IP Addresses in Phoenix Web Applications with Radix Trees by Paraxial.io

Detecting humanity

- Request patterns
  - Are they loading assets at the same time?
  - Are they caching assets correctly?
  - Are they loading pages directly linked by the previous one?
  - How long does a session last?

Detecting humanity

- Request patterns
  - Are they loading assets at the same time?
  - Are they caching assets correctly?
  - Are they loading pages directly linked by the previous one?
  - How long does a session last?
- Are they firing mouse/touch/ key events?

Requiring JavaScript

- Click/touch/key events
- Challenge/response from initial request to get the real content
- Captchas
- Go fully client-side rendered (😱)

Requiring authentication

🎉 No more scrapers!

😬 No more anonymous traffic!

Don’t reinforce digital inequality

- Screen readers
- Keyboard navigation
- Low-end phones in Pakistan
- Scrappy little companies
- Brazilians

Don’t hurt site owners

- Keep traffic levels to human scale
- Respect intellectual property
- Don’t be evil

Perplexity is using stealth, undeclared crawlers to evade website no-crawl directives

2025-08-04

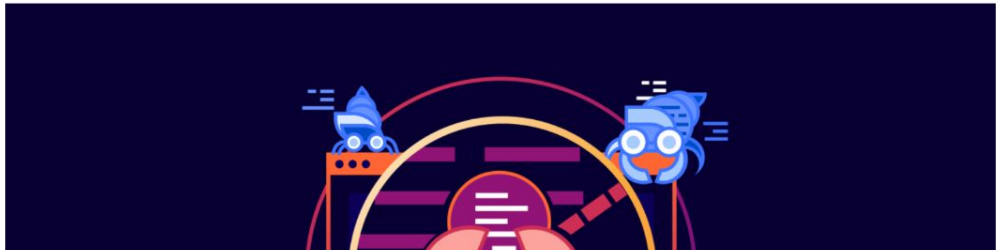
 Gabriel Corral

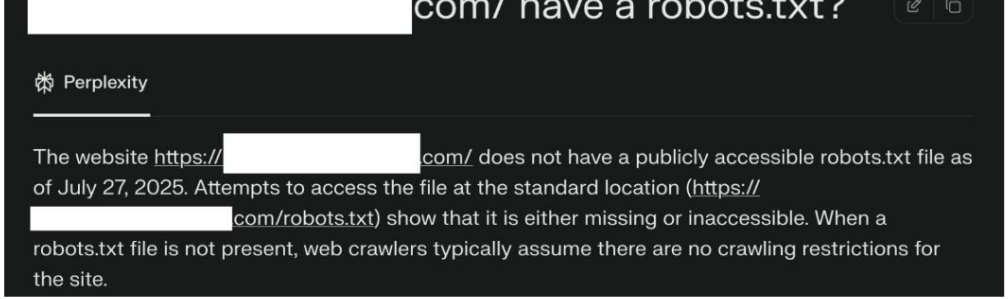
 Vaibhav Singhal

 Brian Mitchell

 Reid Tatoris

5 min read





Obfuscating behavior observed

Bypassing Robots.txt and undisclosed IPs/User Agents

Our multiple test domains explicitly prohibited all automated access by specifying in robots.txt and had specific WAF rules that blocked crawling from Perplexity's public crawlers. We observed that Perplexity uses not only their declared user-agent, but also a generic browser intended to impersonate Google Chrome on macOS when their declared crawler was blocked.

Declared	Mozilla/5.0 AppleWebKit/537.36 (KHTML, like Gecko; compatible; Perplexity-User/1.0; +https://perplexity.ai/perplexity-user)	20-25m daily requests
Stealth	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.0.0 Safari/537.36	3-6m daily requests

Both their declared and undeclared crawlers were attempting to access the content for scraping contrary to the web crawling norms as outlined in RFC 9309.

This undeclared crawler utilized multiple IPs not listed in Perplexity's official IP range, and would rotate through these IPs in response to the restrictive robots.txt policy and

ElixirConf US

Questions?

- Slides: tylerayoung.com
- BlueSky: @tylerayoung.com
- Mastodon: @tylerayoung@fosstodon.org
- Elixir Slack: Tyler Young

BONUS

Dumb tricks to force server-side rendering

- Twitter expects GoogleBot to send auth headers
- Most other bots are expected to execute JavaScript
- Enter: BaiduBot!

```elixir
Req.new(
  url: "http://x.com/TylerAYoung",
  headers: %{
    "User-Agent" => [
      "Mozilla/5.0 (compatible; Baiduspider/2.0; " <>
        "+http://www.baidu.com/search/spider.html)"
    ]
  }
)
|> Req.get!()
```