Vulnerability Report

Vulnerability name: Marco Polo

Presented by:

John Balvin Arias
linkedin twitter

Expert in Web Crawling & Scraping | RPA | Backend Developer(Golang)

Code: https://github.com/johnbalvin/marcopolo

More explanation:

- Fundamentals: https://youtu.be/pynMdmPCNM8
- Using the code: https://youtu.be/-An5LwunxaY

Version 0

Version 0

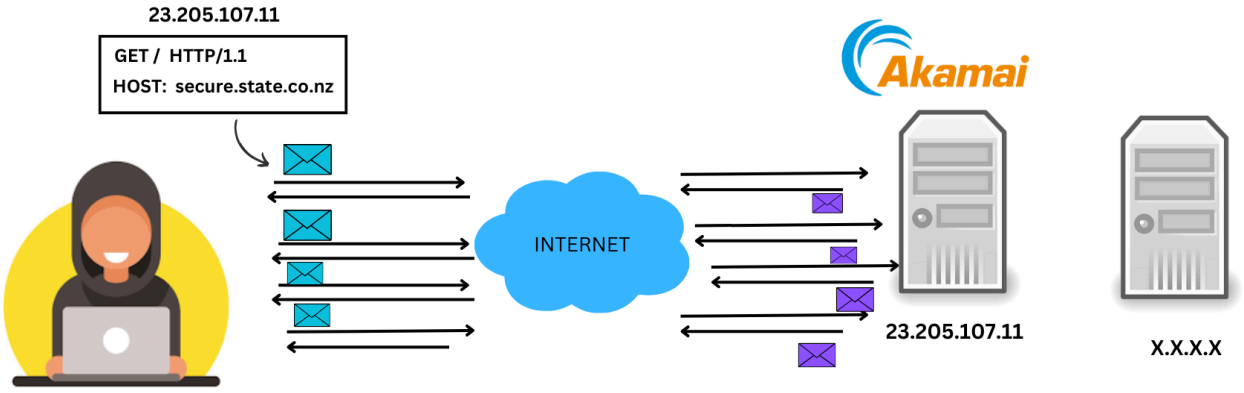
Table of content

The report

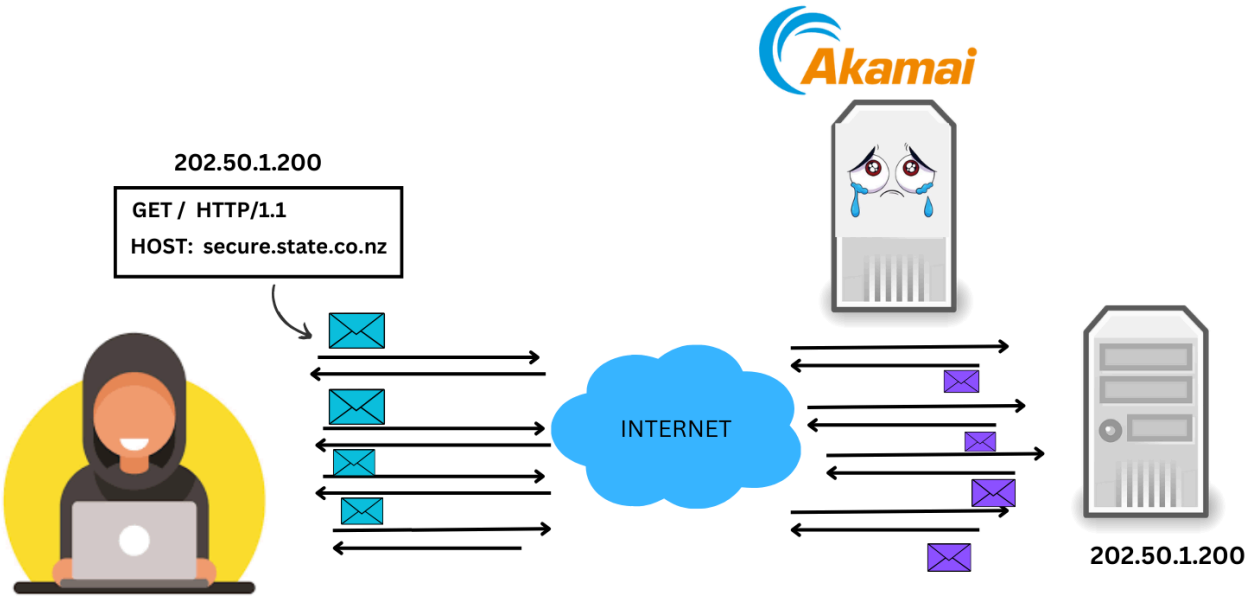
Summary

The vulnerability relies on finding the real IP from the server, this can be used by hackers as the first attack vector.

So this:



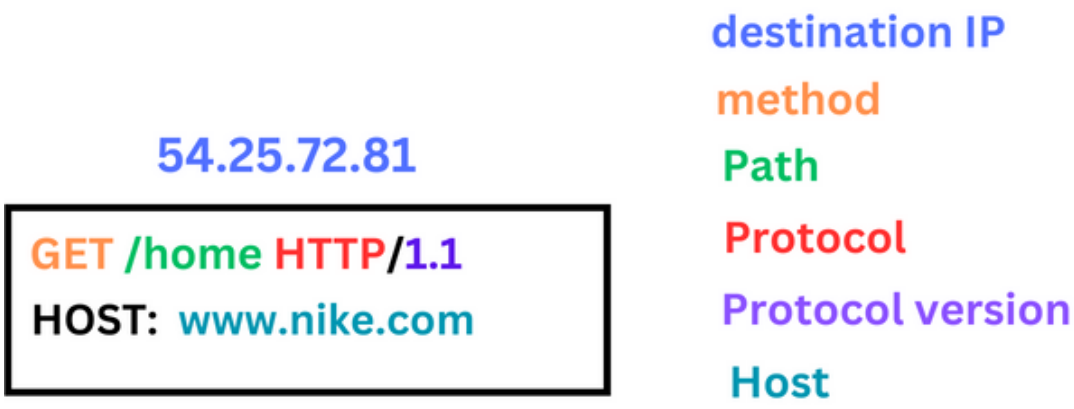
Could be converted to direct attacks just by knowing your server IP:



This is no WAF's fault, direct attacks are possible because the IP is public.

The strategy to get the real IP behind a server is based on the design of the HTTP protocol.

An HTTP package looks like this:



There is more info sent on the package but let's focus on those fields for now.

WHEN MAKING HTTP REQUESTS YOU ARE NOT SENDING IT TO A HOST/DOMAIN, BUT TO AN IP.

What if we try each possible IP until we get a response from the real server

However that takes a lot of time, it could take years, of course you can distribute the job across multiple servers but still take a lot of time, that's why we need some optimizations, some of them are:

- Investigate what ASN provider they use, it could be by checking historical DNS information or it could be by checking unprotected API or unprotected environments, like staging or dev
- Use HTTP first then HTTPs, HTTP for checking if there is any redirection and HTTPs just to confirm
- Blogs talking about what tech they use so we can get the ASN and filter the IPs

It looks simple, because it is actually simple.

Note: for simplicity the "requests" make to the servers on the explanations are HTTP not HTTPS, so if you see something like this:

Don't be scared, I'm not using https for simplicity

The importance of the server IP

There are multiple attacks that can be made to your server, but I'm just gonna focus on the most common one, DDoS attack so you understand why it is so important and you can take precautions on your server.

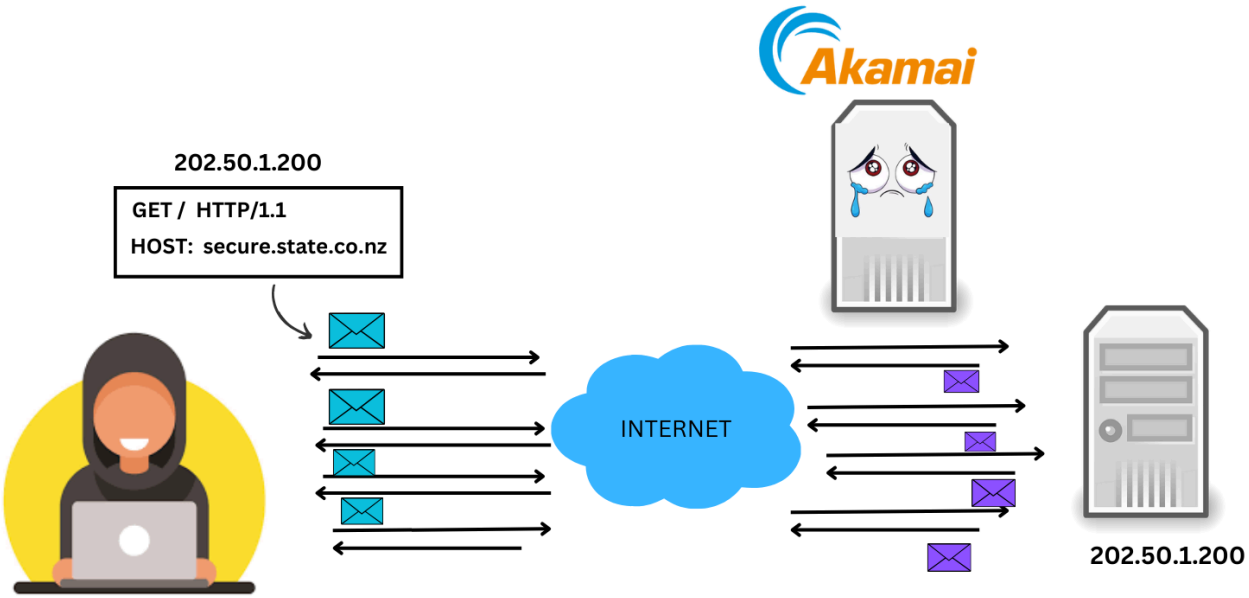
A DDoS attack is super easy to implement, like only 30 lines of code, and it doesn't require a lot of expertise.
Websites use WAF systems like cloudflare, akamai, imperva to prevent this kind of attacks but these websites usually forget their server IP is still accessible to the internet and direct attacks are still possible, someone that knows the server IP can make direct attacks to the server.

This vulnerability is not just for websites using WAF systems, it's actually any website, however using WAF systems gives the illusion you are totally protected and makes you forget your server IP is public and direct attacks are still possible.

Lets see what kind of DDoS attacks a hacker can make.

Noob DDoS

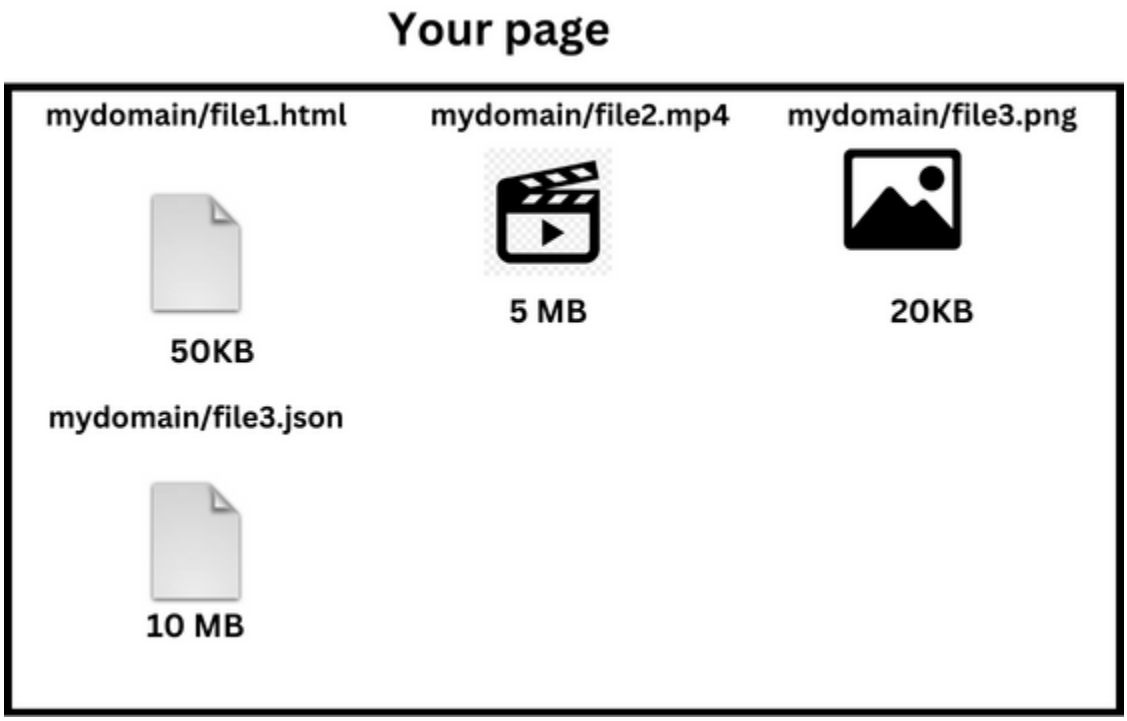
The target is any URL on the website without analyzing anything.
This is the most basic and does almost no harm, because if you have a good cache on the backend, your server will be able to handle this request smoothly.



Network Egress DDoS

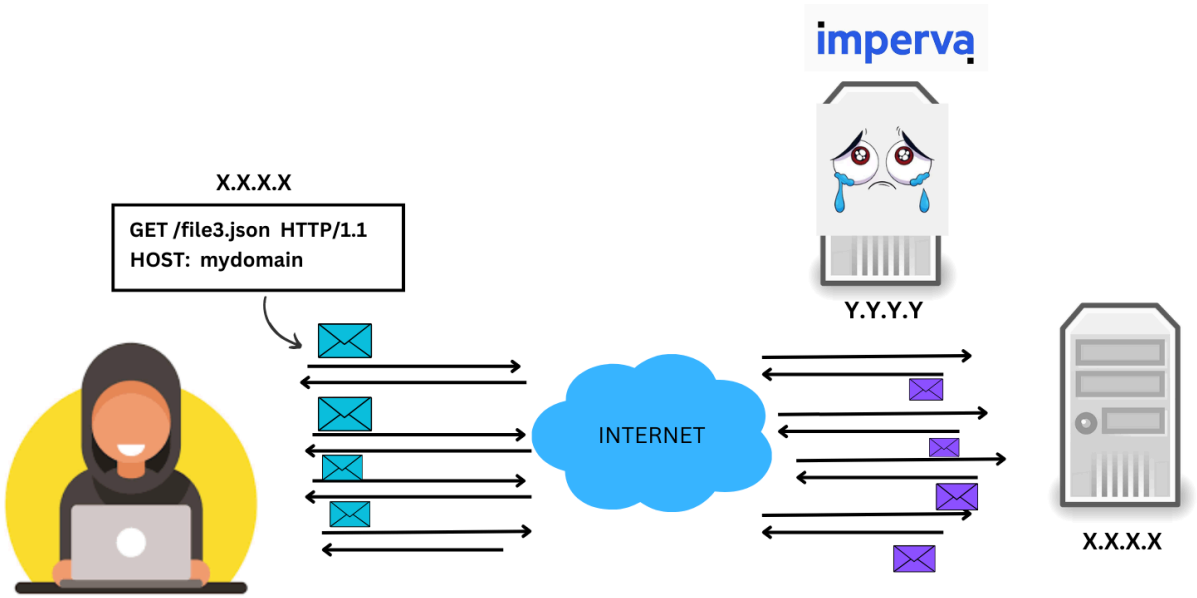
Even worse than a noob DDoS attack is when a hacker makes a DDoS attack to a file with a huge size on your server.

So lets say your page has these files:



A smart hacker will analyze your website and look for the biggest file, in this case:

And will make a DDoS attack to that url:



An attack like that will use a lot of network egress, and hosting providers charge by this network usage.

With the increasing price on the network bandwidth, these are gonna be the common attacks we will see in the upcoming months/years.

Image from twitter, you'll need to verify the price on the hosting website

Cloud Provider	Free allowance	1 TB of egress overage
Cloudflare	--	Free for most services
Heroku	2 TB / mo per app	Not publicly listed
OVHcloud	--	Free and unlimited
Scaleway	--	Free for most services
Hetzner	20-60 TB / mo per instance	$1.09
Linode	1-20 TB / mo per instance	$5.00
Oracle Cloud	10 TB / mo	$8.50
Backblaze	3x the amount of data stored	$10.00
Bunny CDN	--	$10.00
DigitalOcean	100 GB - 10 TB / mo per instance	$10.00
UpCloud	500 GB - 24 TB / mo per instance	$10.86
Vultr	2 TB / mo for most services	$10.00
Fly.io	100 GB / mo	$20.00
Koyeb	100 GB / mo	$40.00
Alibaba Cloud	10 GB / mo	$74.00
Microsoft Azure	100 GB / mo	$78.30
Amazon Web Services	100 GB / mo	$92.16
Railway	--	$100.00
Zeabur	10-100 GB, depends on plan	$100.00
Google Cloud	Depends on service	$111.60
Render	100 GB - 1 TB, depends on plan	$300.00
Vercel	100 GB - 1 TB, depends on plan	$400.00
Netlify	100 GB - 1 TB, depends on plan	$550.00

Let's see why this is a big issue.
Let's say a hacker targets a file with a size of 5MB for a DDoS attack on a page hosted on netlify and let's say for simplicity 1MB=100KB.

Let's also omit the free tier for now. It's not that important because it's low and it won't make such a difference in the result.
In order for a hacker to increase the billing up to $550 for a netlify website, it will take

$$\frac{10^{12}}{5x10^{6}} = 200K$$

It will take about 200K requests, now let's say we have 300 threads running for the DDoS attack and each thread takes 7 Seconds to download the file, it will take:

$$\frac{7\ x\ 200\ x\ 10^{3}}{300} = 4667\ seconds = 78\ minutes$$

Only 78 minutes to increase the bill up to $550.
And this attack is super easy to implement, you need to have a decent bandwidth and a home pc with low resources, imagine a larger file, it will take less requests.
Your server won't crash with these requests, but the billing will hurt more.

And you won't even noticed it until the end of the month when you see your billing like this guy: [Netlify just sent me a $104K bill for a simple static site]

$10^{12} = 1\ T$

$5x10^{6} = 5\ M$

$200\ x\ 10^{3} = 200K$

Database DDoS

For this attack, the hacker will target the database, hacker must know basic sql and database optimizations like indexing.
Sql databases can have huge optimizations by indexing or having a cache on top of it, so for this attack, hackers will try to avoid these two.
So let's say a website have a search option, you can search by name, and you can add filters like id and color:

Id = []
Search value:[]
Color: [yellow,red,blue]

And for making this search the website sends a GET request like this:

```
/search?id=a&value=abc&color=blue
```

So the hacker will just avoid any indexing, and will make the DDoS attack to a broad search requests like this:

```
/search?value=abc
```

And to avoid any cache on top, hacker will make different requests values, so:
One request might look like this:

```
/search?value=a
```

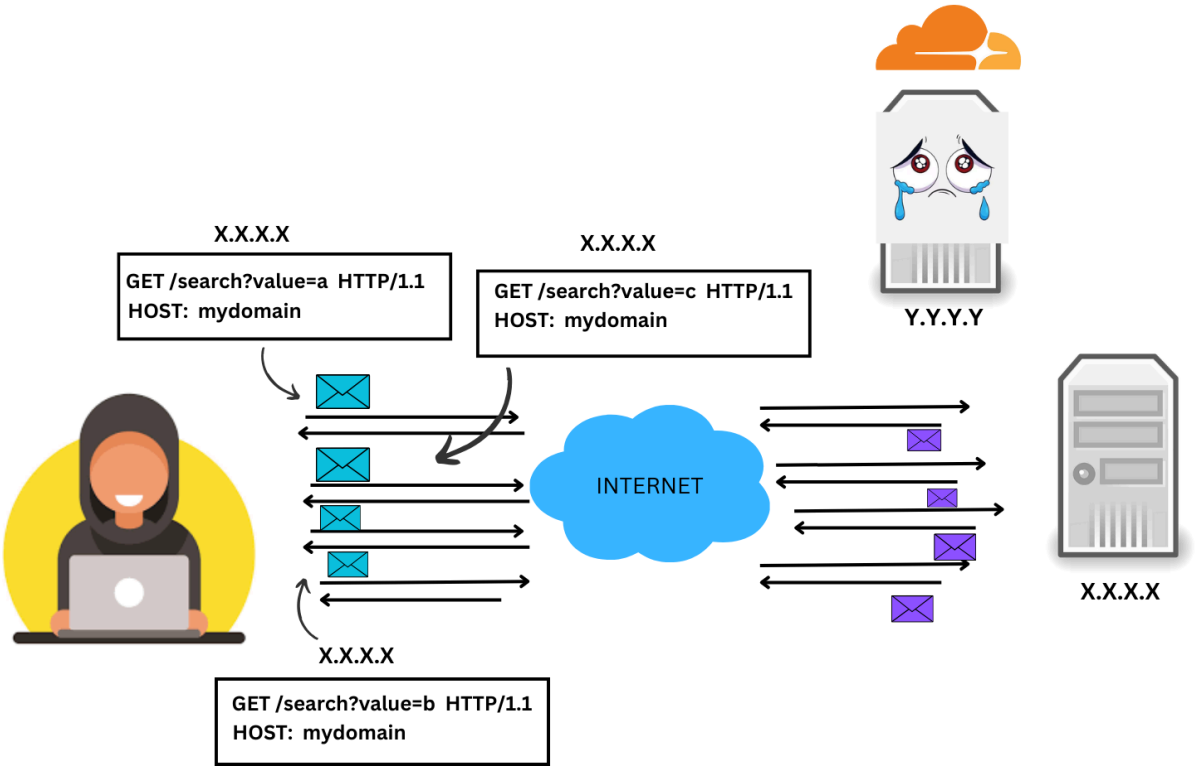
Another request will look like this:

```
/search?value=b
```

And so on:

```
/search?value=c
/search?value=d
/search?value=e
/search?value=e
/search?value=f
/search?value=g
```

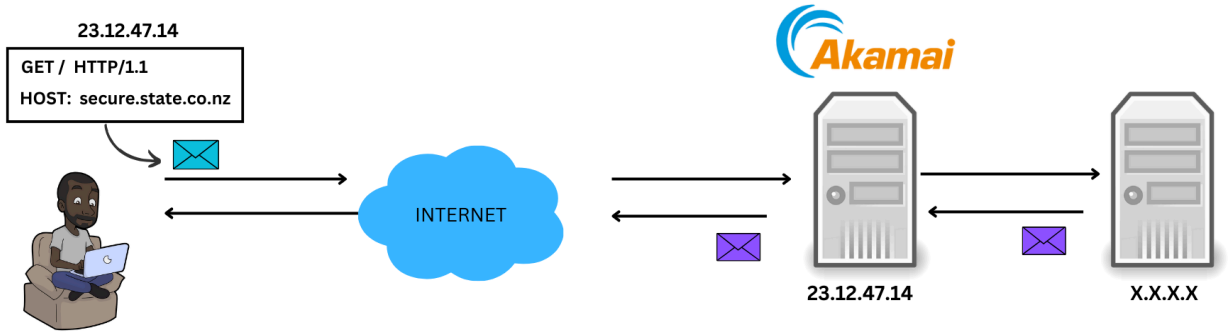
What will eventually happen is the database will stress out a lot and if the website is using a monolithic design, it will crash, or if it’s using a serverless database, the bill will increase a lot.


X.X.X.X
GET /search?value=a HTTP/1.1
HOST: mydomain
X.X.X.X
GET /search?value=c HTTP/1.1
HOST: mydomain
Y.Y.Y.Y
INTERNET
X.X.X.X
X.X.X.X
GET /search?value=b HTTP/1.1
HOST: mydomain

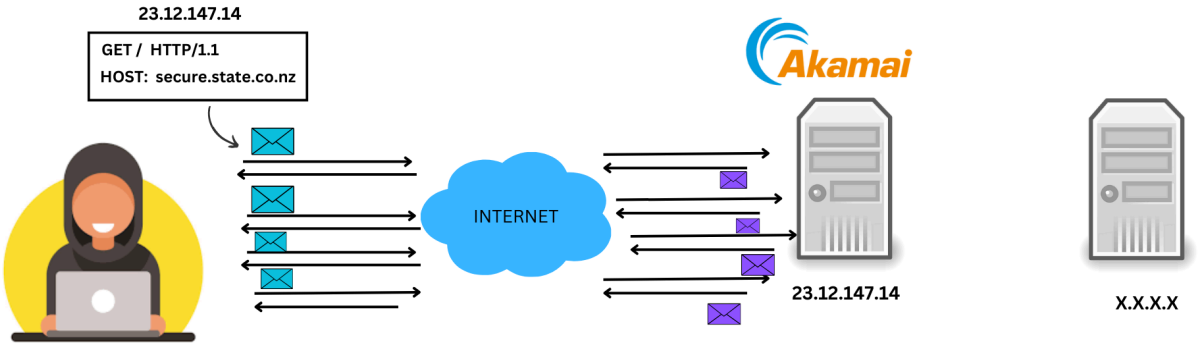
WAF(web application firewall)

I would like to emphasize again that the IP is the most important part for the whole explanation, if you don't get this part, you won't be able to understand how to exploit this vulnerability and how to bypass the WAF systems

Ok so with this in mind let's introduce WAF, so WAF system basically it's a server that sits behind your server and the internet, analyzes all requests and if it doesn't find anything suspicious, it will just continue the request to your server..



If it finds the request is suspicious, then the request won't hit your server and the WAF server will handle it all by itself.



Only if there was a way to bypass these WAF systems once for all.

Meet the vulnerability which I call: Marco Polo

I already mentioned before that this bug relies on the IP, this is because requests are just an IP attach with information, like this:

So what if we try all possible IPs until the real server responds back?

So we do something like this:

etc..

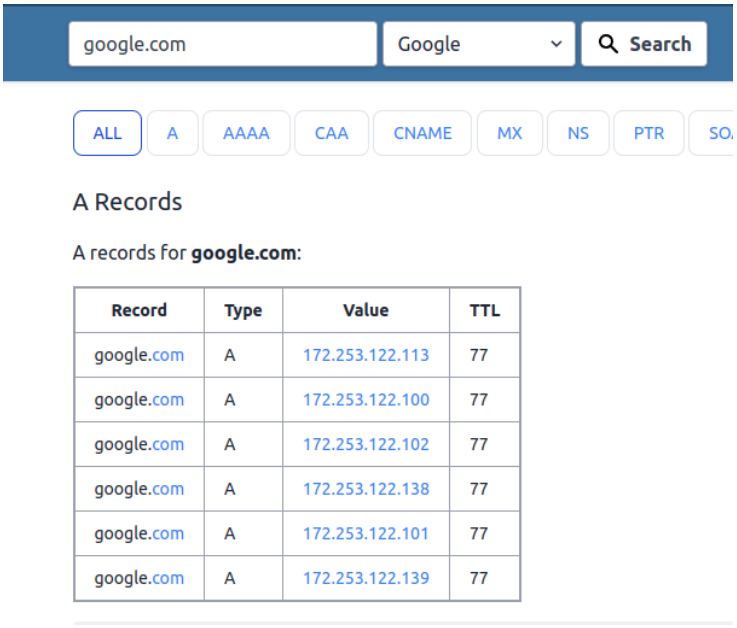
We try all possible IPs until the response looks similar as a response from the real server

So that's basically the vulnerability, but if you try to do it, it will take years to get the server IP, so that's why we need to narrow down the IPs to analyze, for this we need to understand some web fundamentals, DNS and ASN

DNS

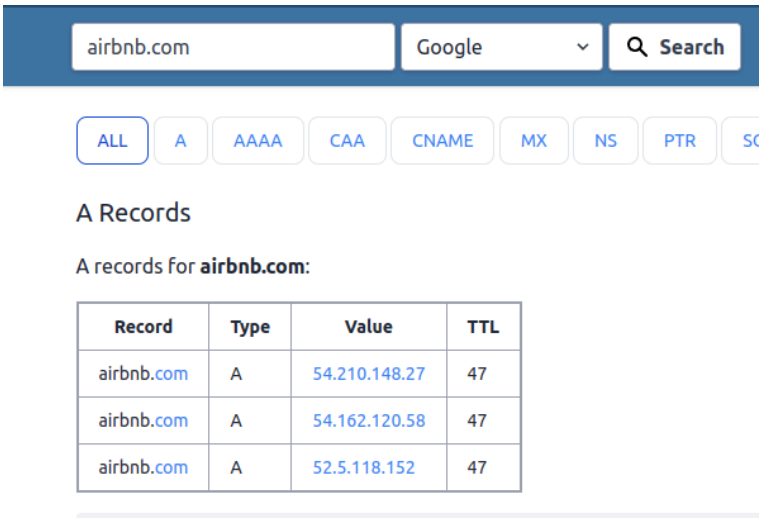
DNS servers contains mapping between domain and IPs
So if type google.com, if you don't have this mapping already saved on your pc, your pc will connect to this DNS server to get what IPs belong to this server:

https://www.whatsmydns.net/dns-lookup?query=google.com&server=google



Record	Type	Value	TTL
google.com	A	172.253.122.113	77
google.com	A	172.253.122.100	77
google.com	A	172.253.122.102	77
google.com	A	172.253.122.138	77
google.com	A	172.253.122.101	77
google.com	A	172.253.122.139	77

https://www.whatsmydns.net/dns-lookup?query=airbnb.com&server=google



Record	Type	Value	TTL
airbnb.com	A	54.210.148.27	47
airbnb.com	A	54.162.120.58	47
airbnb.com	A	52.5.118.152	47

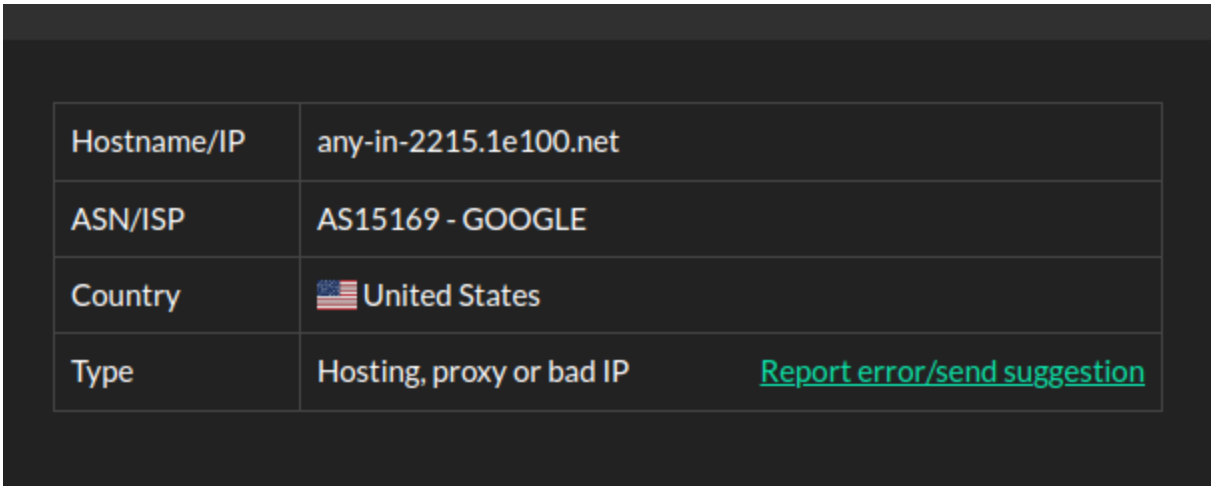
This DNS knowledge is simple and important to narrow down IPs

ASN

Have you ever asked yourself, who “owns” the IPs numbers? You don’t own it even if you pay, you only get to use the internet but don’t own any IP assigned to you.

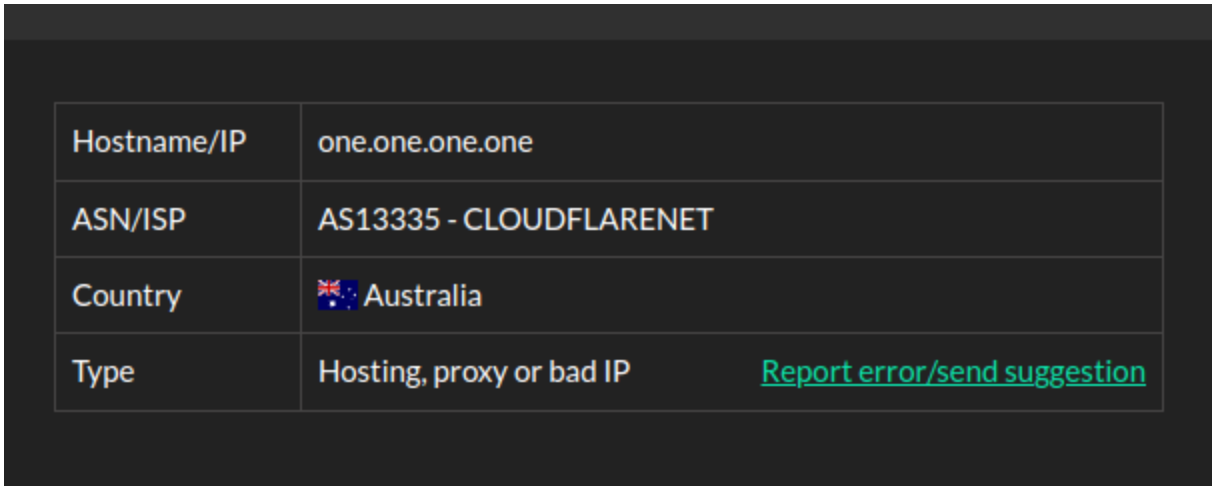
The “owners” of these IPs are entities on a system called ASN(Autonomous System Number).
These are a organizations responsibles for these IPs, so lets say I want to know who “owns” the IP 216.239.34.21, we can use online free tools to verify:

https://iphub.info/?ip=216.239.34.21



Hostname/IP	any-in-2215.1e100.net
ASN/ISP	AS15169 - GOOGLE
Country	United States
Type	Hosting, proxy or bad IP Report error/send suggestion

https://iphub.info/?ip=1.1.1.1



Hostname/IP	one.one.one.one
ASN/ISP	AS13335 - CLOUDFLARENET
Country	Australia
Type	Hosting, proxy or bad IP Report error/send suggestion

Etc…
We are interested on the field “ASN”, so if you check well enough IPs you will see there are multiple ASN, and they are easy identifiable, because the ids follows a known pattern, like this:

ASN[number]

So each ASN organization will “own” a set of IP range, so for example

AS13335,Cloudflare, Inc. owns multiple range IPs like:

- 1.0.0.0 - 1.0.0.255
- 1.1.1.0 - 1.1.1.255
- 8.6.144.0 - 8.6.146.255
- 8.9.231.0 - 8.9.231.255
- 8.10.148.0 - 8.10.148.255

… there are lot more, you can make a research to find them all

Or for example

AS15169,Google LLC, also owns multiple IP range

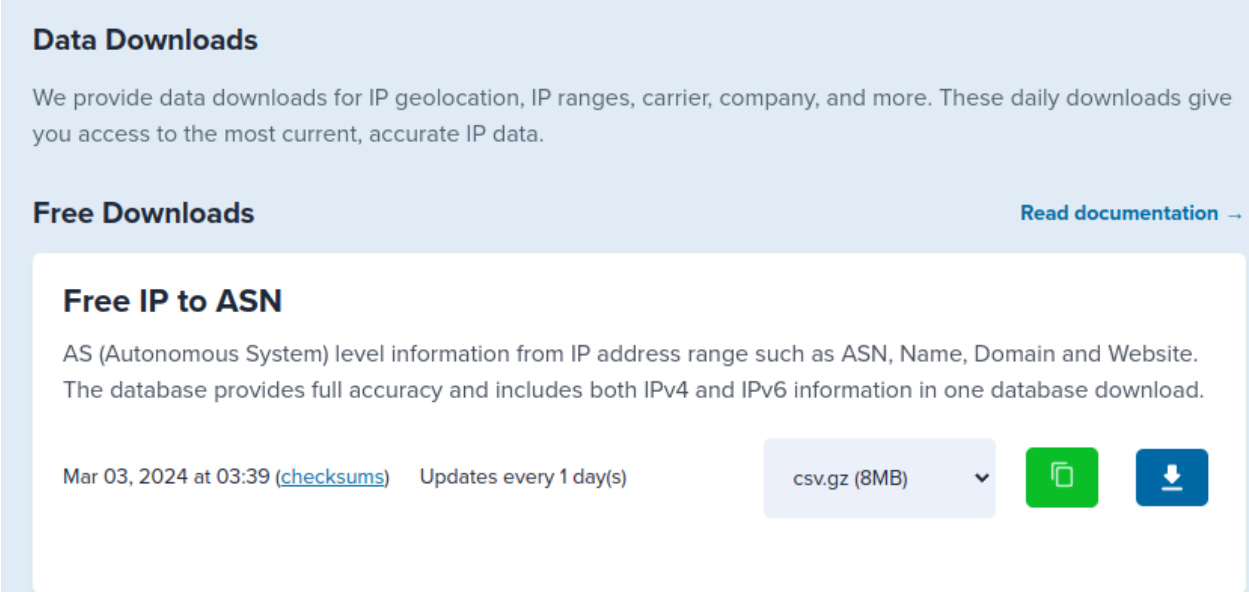
- 8.8.4.0 - 8.8.4.255
- 8.8.8.0 - 8.8.8.255
- 8.35.200.0 - 8.35.207.255
- 34.0.0.0 - 34.0.15.255

… there are lot more, you can make a research to find them all

Etc…

Each ASN owns a multiple range of IPs, this is crucial for the first optimization we are going to make.
Anyway so you can read more on internet if you are interested
https://www.cloudflare.com/learning/network-layer/what-is-an-autonomous-system/

You can get an ASN dataset for free https://ipinfo.io/account/data-downloads you need to sign up to get this dataset



This is gonna be very useful for this project

Optimizations

Unless you are maniac, you can create a bot and send requests to every possible IP and eventually you will find the real server IP but it will take years, sure you can distribute the work to multiple machines and still take a lot of effort, time and resources.
But we can do better right?, of course we can.

These are the optimization we gonna see on this report:

- When sending the http requests, use HTTP first, then HTTPS to verify, https is slower for this kind of projects, that's why we check with http.

- Once we find out the IP from an ASN, stop processing that ASN, this is because you will find out that most of the websites use load balancers, so there could be a lot of IPs; we can stop the whole search process if we find an IP has the ssl certificate for that domain.

- I'm using a column major like approach algorithm for faster searches

- Finding out what ASN/host they use to narrow down the IPs to analyze, we can do this by:
-

    1. Check API endpoints for the domain, on web development most of the time to get response from the server there is a separation between api and the html, you will find for example: the url is https://example.com but the API they used is https://api.example.com and if they didn't protected the api endpoint, we'll be able to know what ASN they used.
    2. Investigating the historical DNS configuration, so let's say the first time a developer creates the website the domain is not configured to use WAF by default, so the first IP they used is exposed, websites like https://securitytrails.com and many others save the DNS historical information and we will use it for this project.
    3. Check subdomains, so let's say we have https://example.com [production], https://dev.example.com [dev], and

https://staging.example.com [staging], the goal will be to find out if this development environments have WAF protection

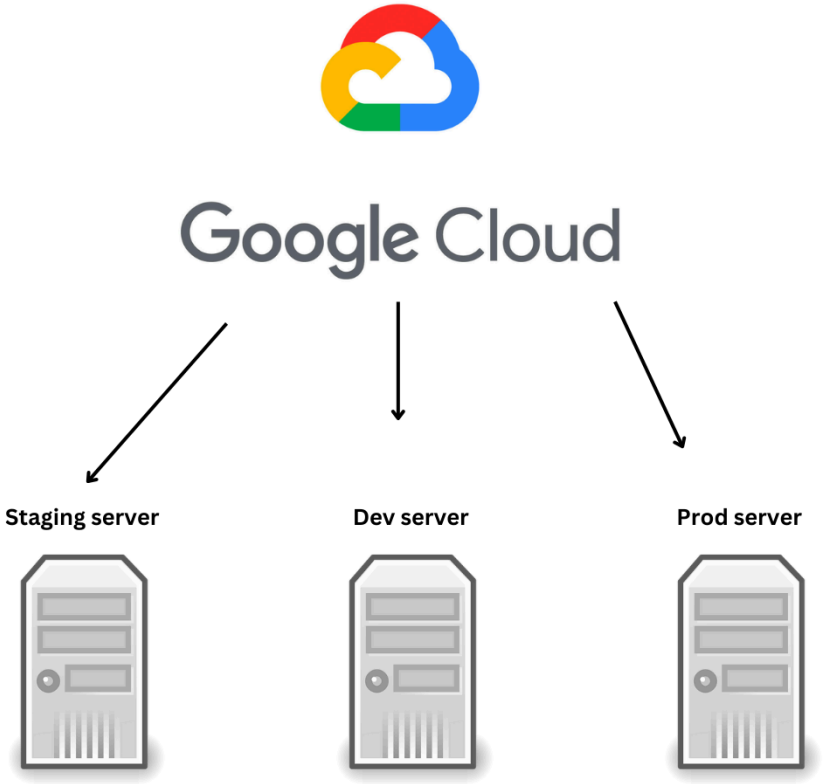
4. Social media/blog post, there are some websites that post public blogs about what tech they used, so lets say they create a post called “how [domain name] handle multiple requests at once”, and in this blog post they mentioned they use Google as host provider, with this information we can just use Google ASN to narrow down the IPs

HOST Providers

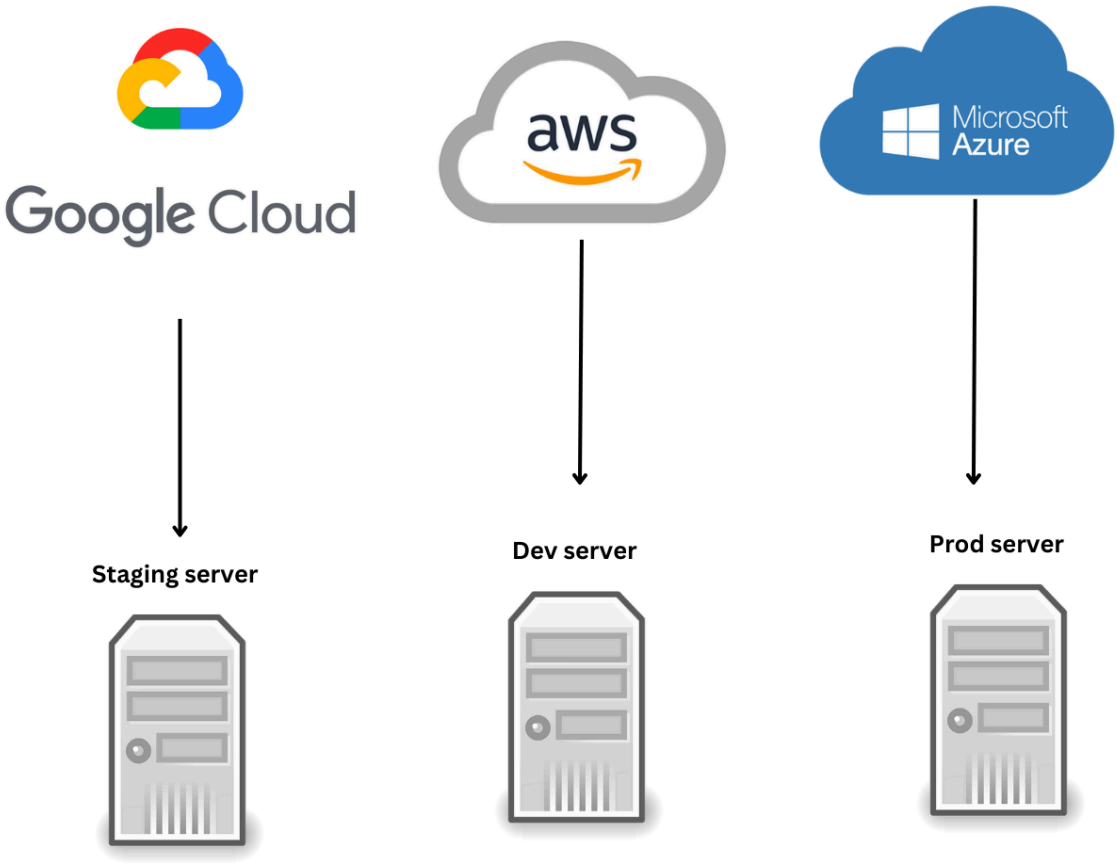
You must already know that there are multiple companies that offers hosting services
Been the most popular: google cloud, aws, microsoft azure

It's a good practice to have staging, dev and production environments all in one single hosting provider.

So if you picked Google cloud(preferred to me), they all are on different server but the provider is the same:



We can’t be friends if you have a design your app like this:



So if you find any of dev or staging server are not protected by the WAF system, we will be able to narrow down the IPs by the ASN name used on dev or staging

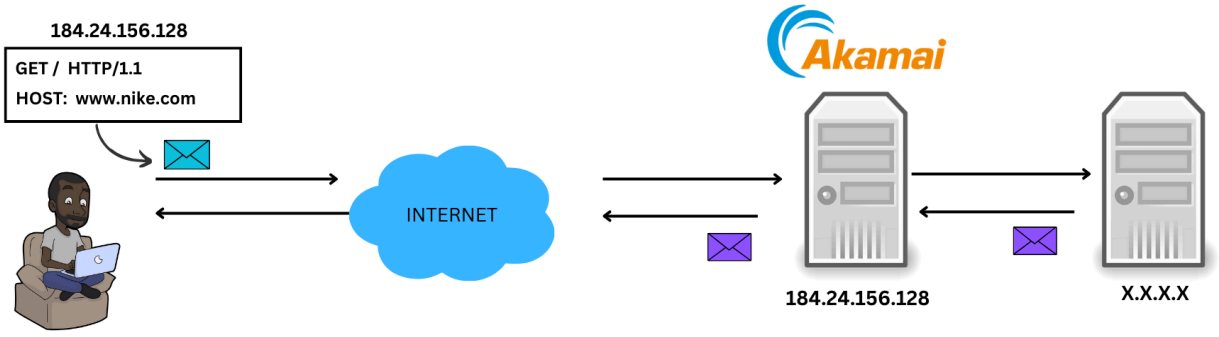
Examples

Let's use [security trails]() to investigate the DNS history for some websites

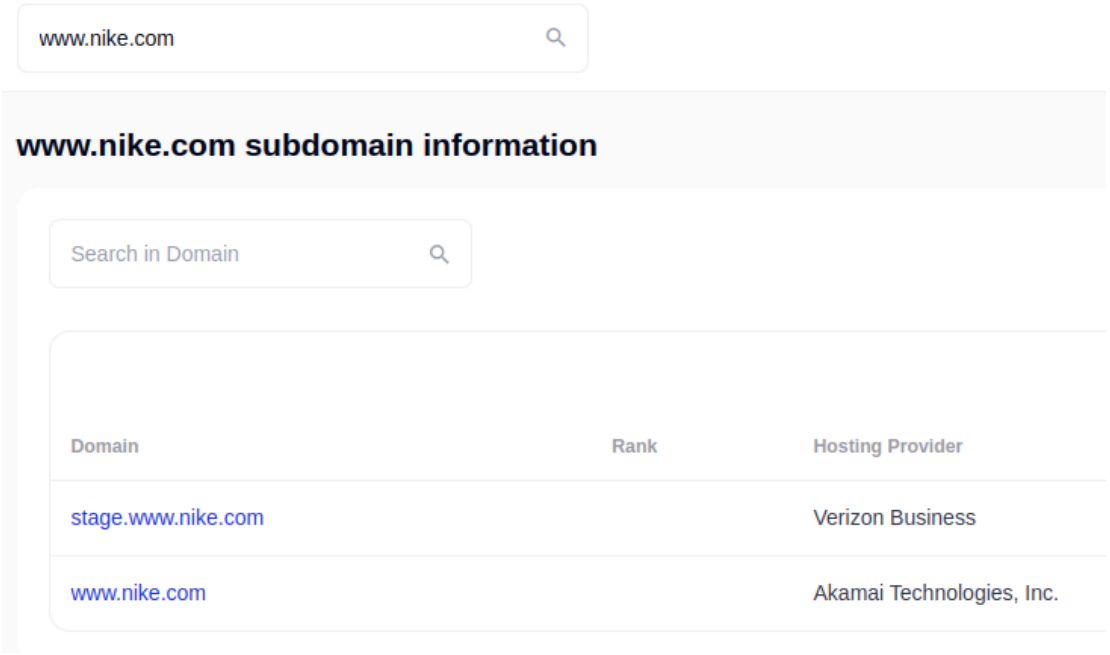
Nike

Website: https://www.nike.com
Security trails: https://securitytrails.com/domain/www.nike.com/dns

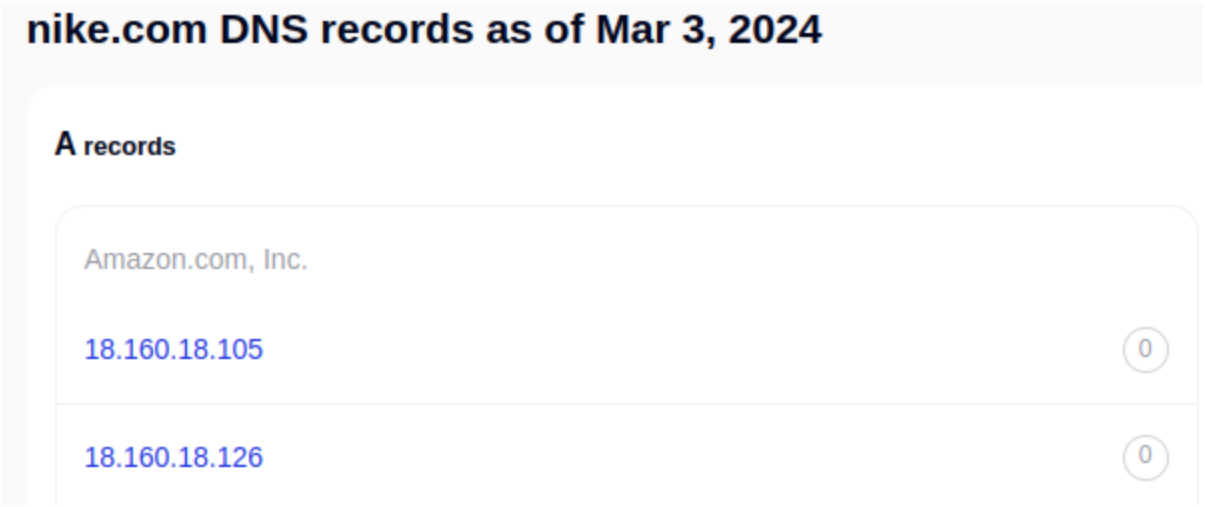
It uses Akamai as WAF service, in my opinion the best WAF so far along with imperva. I struggled a lot to create bots for websites using these services. Let see how the diagram looks like:



Let's check subdomains:
https://securitytrails.com/list/apex_domain/www.nike.com



There is a subdomain which uses Verizon as an ASN provider, so we'll add that ASN name to the code to narrow down the IPs.
Let's check now: https://nike.com
https://securitytrails.com/domain/nike.com/dns



It uses Amazon as an ASN provider, we'll add that ASN to the code too.
Let's Check subdomains https://securitytrails.com/list/apex_domain/nike.com

nike.com

Domain	Rank	Hosting Provider
nike.com	465	Amazon.com, Inc.
store.nike.com	13,386	Amazon.com, Inc.
news.nike.com	23,269	Akamai Technologies, Inc
about.nike.com	32,647	Amazon.com, Inc.
purpose.nike.com	99,664	Akamai Technologies, Inc
jobs.nike.com	153,546	Akamai Technologies, Inc
help-en-us.nike.com	178,261	Amazon.com, Inc.
inside.nike.com	233,297	-
communityimpact.nike.com	400,467	Amazon.com, Inc.
forums.nike.com	644,838	Amazon.com, Inc.
nikeplus.nike.com	793,356	Amazon.com, Inc.
engineering.nike.com	806,872	Fastly, Inc.
investors.nike.com	815,915	Cloudflare, Inc.
nikeid.nike.com	818,954	-
nikerunning.nike.com	837,188	-
secure-nikeplus.nike.com	967,090	Amazon.com, Inc.

There are multiple ASN names besides akamai, we'll take Fastly and Salesforce from there and put it on the code to narrow down the IPs.

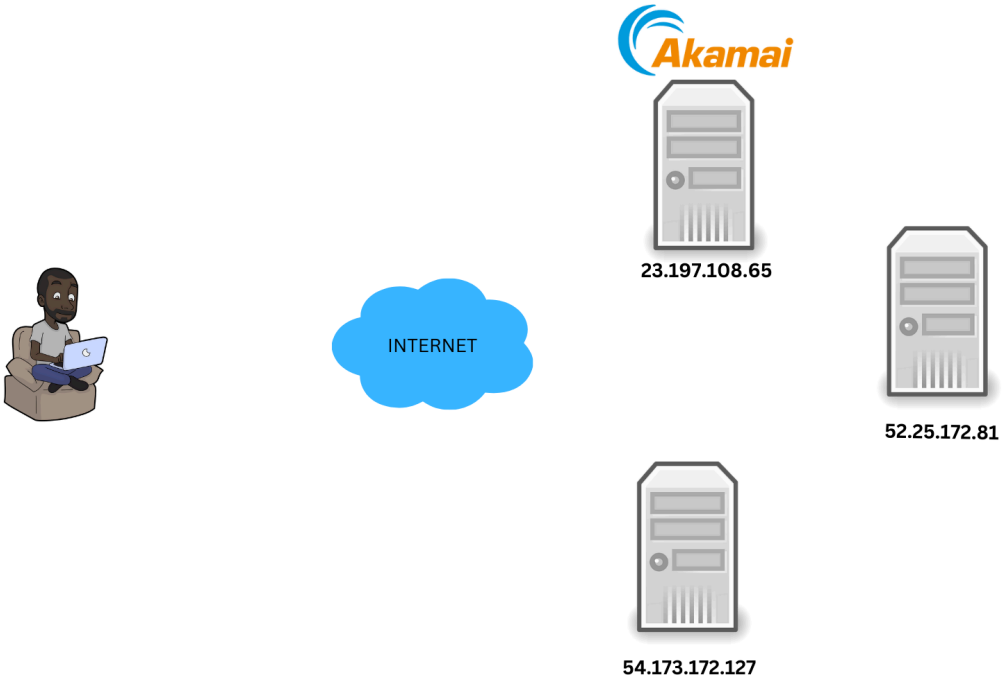
If we run the code with these filters:

```
var Nike = Input{
    URL:         utils.ParseURL("https://www.nike.com"),
    Keyworkds:   []string{"Nike delivers", "athletes"},
    TCPTimeout: time.Millisecond * 150,
    BufferSize: 2048,
    Asn: asn.Asn{
        PrioritiesNames: []string{"Verizon", "amazon", "Fastly", "Salesforce"},
        ForbiddenNames:  ForbidenASN,
    },
}
```

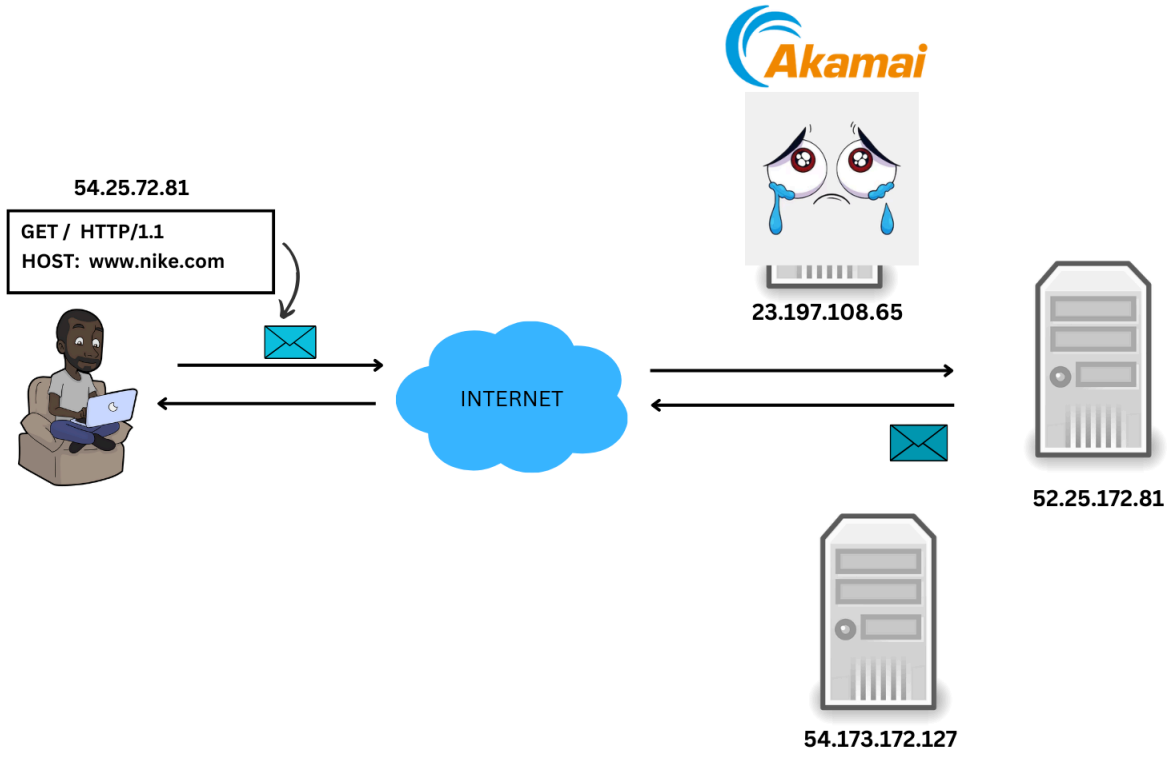
we get this results back:

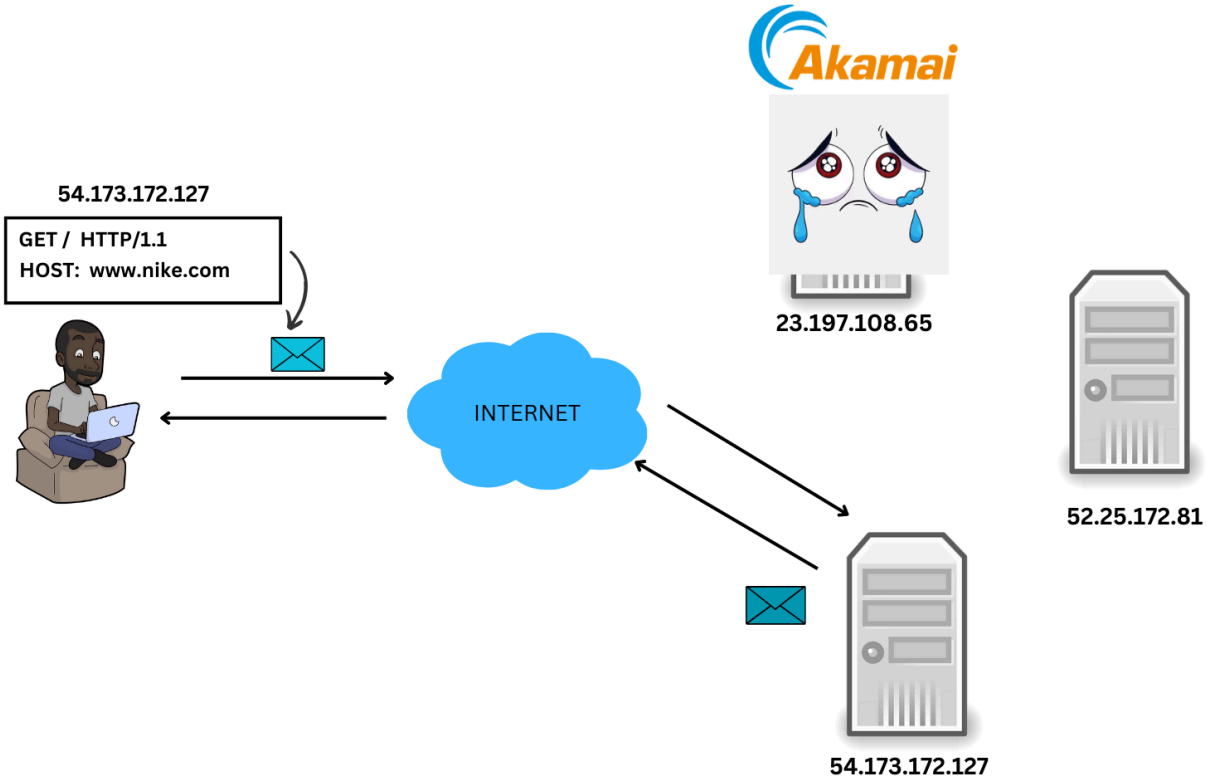

```
{
    "Domain": "www.nike.com",
    "AsnsFound": [
        {
            "ID": "as16509",
            "Name": "amazon.com, inc.",
            "IPs": [
                {
                    "IP": "52.25.172.81",
                    "HashSSLVerified": false
                }
            ]
        },
        {
            "ID": "as14618",
            "Name": "amazon.com, inc.",
            "IPs": [
                {
                    "IP": "54.173.172.127",
                    "HashSSLVerified": false,
                    "CommonSSLCNNames": [
                        "prod.ops.smartbear.io",
                        "Amazon RSA 2048 M02",
                        "Amazon Root CA 1",
                        "Starfield Services Root Certificate Authority - G2"
                    ]
                }
            ]
        }
    ]
}
```

Why multiple IPs you might ask, probably because they use akamai as load balancer to distribute the load on those servers, this is speculation, at the end of this report I'll add a section for all other reasons that server could have more than 1 IP, lets see how it looks now:



We can connect directly to Nike servers:



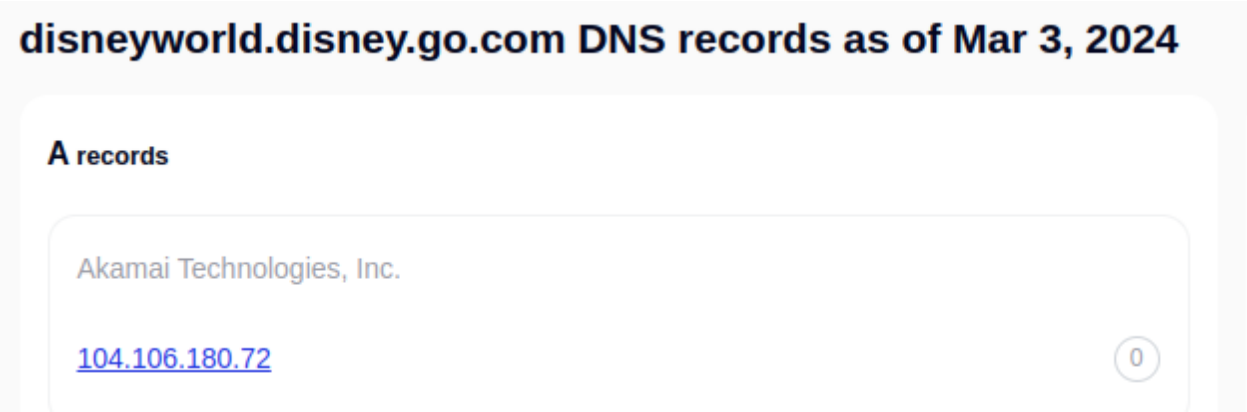


So for this case if you make a database DDoS attack, you'll probably take down one server but there are more servers to back it up.
This is why it's important to use microservices and not monolithic architecture, which is a topic I've seen on twitter a lot, with a monolithic approach you have only on point of failure, if the server crashes it's over for your website, however if you have a microservices approach, database DDoS attacks won't do harm, because if the server crashes, there are more servers to back it up.

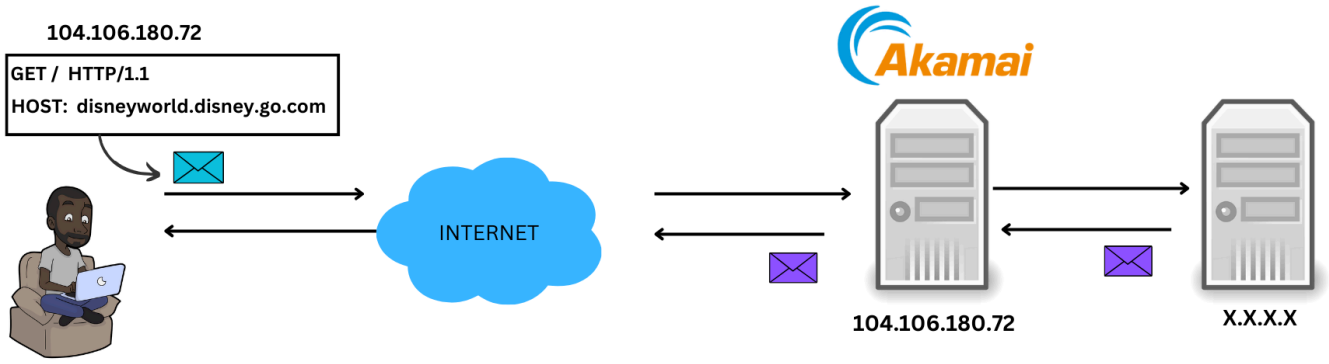
Disney

Website: https://disneyworld.disney.go.com

Security trails:
https://securitytrails.com/domain/disneyworld.disney.go.com/dns



It uses Akamai.
Let see how the diagram looks like:



Let's investigate the DNS information:

Domain	Rank	Hosting Provider
disneyworld.disney.go.com	9,875	Akamai Technologies, Inc.
reviews.disneyworld.disney.go.com	8,837,895	Rackspace Hosting
latest.disneyworld.disney.go.com		Akamai Technologies, Inc.
m.disneyworld.disney.go.com		-
guide.disneyworld.disney.go.com		Disney Worldwide Services, Inc.
lt01.disneyworld.disney.go.com		Akamai Technologies, Inc.
www.disneyworld.disney.go.com		Google LLC
stage.disneyworld.disney.go.com		Akamai Technologies, Inc.
sl.disneyworld.disney.go.com		Akamai Technologies, Inc.
prodc.disneyworld.disney.go.com		Akamai Technologies, Inc.
guide-sl.disneyworld.disney.go.com		Disney Worldwide Services, Inc.

It uses Rackspace, Google, Disney worldwide ASN, if we put it on the code, we should get something like this:

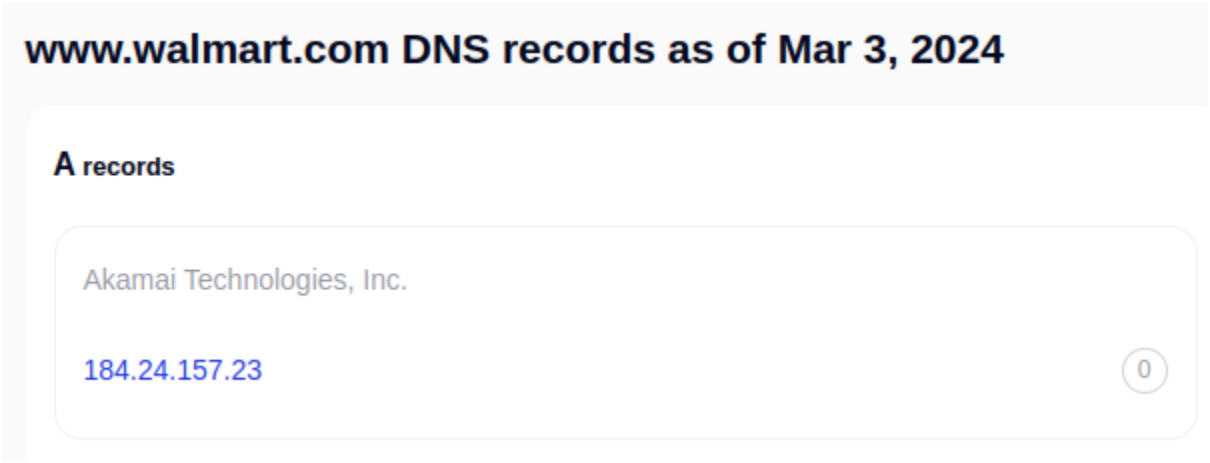
```
var Disney = Input{
    URL:         utils.ParseURL("https://disneyworld.disney.go.com"),
    Keyworkds:  []string{"50th_anniversary_countdown_clock"},
    TCPTimeout: time.Millisecond * 150,
    BufferSize: 2048,
    Asn: asn.Asn{
        PrioritiesNames: []string{"Rackspace Hosting", "Disney Worldwide", "Google"},
        ForbiddenNames:  ForbidenASN,
    },
}
var Walmart = Input{
```

we get this results back:


```
{
    "Domain": "disneyworld.disney.go.com",
    "AsnsFound": [
        {
            "ID": "as16591",
            "Name": "google fiber inc.",
            "IPs": [
                {
                    "IP": "23.228.128.196",
                    "HashSSLVerified": false,
                    "CommonSSLCNNames": [
                        "a248.e.akamai.net",
                        "DigiCert TLS RSA SHA256 2020 CA1"
                    ]
                }
            ]
        },
        {
            "ID": "as396982",
            "Name": "google llc",
            "IPs": [
                {
                    "IP": "34.96.229.8",
                    "HashSSLVerified": false,
                    "CommonSSLCNNames": [
                        "a248.e.akamai.net",
                        "DigiCert TLS RSA SHA256 2020 CA1"
                    ]
                }
```

Walmart

Website: https://www.walmart.com
Security trails: https://securitytrails.com/domain/www.walmart.com/dns



You already know the diagram by now, let's investigate the DNS information:
https://securitytrails.com/domain/www.walmart.com/history/a?page=5

23.1.77.21	Akamai Technologies, Inc.	2017-07-19 (7 years)	2017-07-21 (7 years)	2 days
23.217.159.113	Akamai Technologies, Inc.	2017-07-04 (7 years)	2017-07-06 (7 years)	2 days
104.117.83.63	Akamai International B.V.	2017-06-01 (7 years)	2017-06-03 (7 years)	2 days
23.32.145.194	Akamai Technologies, Inc.	2017-05-30 (7 years)	2017-06-01 (7 years)	2 days
161.170.244.20 161.170.248.20	Wal-Mart Stores, Inc.	2013-01-16 (11 years)	2013-01-18 (11 years)	2 days
161.170.248.20	Wal-Mart Stores, Inc.	2010-09-18 (13 years)	2010-09-20 (13 years)	2 days

So one ASN name is Wal-mart

https://securitytrails.com/list/apex_domain/walmart.com

walmart.com

walmart.com	200	Akamai Technologies, Inc.
corporate.walmart.com	4,386	Akamai Technologies, Inc.
news.walmart.com	31,435	Rackspace Hosting
goto.walmart.com	74,893	Google LLC
blog.walmart.com	81,758	Rackspace Hosting
grocery.walmart.com	91,026	Akamai Technologies, Inc.
careers.walmart.com	111,228	Rackspace Hosting
stock.walmart.com	129,125	Cloudflare, Inc.
linksynergy.walmart.com	146,522	-
marketplace.walmart.com	253,383	Google LLC

walmart.com

plus.walmart.com	Akamai International B.V.
iotedge-01.s00393.vmi.walmart.com	Microsoft Corporation
stg-payment.walmart.com	Wal-Mart Stores, Inc.
supportchat.walmart.com	Fastly, Inc.
wmtmanagedb2c-identity.walmart.com	Akamai Technologies, Inc.
prospecart.walmart.com	Steadfast
ak-ndc-grocery-qa.walmart.com	Wal-Mart Stores, Inc.
caq-vemail.walmart.com	NTT America, Inc.
devhomeloans.walmart.com	AT&T Services, Inc.
api.stg.walmart.com	-
campus.walmart.com	Cloudflare, Inc.
wwwstagephoto.walmart.com	Wal-Mart Stores, Inc.

There are more ASN, so the filters look like this:

```
var Walmart = Input{
    URL:         utils.ParseURL("https://www.walmart.com"),
    Keyworkds:   []string{"walmartimages", "Save Money", "free delivery"},
    TCPTimeout: time.Millisecond * 150,
    BufferSize: 3048,
    Asn: asn.Asn{
        PrioritiesNames: []string{"Rackspace", "Wal-Mart", "CyrusOne", "google",
        "amazon", "NTT America", "AT&T Services", "Steadfast"},
        ForbiddenNames:  ForbidenASN,
    },
}
```

we get this results back:


```
{
    "Domain": "www.walmart.com",
    "AsnsFound": [
        {
            "ID": "as22773",
            "Name": "cox communications inc.",
            "IPs": [
                {
                    "IP": "68.106.122.6",
                    "HashSSLVerified": false,
                    "CommonSSLCNNames": [
                        "a248.e.akamai.net",
                        "DigiCert TLS RSA SHA256 2020 CA1"
                    ]
                },
                {
                    "IP": "68.106.122.5",
                    "HashSSLVerified": false,
                    "CommonSSLCNNames": [
                        "a248.e.akamai.net",
                        "DigiCert TLS RSA SHA256 2020 CA1"
                    ]
                },
                {
                    "IP": "68.106.122.8",
                    "HashSSLVerified": false,
                    "CommonSSLCNNames": [
                        "a248.e.akamai.net",
                        "DigiCert TLS RSA SHA256 2020 CA1"
                    ]
```

Adidas

Website: https://www.adidas.com/us
Security trails: https://securitytrails.com/domain/www.adidas.com/dns

https://securitytrails.com/domain/www.adidas.com/history/a?page=3

104.70.189.96	Akamai International B.V.	2021-02-10 (3 years)	2021-02-17 (3 years)	7 days
104.70.189.96	Akamai International B.V.	2021-01-15 (3 years)	2021-02-07 (3 years)	23 days
104.70.189.96	Akamai International B.V.	2021-01-12 (3 years)	2021-01-14 (3 years)	2 days
104.70.189.96	Akamai International B.V.	2020-12-19 (3 years)	2021-01-09 (3 years)	21 days
23.210.196.109	Globe Telecoms	2020-12-15 (3 years)	2020-12-19 (3 years)	4 days
104.70.189.96	Akamai International B.V.	2020-12-02 (3 years)	2020-12-15 (3 years)	13 days

Domain	Rank	Hosting Provider
www.adidas.com		Akamai International B.V.
new-origin.www.adidas.com		IT Service Provider located in Nuernberg, Germany
vpdc.origin.www.adidas.com		Amazon.com, Inc.
new-test.origin.www.adidas.com		IT Service Provider located in Nuernberg, Germany
test.origin.www.adidas.com		IT Service Provider located in Nuernberg, Germany
origin.www.adidas.com		Amazon.com, Inc.

There are lot of ASN names
So the filters look like this:

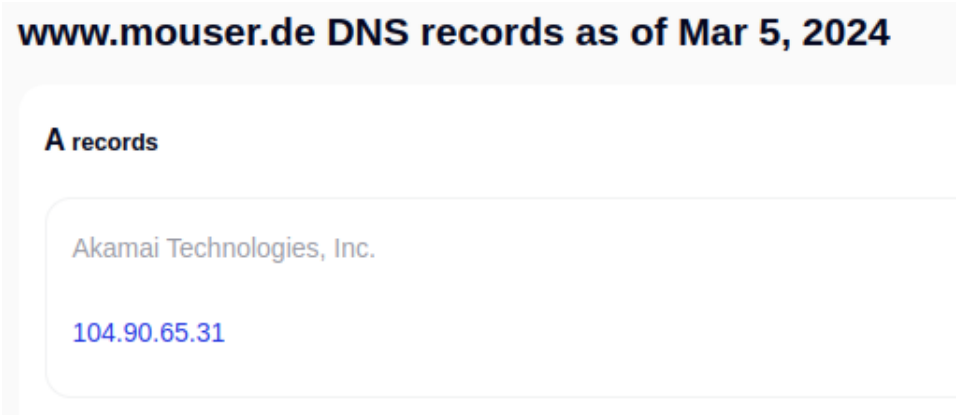
```
var Adidas = Input{
    URL:        utils.ParseURL("https://www.adidas.com/us"),
    Keyworkds:  []string{"adidas", "Official", "running"},
    TCPTimeout: time.Millisecond * 150,
    BufferSize: 2048,
    Asn: asn.Asn{
        PrioritiesNames: []string{"Globe Telecoms", "noris network", "CenturyLink Communications",
        "amazon"},
        ForbiddenNames:  ForbidenASN,
    },
}
```

we get this results back:


```
{
    "Domain": "www.adidas.com",
    "AsnsFound": [
        {
            "ID": "as4775",
            "Name": "globe telecoms",
            "IPs": [
                {
                    "IP": "23.33.48.4",
                    "HashSSLVerified": false,
                    "CommonSSLCNNames": [
                        "*.test.edgekey.net",
                        "DigiCert TLS RSA SHA256 2020 CA1"
                    ]
                }
            ]
        },
        {
            "ID": "as16509",
            "Name": "amazon.com, inc.",
            "IPs": [
                {
                    "IP": "75.2.29.124",
                    "HashSSLVerified": false,
                    "CommonSSLCNNames": [
                        "*.live.prod.spodn.com",
                        "R3"
                    ]
                }
            ]
```

Mouser

Website: https://www.mouser.de
Security trails: https://securitytrails.com/domain/www.mouser.de/dns



https://securitytrails.com/domain/mouser.de/dns

https://securitytrails.com/domain/www.mouser.de/history/a?page=3

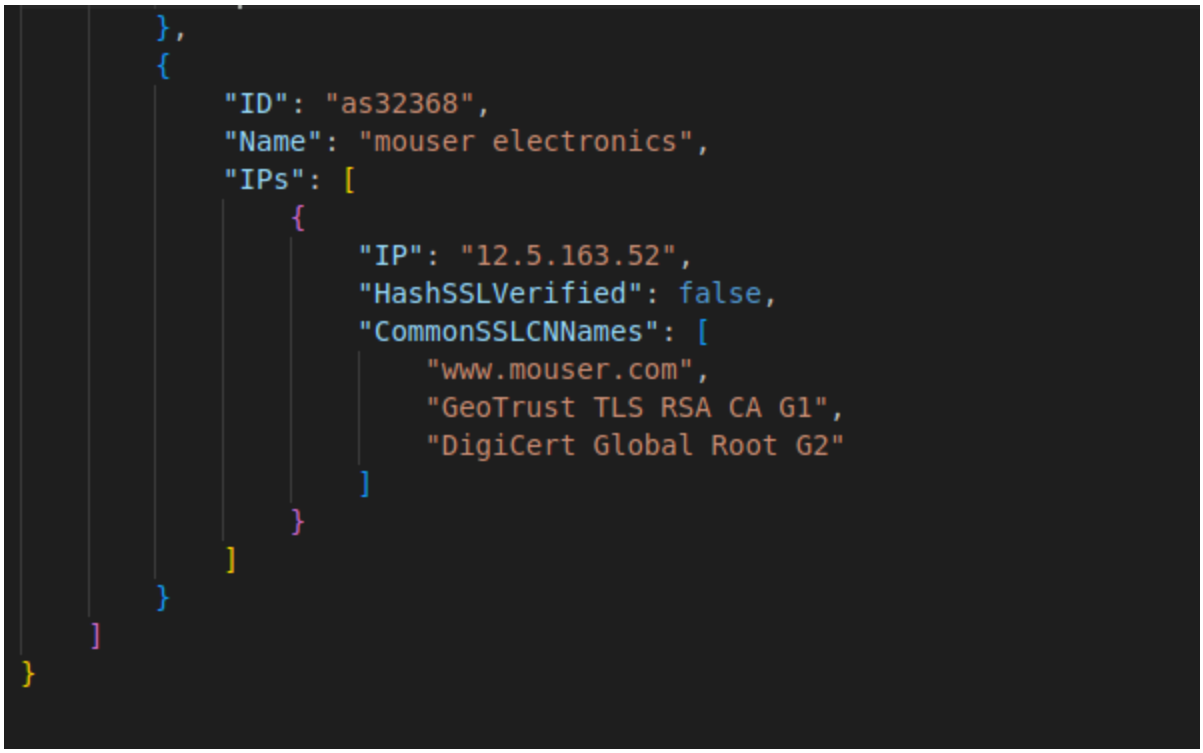
96.17.141.186	Akamai International B.V.	2020-05-16 (4 years)	2020-05-18 (4 years)	2 days
23.218.40.202	Rogers Communications Canada Inc.	2020-05-09 (4 years)	2020-05-11 (4 years)	2 days
23.218.40.202	Rogers Communications Canada Inc.	2020-05-06 (4 years)	2020-05-08 (4 years)	2 days

The filters look like this:

```
var Mouser = Input{
    URL:        utils.ParseURL("https://www.mouser.de"),
    TCPTimeout: time.Millisecond * 250,
    Keyworkds:  []string{"Distributor", "Deutschland"},
    BufferSize: 2048,
    Asn: asn.Asn{
        PrioritiesNames: []string{"mouser electronics", "Rogers Communications"},
        ForbiddenNames:  ForbidenASN,
    },
}
var VerifySos = Input{
```

And the final results looks like this:

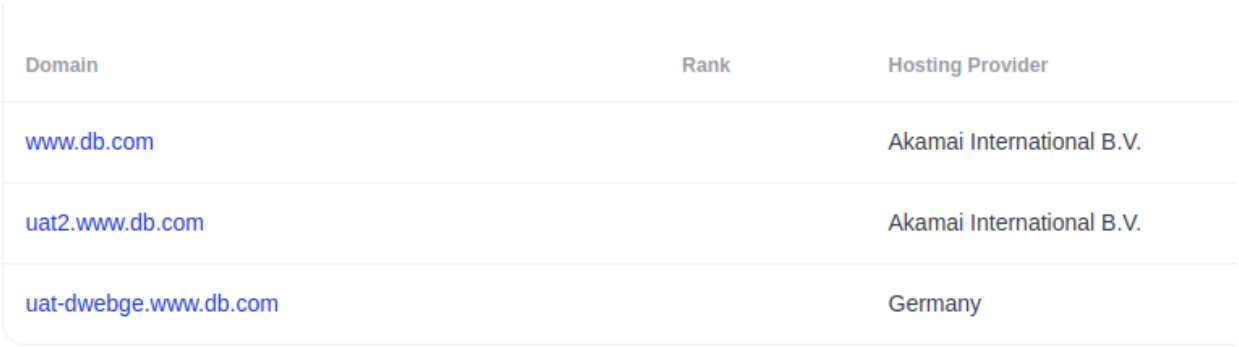

```
{
    "Domain": "www.mouser.de",
    "AsnsFound": [
        {
            "ID": "as812",
            "Name": "rogers communications canada inc.",
            "IPs": [
                {
                    "IP": "184.29.112.4",
                    "HashSSLVerified": false,
                    "CommonSSLCNNames": [
                        "www.casio-intl.com",
                        "DigiCert TLS RSA SHA256 2020 CA1"
                    ]
                },
                {
                    "IP": "23.218.40.2",
                    "HashSSLVerified": false,
                    "CommonSSLCNNames": [
                        "*.test.edgekey.net",
                        "DigiCert TLS RSA SHA256 2020 CA1"
                    ]
                },
                {
                    "IP": "104.71.244.4",
                    "HashSSLVerified": false,
                    "CommonSSLCNNames": [
                        "*.test.edgekey.net",
                        "DigiCert TLS RSA SHA256 2020 CA1"
                    ]
                },
```


},
{
"ID": "as32368",
"Name": "mouser electronics",
"IPs": [
{
"IP": "12.5.163.52",
"HashSSLVerified": false,
"CommonSSLCNNames": [
"www.mouser.com",
"GeoTrust TLS RSA CA G1",
"DigiCert Global Root G2"
]
}
]
}
]
}

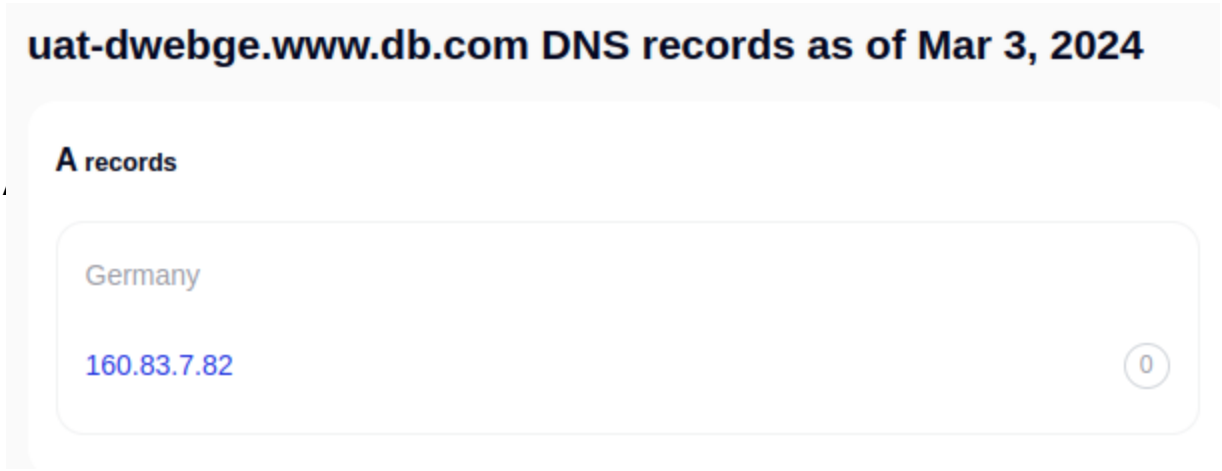
Deutsche bank

Website: https://www.db.com/index?language_id=1&kid=sl.redirect-en.shortcut
Security trails: https://securitytrails.com/domain/www.db.com/dns

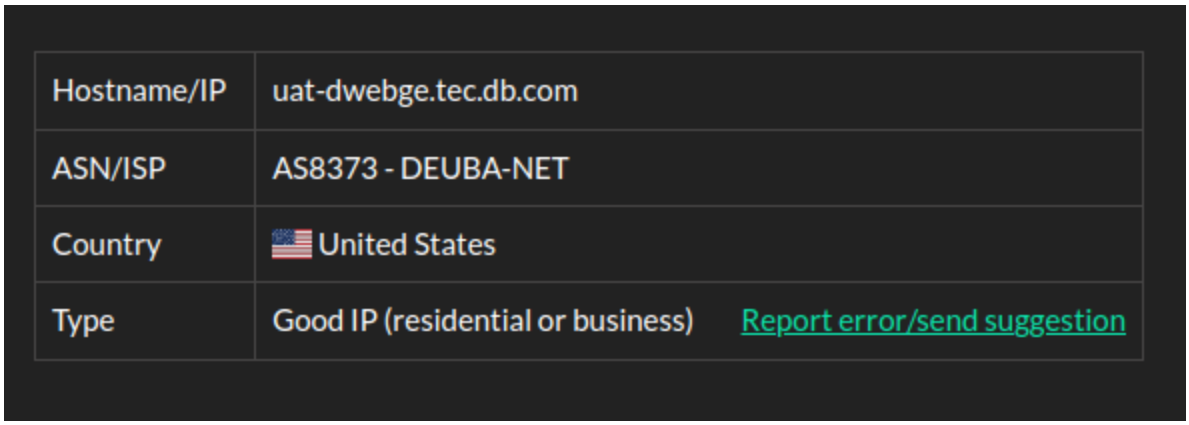
https://securitytrails.com/list/apex_domain/www.db.com

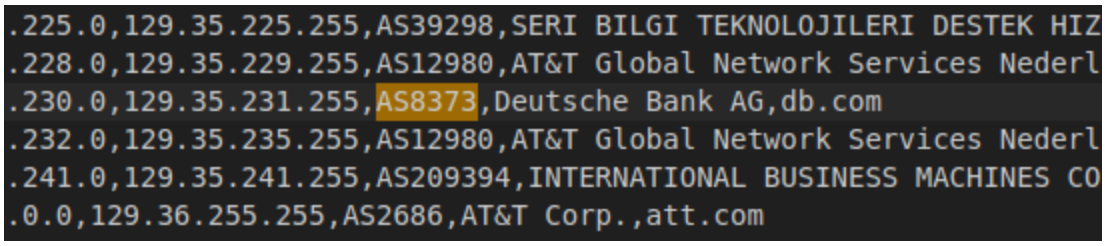


To verify the ASN we go to
https://securitytrails.com/domain/uat-dwebge.www.db.com/dns



https://iphub.info/?ip=160.83.7.82





https://securitytrails.com/domain/db.com/history/a

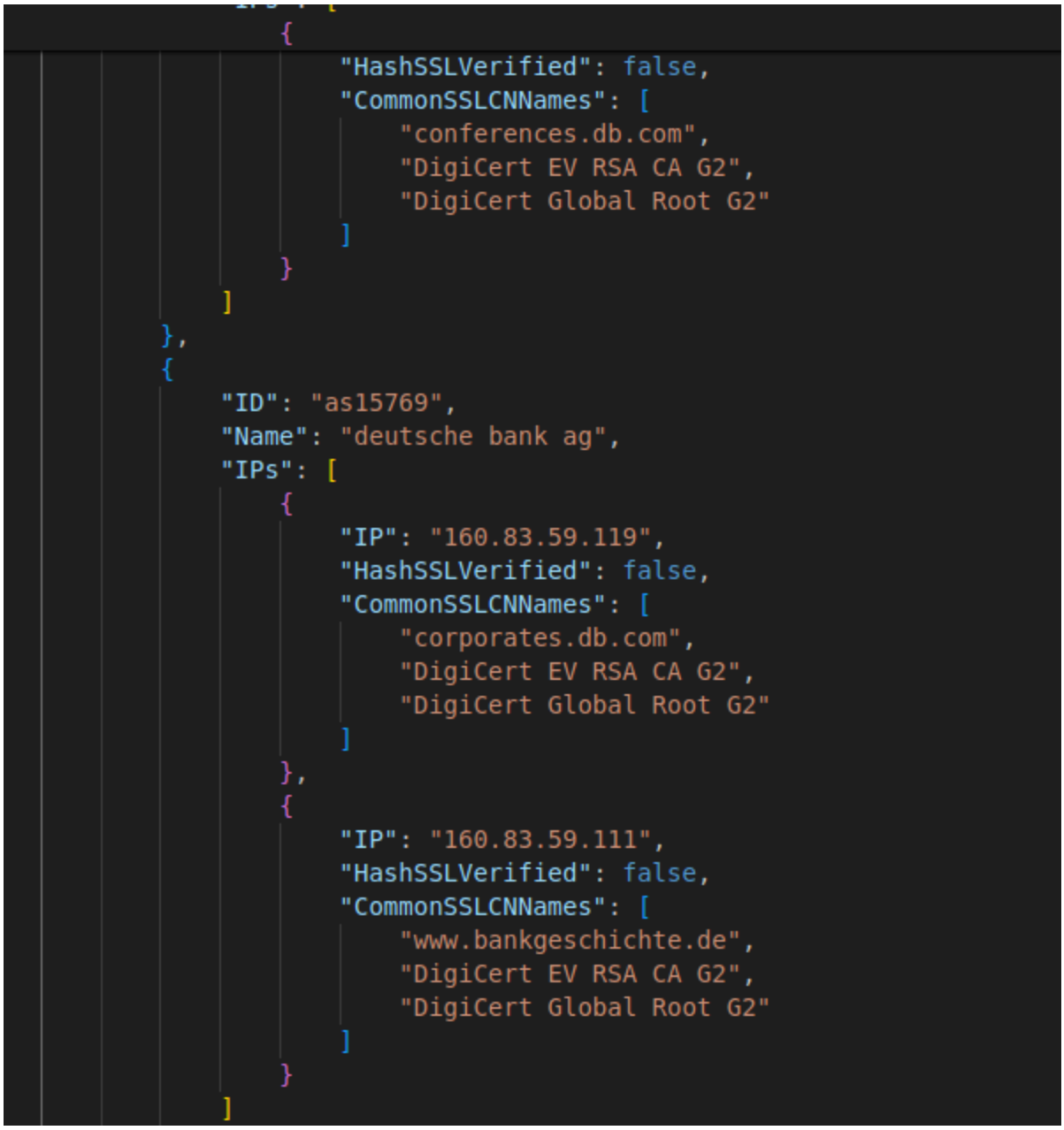
db.com	13,509	Germany
country.db.com	143,390	Akamai International B.V.
investor-relations.db.com	150,161	Akamai International B.V.
developer.db.com	487,126	Google LLC
pwm.db.com	536,849	Germany
etf.db.com	562,745	Plus.line AG
careers.db.com	1,136,634	Akamai International B.V.
corporates.db.com	1,447,370	Akamai International B.V.
deutschewealthonline.db.com	1,489,159	Akamai International B.V.
art.db.com	1,545,407	Akamai International B.V.
cib.db.com	1,619,960	Akamai International B.V.
research.db.com	1,661,625	Markit On Demand, Inc.
wtk.db.com	1,705,340	Germany
adr.db.com	1,726,732	Deutsche Bank AG

So filters looks like this:

```
var DeutscheBank = Input{
    URL:        utils.ParseURL("https://www.db.com/index?language_id=1&kid=sl.redirect-en.shortcut"),
    Keyworkds:  []string{"Deutsche Bank", "dwebcms"},
    TCPTimeout: time.Millisecond * 150,
    BufferSize: 2048,
    Asn: asn.Asn{
        PrioritiesNames: []string{"Deutsche Bank", "Markit On Demand", "google"},
        ForbiddenNames:  ForbidenASN,
    },
}
```

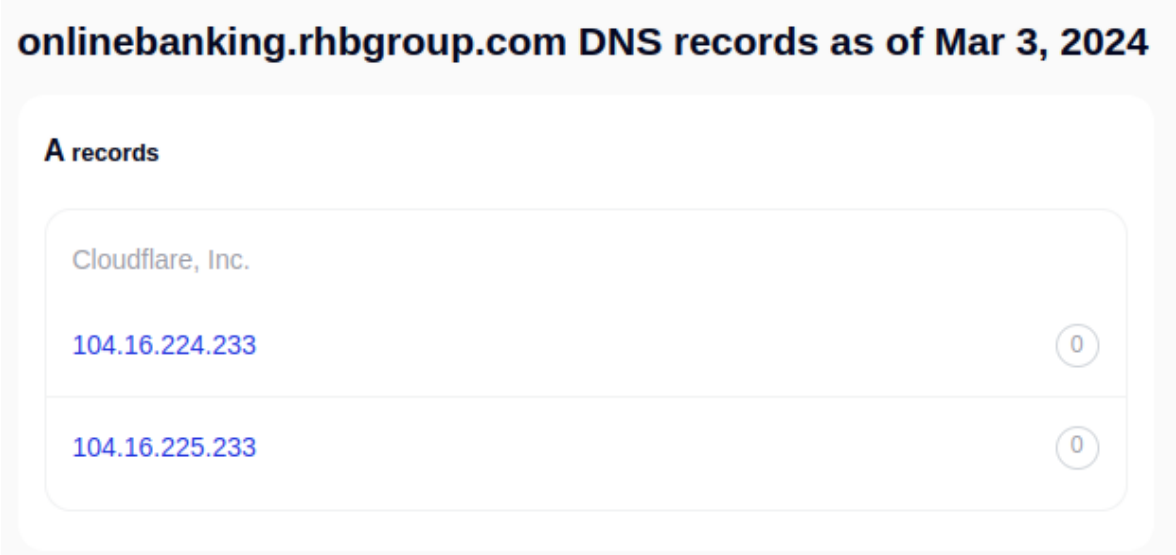
we get this results back:

```
{
    "Domain": "www.db.com",
    "AsnsFound": [
        {
            "ID": "as16591",
            "Name": "google fiber inc.",
            "IPs": [
                {
                    "IP": "23.228.128.196",
                    "HashSSLVerified": false,
                    "CommonSSLCNNames": [
                        "a248.e.akamai.net",
                        "DigiCert TLS RSA SHA256 2020 CA1"
                    ]
                },
                {
                    "IP": "23.228.128.201",
                    "HashSSLVerified": false,
                    "CommonSSLCNNames": [
                        "a248.e.akamai.net",
                        "DigiCert TLS RSA SHA256 2020 CA1"
                    ]
                },
                {
                    "IP": "23.228.128.200",
                    "HashSSLVerified": false,
                    "CommonSSLCNNames": [
                        "a248.e.akamai.net",
                        "DigiCert TLS RSA SHA256 2020 CA1"
```


{
"HashSSLVerified": false,
"CommonSSLCNNames": [
"conferences.db.com",
"DigiCert EV RSA CA G2",
"DigiCert Global Root G2"
]
}
]
},
{
"ID": "as15769",
"Name": "deutsche bank ag",
"IPs": [
{
"IP": "160.83.59.119",
"HashSSLVerified": false,
"CommonSSLCNNames": [
"corporates.db.com",
"DigiCert EV RSA CA G2",
"DigiCert Global Root G2"
]
},
{
"IP": "160.83.59.111",
"HashSSLVerified": false,
"CommonSSLCNNames": [
"www.bankgeschichte.de",
"DigiCert EV RSA CA G2",
"DigiCert Global Root G2"
]
}
]

RHB bank

Website: https://onlinebanking.rhbgroup.com/my/login
Security trails:
https://securitytrails.com/domain/onlinebanking.rhbgroup.com/dns



https://securitytrails.com/domain/rhbgroup.com/history/a

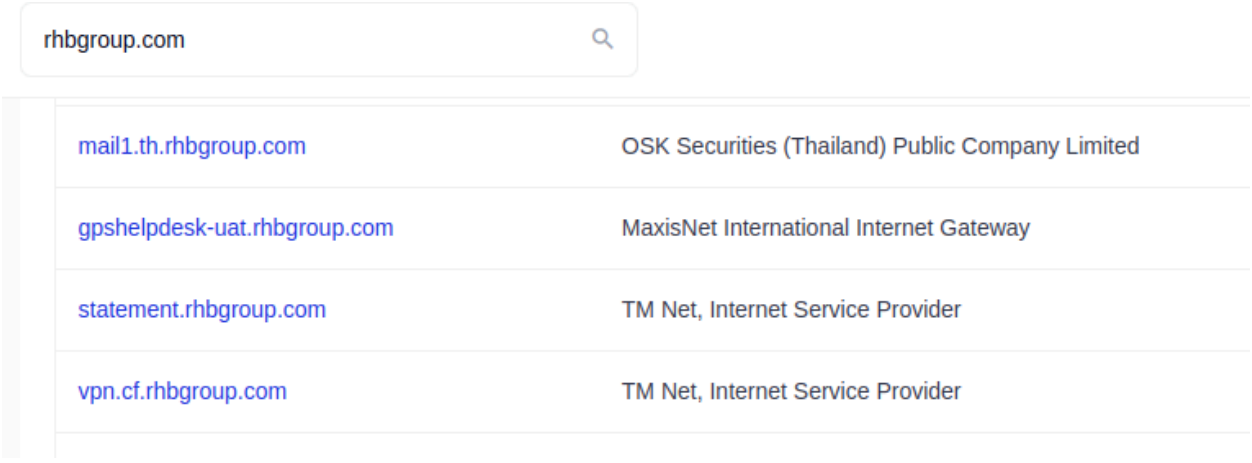
Sometime you will need to verify the ASN name because it shows incorrect ASN name for some reason:

https://iphub.info/?ip=112.137.173.39

Hostname/IP	112.137.173.39
ASN/ISP	AS17971 - TTSSB-MY
Country	Malaysia
Type	Mixed ISP (hosting & residential) Report error/send suggestion

```
143.255,AS4809,China Telecom Next Generation Carrier Network,china
150.255,AS17971,TM TECHNOLOGY SERVICES SDN. BHD.,tmone.com.my
151.255,AS18206,TM TECHNOLOGY SERVICES SDN. BHD.,tmone.com.my
159.255,AS17971,TM TECHNOLOGY SERVICES SDN. BHD.,tmone.com.my
175.255,AS135047,NTT NEW ZEALAND LIMITED,dimensiondata.com
```

https://securitytrails.com/list/apex_domain/rhbgroup.com



If you keep investigating, the filter will look like this:

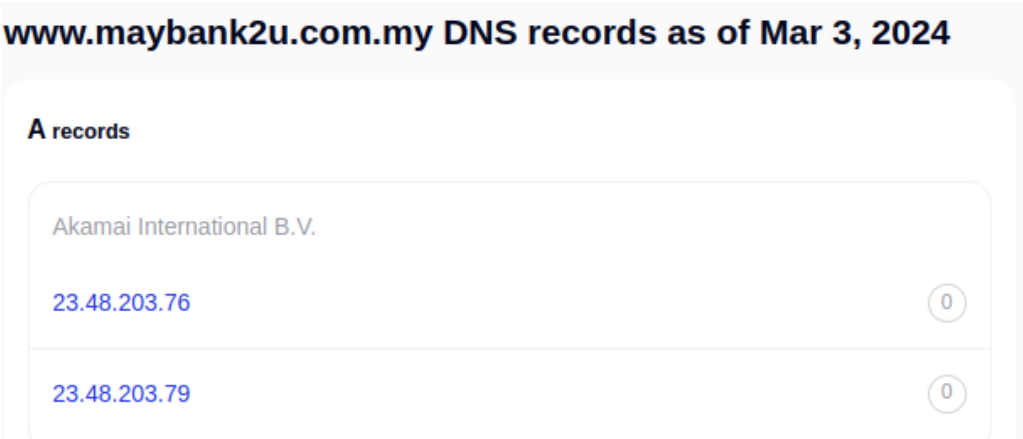
```
var Rhbgroup = Input{
    URL:         utils.ParseURL("https://onlinebanking.rhbgroup.com/my/login"),
    Keyworkds:   []string{"Online banking"},
    TCPTimeout: time.Millisecond * 150,
    BufferSize: 2048,
    Asn: asn.Asn{
        PrioritiesNames: []string{"OSK Securities", "TM TECHNOLOGY SERVICES", "MaxisNet International Int",
        "TTNET-MY", "Amazon", "Binariang Berhad", "Digital Singapore", "AIMS Data Centre", "PT Cyberindo",
        "Forcepoint Cloud", "Telekomunikasi", "SingNet", "Gilead Sciences", "Cyberindo", "HGC Global"},
        ForbiddenNames:  ForbidenASN,
    },
}
```

we get this results back:


```
{
    "Domain": "onlinebanking.rhbgroup.com",
    "AsnsFound": [
        {
            "ID": "as16509",
            "Name": "amazon.com, inc.",
            "IPs": [
                {
                    "IP": "52.220.33.94",
                    "HashSSLVerified": false,
                    "CommonSSLCNNames": [
                        "ap-core01.issuebook.ihsmarkit.com",
                        "Amazon RSA 2048 M02",
                        "Amazon Root CA 1",
                        "Starfield Services Root Certificate Authority - G2"
                    ]
                }
            ]
        },
        {
            "ID": "as14618",
            "Name": "amazon.com, inc.",
            "IPs": [
                {
                    "IP": "3.91.14.114",
                    "HashSSLVerified": false,
                    "CommonSSLCNNames": [
                        "www.example.com"
                    ]
```

Maybank2u bank

Website: https://www.maybank2u.com.my/home/m2u/common/login.do
Security trails: https://securitytrails.com/domain/www.maybank2u.com.my/dns



https://securitytrails.com/list/apex_domain/maybank2u.com.my

Domain	Rank	Hosting Provider
maybank2u.com.my	32,201	Akamai International B.V.
emerchant.maybank2u.com.my	865,147	VADS Managed Business Internet Service Provider
apps5.maybank2u.com.my		VADS Managed Business Internet Service Provider
ebpp.maybank2u.com.my		VADS Managed Business Internet Service Provider
payee.maybank2u.com.my		VADS Managed Business Internet Service Provider
warrants.maybank2u.com.my		Amazon.com, Inc.
mobile.maybank2u.com.my		VADS Managed Business Internet Service Provider
stagingm2umobile.maybank2u.com.my		Arcnet NTT MSC ISP
ecossitpf2.maybank2u.com.my		VADS Managed Business Internet Service Provider
orig-1n-www.maybank2u.com.my		Binariang Berhad

Hostname/IP	202.162.17.80
ASN/ISP	AS18206 - TTSSB-MY
Country	Malaysia
Type	Good IP (residential or business) Report error/send suggestion

```
4.197.223.255,AS9971,SK Broadband Co Ltd,skbroadband.com
4.197.239.255,AS18206,TM TECHNOLOGY SERVICES SDN. BHD.,tmone.com.my
198.29.255,AS4766,Korea Telecom,kt.com
.198.30.255,AS10195,HAIonNet,haion.net
```

This is the final filter:

```
var Maybank2u = Input{
    URL:         utils.ParseURL("https://www.maybank2u.com.my/home/m2u/common/login.do"),
    Keyworkds:   []string{"Maybank2u", "Malaysia"},
    TCPTimeout: time.Millisecond * 250,
    BufferSize: 2048,
    Asn: asn.Asn{
        PrioritiesNames: []string{"Binariang Berh", "Philippine Long Distan", "NTT America",
        "CenturyLink", "Net Onboard Sdn", "Arcnet", "TM TECHNOLOGY", "Reliance Jio Infoco"},
        ForbiddenNames:  ForbidenASN,
    },
}
```

This is the final filter:

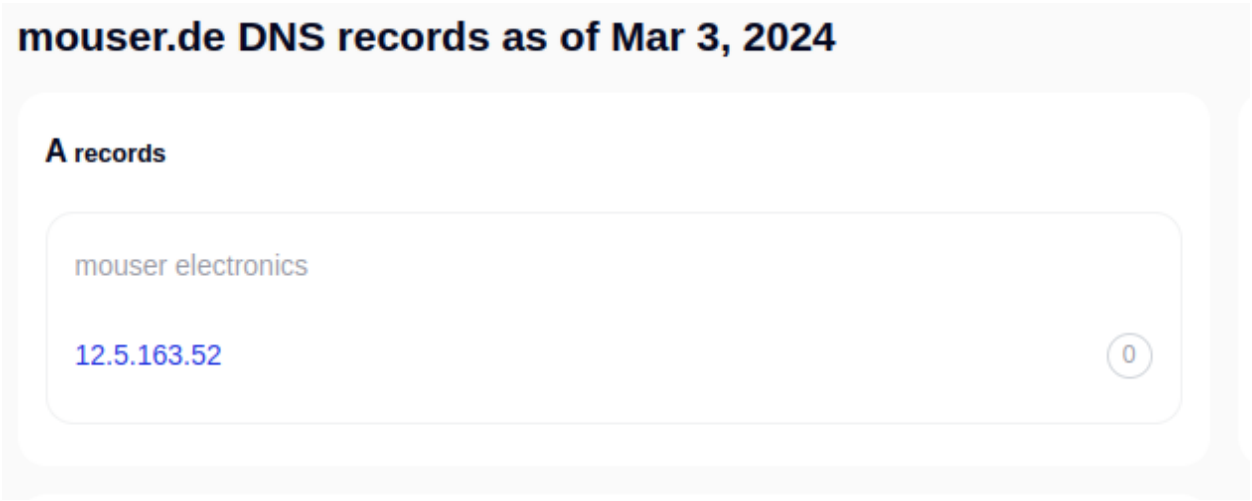

```
"Domain": "www.maybank2u.com.my",
"AsnsFound": [
    {
        "ID": "as9534",
        "Name": "binariang berhad",
        "IPs": [
            {
                "IP": "23.76.108.5",
                "HashSSLVerified": false,
                "CommonSSLCNNames": [
                    "*.test.edgekey.net",
                    "DigiCert TLS RSA SHA256 2020 CA1"
                ]
            }
        ]
    },
    {
        "ID": "as55836",
        "Name": "reliance jio infocomm limited",
        "IPs": [
            {
                "IP": "184.84.221.11",
                "HashSSLVerified": false,
                "CommonSSLCNNames": [
                    "a248.e.akamai.net",
                    "DigiCert TLS RSA SHA256 2020 CA1"
                ]
            }
        ]
    },
    {
        "ID": "as9299",
        "Name": "philippine long distance telephone company",
        "IPs": [
            {
                "IP": "23.202.148.12",
                "HashSSLVerified": false,
                "CommonSSLCNNames": [
                    "*.dl.playstation.net",
                    "DigiCert TLS RSA SHA256 2020 CA1"
```


]
}
]
},
{
"ID": "as18206",
"Name": "tm technology services sdn. bhd.",
"IPs": [
{
"IP": "202.162.17.68",
"HashSSLVerified": true,
"CommonSSLCNNames": [
"www.maybank2u.com.my",
"Sectigo RSA Extended Validation Secure Server CA",
"USERTrust RSA Certification Authority"
]
}
]
}
]
}

Mouser

Website: https://www.mouser.de
Security trails: https://securitytrails.com/domain/www.mouser.de/dns

https://securitytrails.com/domain/mouser.de/dns



https://securitytrails.com/domain/www.mouser.de/history/a?page=3

96.17.141.186	Akamai International B.V.	2020-05-16 (4 years)	2020-05-18 (4 years)	2 days
23.218.40.202	Rogers Communications Canada Inc.	2020-05-09 (4 years)	2020-05-11 (4 years)	2 days
23.218.40.202	Rogers Communications Canada Inc.	2020-05-06 (4 years)	2020-05-08 (4 years)	2 days
104.127.69.191	Akamai International B.V.	2020-05-04 (4 years)	2020-05-06 (4 years)	2 days

This is how the filters will look like this:

```
var Mouser = Input{
    URL:         utils.ParseURL("https://www.mouser.de"),
    TCPTimeout: time.Millisecond * 150,
    Keyworkds:  []string{"Distributor", "Deutschland"},
    BufferSize: 2048,
    Asn: asn.Asn{
        PrioritiesNames: []string{"mouser electronics", "Rogers Communications"},
        ForbiddenNames:  ForbidenASN,
    },
}
```

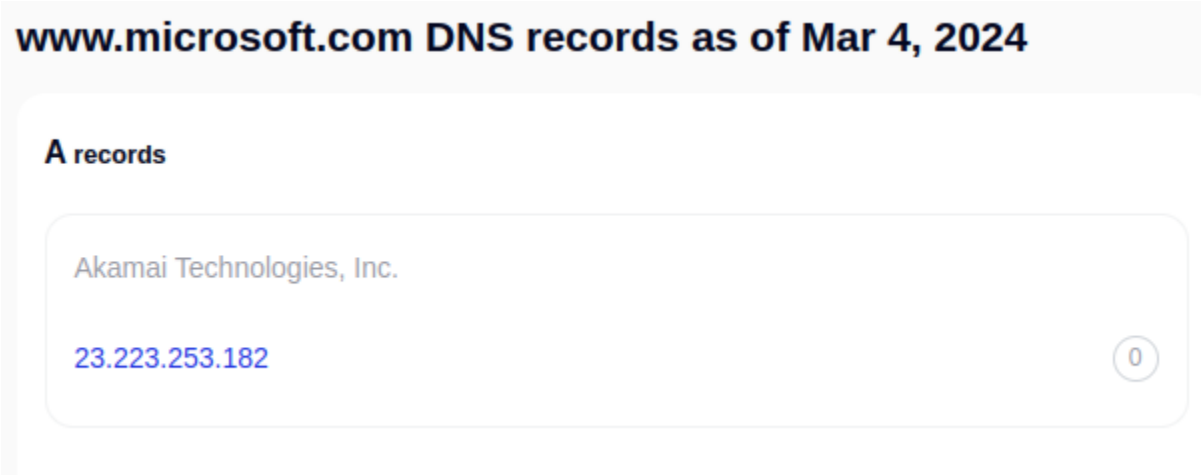
we get this results back:


```
{
    "Domain": "www.mouser.de",
    "AsnsFound": [
        {
            "ID": "as812",
            "Name": "rogers communications canada inc.",
            "IPs": [
                {
                    "IP": "184.29.112.4",
                    "HashSSLVerified": false,
                    "CommonSSLCNNames": [
                        "www.casio-intl.com",
                        "DigiCert TLS RSA SHA256 2020 CA1"
                    ]
                },
                {
                    "IP": "23.218.40.2",
                    "HashSSLVerified": false,
                    "CommonSSLCNNames": [
                        "*.test.edgekey.net",
                        "DigiCert TLS RSA SHA256 2020 CA1"
                    ]
                },
                {
                    "IP": "104.71.244.4",
                    "HashSSLVerified": false,
                    "CommonSSLCNNames": [
                        "*.test.edgekey.net",
```

```
        },
        {
            "ID": "as32368",
            "Name": "mouser electronics",
            "IPs": [
                {
                    "IP": "12.5.163.52",
                    "HashSSLVerified": false,
                    "CommonSSLCNNames": [
                        "www.mouser.com",
                        "GeoTrust TLS RSA CA G1",
                        "DigiCert Global Root G2"
                    ]
                }
            ]
        }
    ]
}
```

Microsoft

Website: https://www.microsoft.com
Security trails: https://securitytrails.com/domain/www.microsoft.com/dns



Let's see subdomains:
https://securitytrails.com/domain/www.microsoft.com/history/a

Domain	Rank	Hosting Provider
unistore.www.microsoft.com	3,690,586	Akamai Technologies, Inc.
dev.www.microsoft.com		-
controls-ppe.platform.account.www.microsoft.com		Microsoft Corporation
unistore-ppe.www.microsoft.com		Akamai Technologies, Inc.
unistorehome-ppe.www.microsoft.com		Microsoft Corporation
assets.www.microsoft.com		-
unistoreccservice-ppe.www.microsoft.com		-
controls.platform.account.www.microsoft.com		-
unistoreblends-int.www.microsoft.com		Microsoft Corporation
unistorehome-int.www.microsoft.com		Microsoft Corporation

Now lets see some of the DNS history:

https://securitytrails.com/domain/www.microsoft.com/history/a?page=5

23.217.196.148	Akamai Technologies, Inc.	2018-12-09 (5 years)	2018-12-16 (5 years)	7 days
23.217.196.148	Akamai Technologies, Inc.	2018-12-06 (5 years)	2018-12-08 (5 years)	2 days
204.237.214.199	GTT	2018-12-03 (5 years)	2018-12-05 (5 years)	2 days

Lets see what is the full name if the ASN for that mysterious IP:
https://iphub.info/?ip=204.237.214.199

Hostname/IP	204.237.214.199
ASN/ISP	AS3257 - GTT-BACKBONE
Country	United States
Type	Hosting, proxy or bad IP Report error/send suggestion

```
:ffff,AS200168,Telenor Norge AS,telenor.no
:ffff,AS200168,Telenor Norge AS,telenor.no
:ffff,AS6831,Net & Com s.r.l.,netecom.it
:ffff,AS33856,Turkiye Cumhuriyet Merkez Bankasi Anonim Sirketi,tcm
:ffff,AS3257,GTT Communications Inc.,gtt.net
:ffff,AS210156,REGIONAL SYSTEME INFORMATIQUE SARL,rsi-informatique
ff,AS198325,Yeni Telekom Internet Hizmetleri Ltd. Sirketi,yenitele
```

Let's dig in more:

community.advertising.microsoft.com	237,504	Microsoft Corporation
ajax.microsoft.com	240,874	Edgecast Inc.
inculture.microsoft.com	240,990	Cloudflare, Inc.

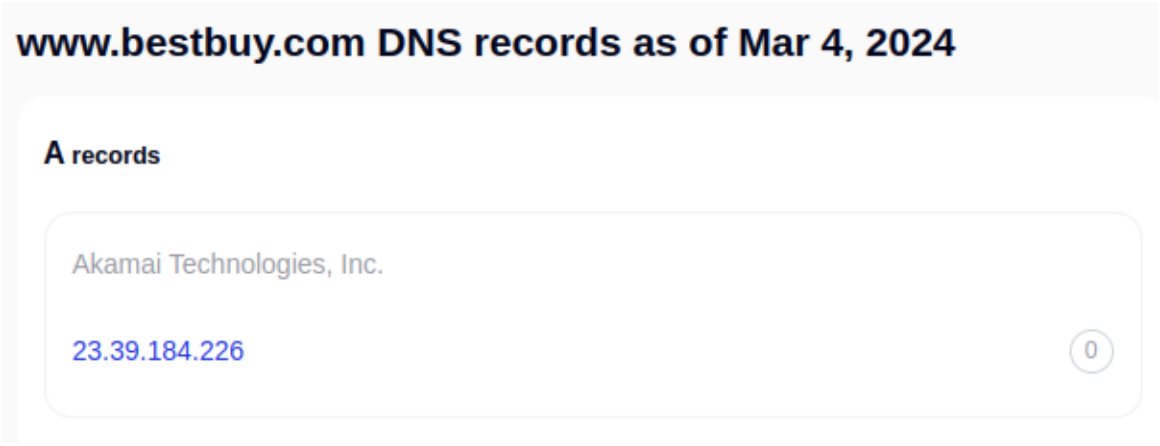
So filters will look like this:

```
var Microsoft = Input{
    URL:        utils.ParseURL("https://www.microsoft.com"),
    Keyworkds:  []string{"Microsoft", "Cloud", "Computers"},
    TCPTimeout: time.Millisecond * 150,
    BufferSize: 2048,
    Asn: asn.Asn{
        PrioritiesNames: []string{"GTT Communications", "Microsoft", "Edgecast"},
        ForbiddenNames:  ForbidenASN,
    },
}
var BestBuy = Input{
```

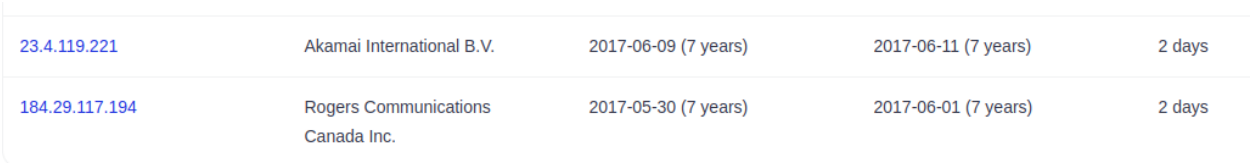
we get this results back:

BestBuy

Website: https://www.bestbuy.com
Security trails: https://securitytrails.com/domain/www.bestbuy.com/dns

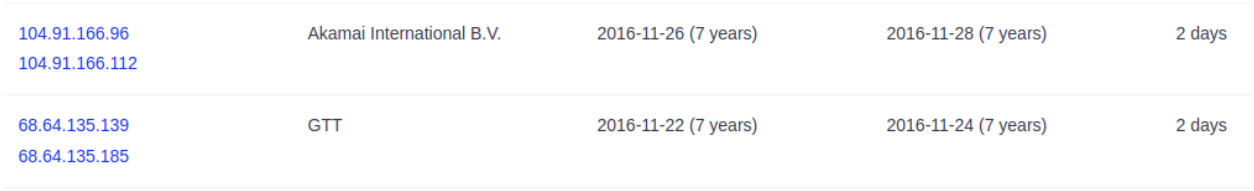


If we dig into the DNS history, a small mistakes pops up:
https://securitytrails.com/domain/www.bestbuy.com/history/a?page=5



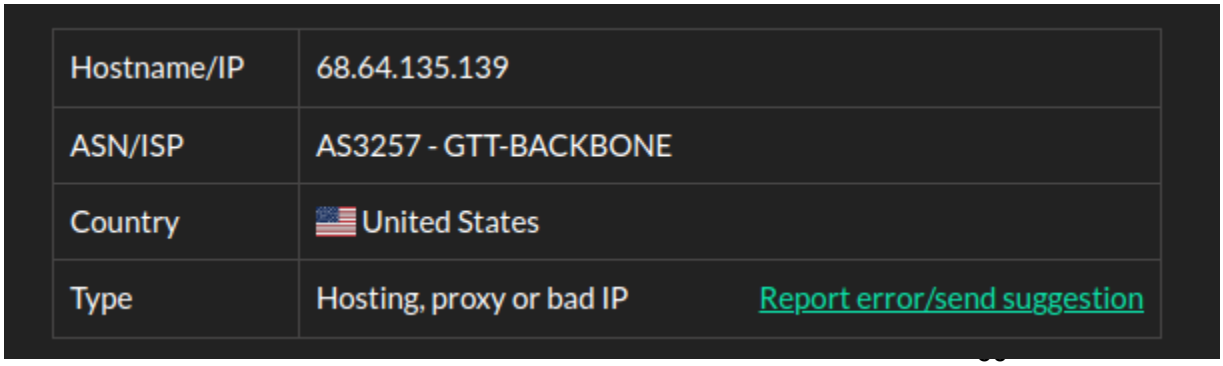
23.4.119.221	Akamai International B.V.	2017-06-09 (7 years)	2017-06-11 (7 years)	2 days
184.29.117.194	Rogers Communications Canada Inc.	2017-05-30 (7 years)	2017-06-01 (7 years)	2 days

They used Rogers Communications one time, we'll add this filter to the code



104.91.166.96 104.91.166.112	Akamai International B.V.	2016-11-26 (7 years)	2016-11-28 (7 years)	2 days
68.64.135.139 68.64.135.185	GTT	2016-11-22 (7 years)	2016-11-24 (7 years)	2 days

https://iphub.info/?ip=68.64.135.139



Hostname/IP	68.64.135.139	
ASN/ISP	AS3257 - GTT-BACKBONE	
Country	United States	
Type	Hosting, proxy or bad IP	Report error/send suggestion

```
.33.56.0,46.33.59.255,AS31593,"TV Company ""Black Sea"" Ltd",blacksea.net.ua
.33.60.0,46.33.63.255,AS48082,FOP Mikhailyuk Yuri Ivanovitch,odessa.tv
.33.64.0,46.33.95.255,AS3257,GTT Communications Inc.,gtt.net
.33.96.0,46.33.127.255,AS29208,"Quantcom, a.s.",quantcom.cz
.33.128.0,46.33.159.255,AS51561,ICUK Computing Services Limited,icuk.net
.33.160.0,46.33.191.255,AS50583,Index Education SAS,index-education.com
```

https://securitytrails.com/list/apex_domain/bestbuy.com

benefitsguide.bestbuy.com	6,417,545	Amazon.com, Inc.
my.bestbuy.com	6,614,657	Akamai Technologies, Inc.
advertising.bestbuy.com	6,902,277	Rackspace Hosting
products.semweb.bestbuy.com	9,113,125	-
influencernetwork.bestbuy.com	9,279,472	Akamai Technologies, Inc.
bbytagservices-bdc-origin.bestbuy.com		Best Buy Co., Inc.

The filters will look like this:

```
var BestBuy = Input{
    URL:         utils.ParseURL("https://www.bestbuy.com"),
    Keyworkds:   []string{"Best Buy", "Shop Now"},
    TCPTimeout: time.Millisecond * 150,
    BufferSize: 2048,
    Asn: asn.Asn{
        PrioritiesNames: []string{"Rogers Communicatio", "Rackspace", "Amazon",
        "Best Buy", "GTT Communications"},
        ForbiddenNames:  ForbidenASN,
    },
}
```

we get this results back:

Instancart

Website: https://www.instacart.com
Security trails: https://securitytrails.com/domain/www.instacart.com/history/a

Lets see the history:
https://securitytrails.com/domain/instacart.com/history/a

IP Addresses	Organization	First Seen	Last Seen	Duration Seen
104.18.37.67 172.64.150.189	Cloudflare, Inc.	2023-08-31 (6 months)	2024-03-04 (today)	6 months
104.18.16.6 104.18.17.6	Cloudflare, Inc.	2021-09-20 (2 years)	2023-08-31 (6 months)	2 years
3.209.64.57 3.234.202.147 35.171.122.66 52.0.198.133 52.206.82.202 54.157.99.141 54.166.214.4 100.26.93.175	Amazon.com, Inc.	2021-08-30 (3 years)	2021-09-17 (2 years)	18 days

What about subdomains:
https://securitytrails.com/list/apex_domain/instacart.com

prd-tfm-platform-preview-api.enterprise.instacart.com	Google LLC
publix.mailin1-com.edge.instacart.com	-
stop-shop-express.pbis-cf.instacart.com	Cloudflare, Inc.
uat-gar-platform-merch-web.enterprise.instacart.com	Google LLC
uat-top-platform-pricing-web.enterprise.instacart.com	Google LLC
uat.lnb.cf.enterprise.instacart.com	Cloudflare, Inc.

So it uses Google and amazon, so filters will look like this:

```
var InstantCart = Input{
    URL:         utils.ParseURL("https://www.instacart.com"),
    Keyworkds:   []string{"delivery", "pickup", "grocers"},
    TCPTimeout: time.Millisecond * 150,
    BufferSize: 2048,
    Asn: asn.Asn{
        PrioritiesNames: []string{"Amazon", "google"},
        ForbiddenNames:  ForbidenASN,
    },
}
```

we get this results back:


```
{
    "Domain": "www.instacart.com",
    "AsnsFound": [
        {
            "ID": "as16509",
            "Name": "amazon.com, inc.",
            "IPs": [
                {
                    "IP": "13.226.136.2",
                    "HashSSLVerified": true,
                    "CommonSSLCNNames": [
                        "*.observian.com",
                        "Amazon RSA 2048 M03",
                        "Amazon Root CA 1",
                        "Starfield Services Root Certificate Authority - G2"
                    ]
                }
            ]
        },
        {
            "ID": "as14618",
            "Name": "amazon.com, inc.",
            "IPs": [
                {
                    "IP": "52.200.1.85",
                    "HashSSLVerified": true,
                    "CommonSSLCNNames": [
                        "instacart.com",
                        "Amazon RSA 2048 M01",
                        "Amazon Root CA 1",
                        "Starfield Services Root Certificate Authority - G2"
                    ]
                }
            ]
        }
    ]
}
```

Cars

Website: https://www.cars.com
Security trails: https://securitytrails.com/domain/www.cars.com/dns

https://securitytrails.com/domain/www.cars.com/history/a?page=5

23.196.120.188	Akamai Technologies, Inc.	2017-05-30 (7 years)	2017-06-01 (7 years)	2 days
74.119.98.141	Classified Ventures	2016-01-14 (8 years)	2016-01-20 (8 years)	6 days
74.119.98.141	Classified Ventures	2016-01-06 (8 years)	2016-01-13 (8 years)	7 days
74.119.98.141	Classified Ventures	2015-11-27 (8 years)	2016-01-05 (8 years)	1 month

https://securitytrails.com/domain/cars.com/dns

So Amazon and Classified Venture will be part of the filters:

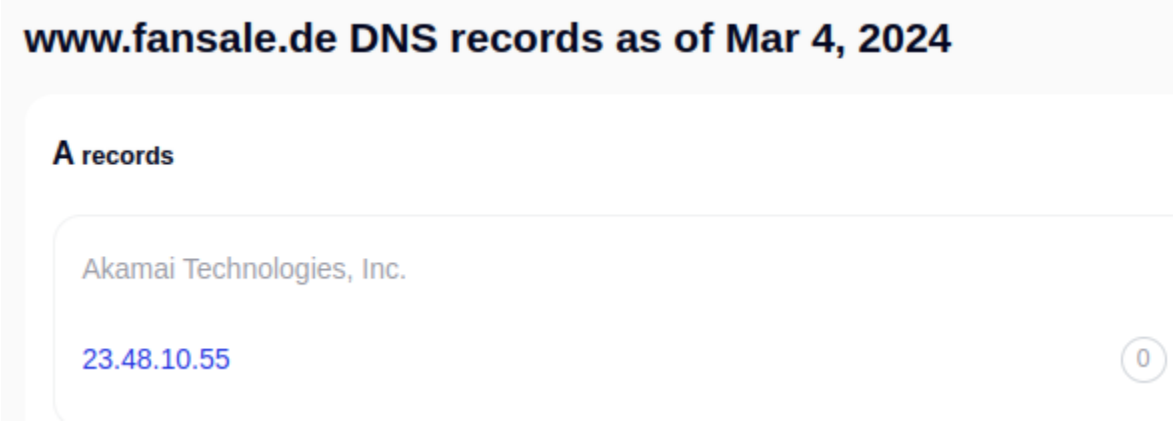
```
var Cars = Input{
    URL:         utils.ParseURL("https://www.cars.com"),
    Keyworkds:   []string{"New Cars", "Dealers"},
    TCPTimeout: time.Millisecond * 150,
    BufferSize: 2048,
    Asn: asn.Asn{
        PrioritiesNames: []string{"Classified Venture", "Amazon"},
        ForbiddenNames:  ForbidenASN,
    },
}
```

we get this results back:


```
{
    "Domain": "www.cars.com",
    "AsnsFound": [
        {
            "ID": "as16509",
            "Name": "amazon.com, inc.",
            "IPs": [
                {
                    "IP": "13.32.47.2",
                    "HashSSLVerified": true,
                    "CommonSSLCNNames": [
                        "*.dhanrajw.com",
                        "Amazon RSA 2048 M01",
                        "Amazon Root CA 1",
                        "Starfield Services Root Certificate Authority - G2"
                    ]
                },
                {
                    "IP": "13.32.18.2",
                    "HashSSLVerified": true,
                    "CommonSSLCNNames": [
                        "cloudfront.quizscore.com",
                        "Thawte TLS RSA CA G1"
                    ]
                },
                {
                    "IP": "13.33.158.2",
                    "HashSSLVerified": true,
                    "CommonSSLCNNames": [
                        "*.cloudfront.net",
                        "Amazon RSA 2048 M01",
                        "Amazon Root CA 1",
                        "Starfield Services Root Certificate Authority - G2"
                    ]
                }
```

Fansale

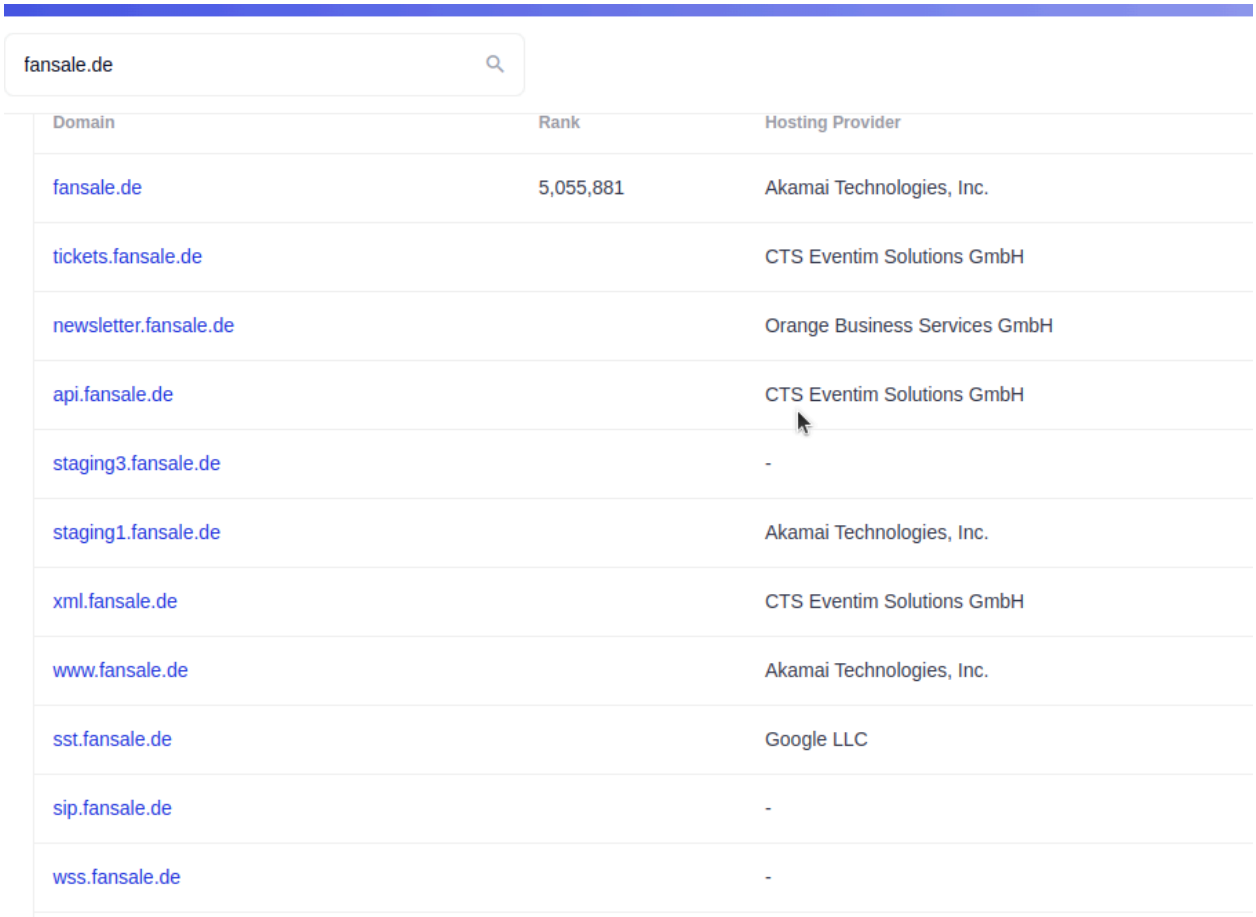
Website: https://www.fansale.de/fansale/
Security trails: https://securitytrails.com/domain/www.fansale.de/dns



Lets see this DNS history:
https://securitytrails.com/domain/www.fansale.de/history/a?page=5

23.208.68.212	Akamai Technologies, Inc.	2017-08-30 (7 years)	2017-09-01 (7 years)	2 days
23.209.114.71	Mobile Telecommunication Company Saudi Arabia Joint-Stock company	2017-08-18 (7 years)	2017-08-20 (7 years)	2 days

Security trails link: https://securitytrails.com/list/apex_domain/fansale.de



And Bingo! We have a winner here, unfortunately the api domain is not protected, api.fansale.de , and we can see the ASN name they use is CTS Eventim Solutions GmbH.
So the solution to this problem is instead of checking for all possible IPs in the history, we just check the ones with ASN name CTS Eventim Solutions GmbH

```
var Fansale = Input{
    URL:         utils.ParseURL("https://www.fansale.de/fansale/"),
    Keyworkds:   []string{"fanSALE", "Tickets"},
    TCPTimeout: time.Millisecond * 150,
    BufferSize: 2048,
    Asn: asn.Asn{
        PrioritiesNames: []string{"Mobile Telecommunication Company Saudi
        Arabia", "CTS Eventim"},
        ForbiddenNames:  ForbidenASN,
    },
}
```

we get this results back:

esultsbefore > Fansale > {} fansale.json > ...


```
{
    "Domain": "www.fansale.de",
    "AsnsFound": [
        {
            "ID": "as204253",
            "Name": "cts eventim solutions gmbh",
            "IPs": [
                {
                    "IP": "185.109.196.51",
                    "HashSSLVerified": false,
                    "CommonSSLCNNames": [
                        "*.eventim.de",
                        "Thawte TLS RSA CA G1"
                    ]
                },
                {
                    "IP": "185.109.196.68",
                    "HashSSLVerified": false,
                    "CommonSSLCNNames": [
                        "*.eventim.de",
                        "Thawte TLS RSA CA G1"
                    ]
                }
            ]
        }
    ]
}
```

Verify Gov

Website: https://verify.sos.ga.gov/verification/Search.aspx
Security trails: https://securitytrails.com/domain/verify.sos.ga.gov/dns

Let's see the DNS history:

104.18.10.78 104.18.11.78	Cloudflare, Inc.	2021-05-03 (3 years)	2022-04-16 (2 years)	12 months
173.245.101.119	Quality Technology Services, LLC	2020-10-08 (3 years)	2021-05-03 (3 years)	7 months
50.232.155.145	Comcast Cable Communications, LLC	2019-12-05 (4 years)	2020-10-08 (3 years)	10 months

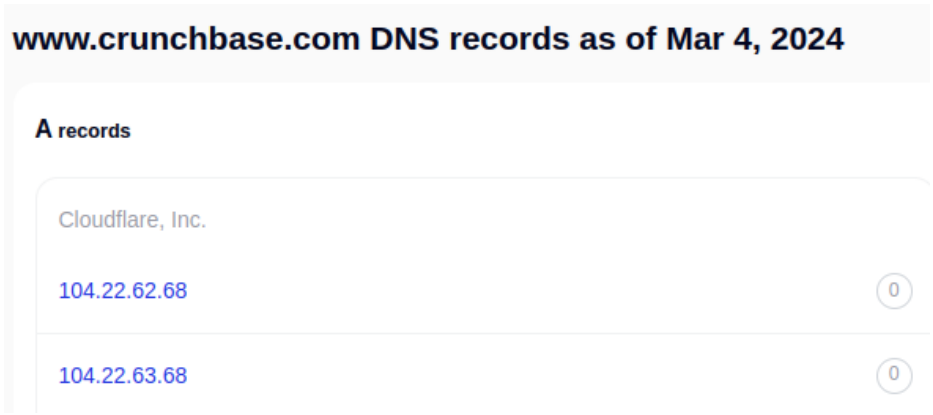
This one is super easy, Comcast Cable and Quality Technology Services filters
Filters will look like this:

```
}
var VerifySos = Input{
    URL:         utils.ParseURL("https://verify.sos.ga.gov/verification/Search.aspx"),
    Keyworkds:   []string{"Verification", "elicense2000"},
    TCPTimeout: time.Millisecond * 150,
    BufferSize: 2048,
    Asn: asn.Asn{
        PrioritiesNames: []string{"Quality Technology Services", "Comcast Cable"},
        ForbiddenNames:  ForbidenASN,
    },
}
```

The results looks like this:

CrunchBase

Website: https://www.crunchbase.com
Security trails: https://securitytrails.com/domain/www.crunchbase.com/dns



Let's investigate: https://securitytrails.com/domain/www.crunchbase.com/history/a

IP Addresses	Organization	First Seen	Last Seen	Duration Seen
104.22.62.68 104.22.63.68 172.67.41.8	Cloudflare, Inc.	2022-02-08 (2 years)	2024-03-04 (today)	2 years
34.214.87.149 35.83.69.153 52.39.139.103	Amazon.com, Inc.	2022-02-06 (2 years)	2022-02-08 (2 years)	2 days
35.164.170.246 52.24.238.108 52.40.63.248	Amazon.com, Inc.	2022-01-25 (2 years)	2022-02-04 (2 years)	10 days

https://securitytrails.com/list/apex_domain/crunchbase.com

Domain	Rank	Hosting Provider
crunchbase.com	505	Cloudflare, Inc.
news.crunchbase.com	11,340	Google LLC
about.crunchbase.com	24,969	Google LLC
data.crunchbase.com	130,438	Cloudflare, Inc.
static.crunchbase.com	1,153,217	Amazon.com, Inc.
support.crunchbase.com	1,252,231	Cloudflare, Inc.

We get Google and Amazon, this is how the filter will look like:

```
var crunchbase = Input{
    URL:         utils.ParseURL("https://www.crunchbase.com"),
    Keyworkds:   []string{"Crunchbase: Discover innovative"},
    BufferSize: 2048,
    Asn: asn.Asn{
        PrioritiesNames: []string{"amazon", "Google"},
        ForbiddenNames:  ForbidenASN,
    },
}
```

we get this results back:


```
{
    "Domain": "www.crunchbase.com",
    "AsnsFound": [
        {
            "ID": "as16509",
            "Name": "amazon.com, inc.",
            "IPs": [
                {
                    "IP": "52.24.3.19",
                    "HashSSLVerified": true,
                    "CommonSSLCNNames": [
                        "crunchbase.com",
                        "Amazon RSA 2048 M01",
                        "Amazon Root CA 1",
                        "Starfield Services Root Certificate Authority - G2"
                    ]
                }
            ]
        },
        {
            "ID": "as396982",
            "Name": "google llc",
            "IPs": [
                {
                    "IP": "34.168.222.209",
                    "HashSSLVerified": false,
                    "CommonSSLCNNames": [
                        "*.sta.test.gemalto.com",
                        "dockerca"
                    ]
                }
            ]
        }
    ]
}
```

Secure State

Website: https://secure.state.co.nz/car
Security trails: https://securitytrails.com/domain/www.mouser.de/dns

This one was sad because they did everything good except for one little mistake at the the beginning of the website history

https://securitytrails.com/domain/secure.state.co.nz/history/a?page=3

23.50.51.73 23.50.51.121	Akamai International B.V.	2020-06-09 (4 years)	2020-06-11 (4 years)	2 days
23.217.129.129 23.217.129.154	Akamai International B.V.	2020-06-05 (4 years)	2020-06-08 (4 years)	3 days
202.50.1.200	IAG New Zealand	2019-01-03 (5 years)	2020-06-05 (4 years)	1 year

So the filter looks like this:

```
var Secure_state = Input{
    URL:        utils.ParseURL("https://secure.state.co.nz/car"),
    Keyworkds:  []string{"State Insurance", "secure.state.co.nz/car/favicon.ico"},
    TCPTimeout: time.Millisecond * 150,
    BufferSize: 2048,
    Asn: asn.Asn{
        PrioritiesNames: []string{"IAG New Zealand"},
        ForbiddenNames:  ForbidenASN,
    },
}
```

we get this results back:


```
{
    "Domain": "verify.sos.ga.gov",
    "AsnsFound": [
        {
            "ID": "as29748",
            "Name": "quality technology services, llc",
            "IPs": [
                {
                    "IP": "173.245.101.120",
                    "HashSSLVerified": true,
                    "CommonSSLCNNames": [
                        "*.sos.ga.gov",
                        "DigiCert Global G2 TLS RSA SHA256 2020 CA1"
                    ]
                }
            ]
        }
    ]
}
```

Starngage

Website: https://starngage.com/plus/en-us
Security trails: https://securitytrails.com/domain/starngage.com/dns

And unfortunately the api domains are not protected so we can see what ASN they are using

https://securitytrails.com/list/apex_domain/starngage.com

Domain	Rank	Hosting Provider
starngage.com	165,763	Cloudflare, Inc.
a.starngage.com	2,062,513	Cloudflare, Inc.
www.starngage.com		Cloudflare, Inc.
starngage-edm-prod-assets.starngage.com		Cloudflare, Inc.
email.starngage.com		-
cms.starngage.com		Cloudflare, Inc.
api.staging.starngage.com		Amazon.com, Inc.
api.plus.starngage.com		Cloudflare, Inc.
graph.starngage.com		Cloudflare, Inc.
a.staging.starngage.com		Amazon.com, Inc.

The input:

```
}
var starngage = Input{
    URL:         utils.ParseURL("https://starngage.com/plus/en-us"),
    Keyworkds:   []string{"Influencer", "Agencies"},
    TCPTimeout: time.Millisecond * 250,
    BufferSize: 2048,
    Asn: asn.Asn{
        PrioritiesNames: []string{"Amazon", "GoDaddy"},
        ForbiddenNames:  ForbidenASN,
    },
}
var Fansale = Input{
```

The result:

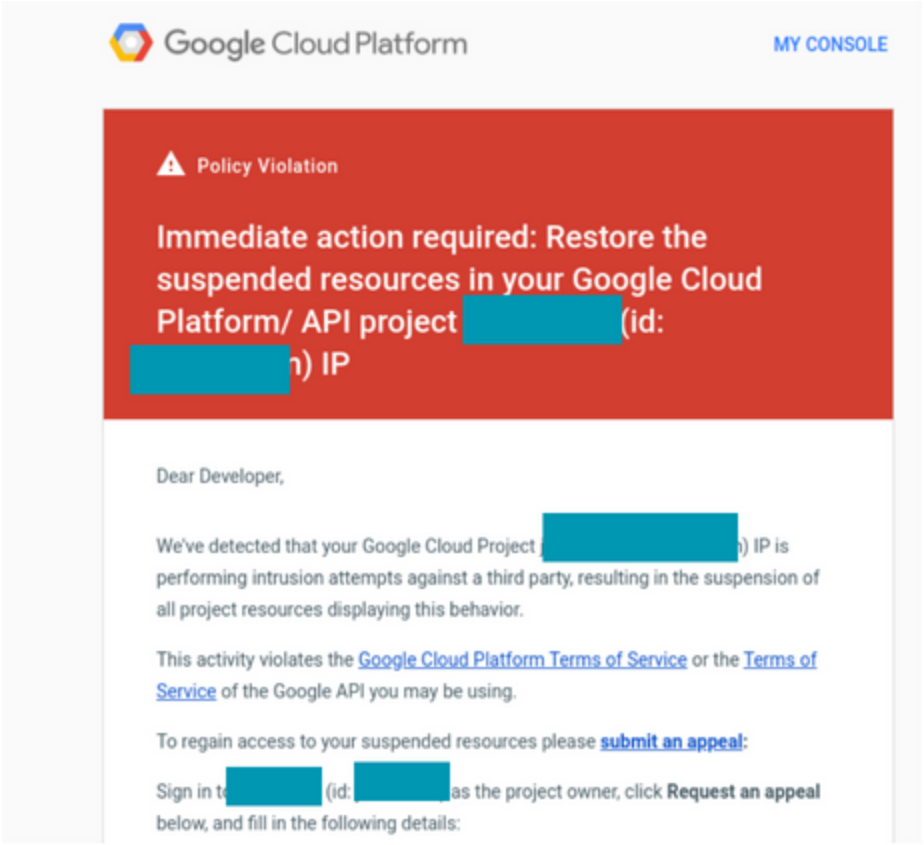

```
{
    "Domain": "starngage.com",
    "AsnsFound": [
        {
            "ID": "as16509",
            "Name": "amazon.com, inc.",
            "IPs": [
                {
                    "IP": "18.218.52.124",
                    "HashSSLVerified": false,
                    "CommonSSLCNNames": [
                        "ceong.com.br",
                        "R3",
                        "ISRG Root X1"
                    ]
                },
                {
                    "IP": "54.185.52.204",
                    "HashSSLVerified": false,
                    "CommonSSLCNNames": [
                        "api.us.apolloshield.com",
                        "R3",
                        "ISRG Root X1"
                    ]
                },
                {
                    "IP": "34.215.70.243",
                    "HashSSLVerified": false,
                    "CommonSSLCNNames": [
                        "*.xenial.com",
                        "Entrust Certification Authority - L1K",
                        "Entrust Root Certification Authority - G2"
                    ]
                },
                {
                    "IP": "13.43.65.192",
                    "HashSSLVerified": false
```

Precautions

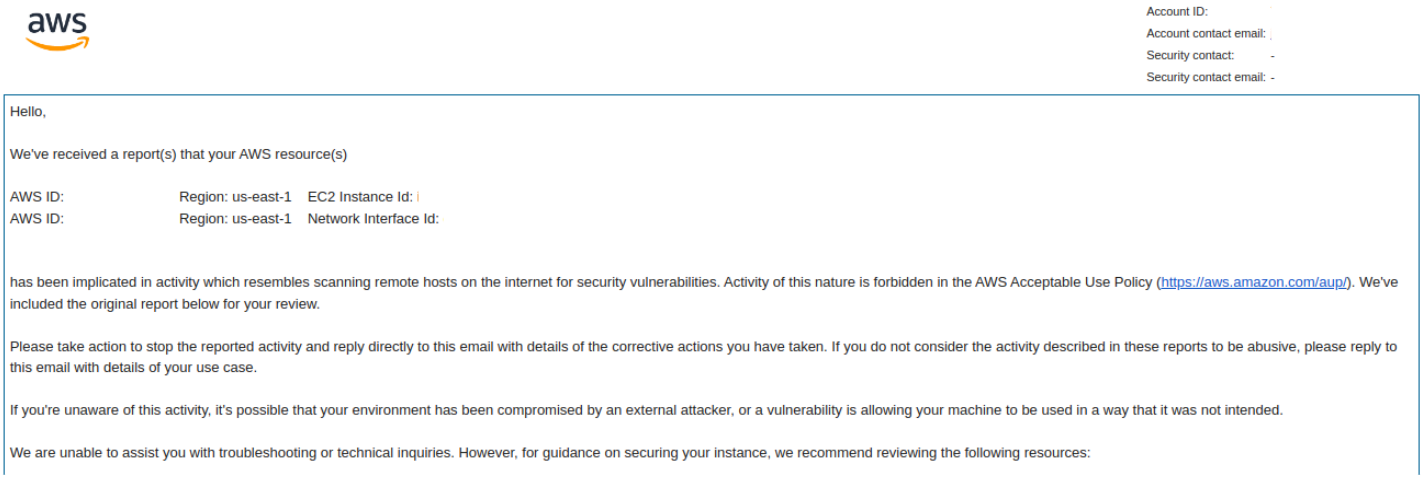
- Some ASN providers like **Hetzner** analyzes the traffic on their IPs, if they find something suspicious on the traffic, they will alert to however is the owner of the origin IP, what this means is that if you use the code locally or on the cloud like Google Cloud, aws, Azure .. etc, **Hetzner** will detect this and inform these ASN and these ASN providers will block your account/project, I got my project blocked on Google and AWS, so be aware., if you want to use the code, use a VPN, it will be slower but you will avoid being labeled by your ASN. This could be fixed by adding proxy support but I probably won't do it because that defeats the purpose of this research, which is to prevent websites from making mistakes like this and prevent future hackers from exploiting this vulnerability.

- Be aware that if you find an IP, this WON'T guarantee that's the original server, it could be a scammer website, so don't enter login information because you can might get hacked, to mitigate this you can first check if the website has the SSL certificate for that domain, which still won't guarantee it's the legit website but you have more chance it's not a scammer website

Ban from Google:



Ban from AWS:



Multiple IPs on domains

Why is there more than one IP on some domains?
Multiple reasons:

1) Load balancer, they probably are using it to distribute the load across multiple servers

2) Scammers website, there is exits the possibility the server is impersonating the website, but it will be super weird because the hacker will need somehow access to the victim's network environment

1. Somehow got access to the victim's pc and changed the DNS host configuration to redirect all the traffic from that domain to the hacker's server IP.
2. Hacker managed to make a reverse proxy, probably by hacking a router and then offering free wifi on public spaces, he can make a reverse a reverse proxy, analyze the traffic and save victim's information

How can I Verify it's the legitimate server:

1. Check the ssl certificates from that IP

Let's take as example https://secure.state.co.nz , we can make sure the IP we found it's actually the real server and not load balancer or reverse proxy, the IP found was 202.50.1.200
So lets check if that IP contains the SSL certificates, we will use the command:

```
$ openssl s_client -servername [domain] -connect [IP]:443 -showcerts
```

So we can run the command:

```
$ openssl s_client -servername secure.state.co.nz -connect 202.50.1.200:443
-showcerts
```

And if we check closely, the ssl certificate contains information about this domain on that IP, so we can be 99% sure that this the target server, the 1% could be if a hacker somehow got access to the SSL certificate on the original server VM, copied this certificate on another VM and its impersonating the website.

However if the website doesn't have a ssl certificate it doesn't mean it's not a legit server, it could be using unencrypted HTTP to connect to the WAF.

Suggestions

If you goal is security:

1. Setup the WAF if your hosting has it, some popular have it already:
    - Google Cloud: Google Cloud armor
    - AWS: AWS WAF
    - Azure: Azure WAF
    - IBM: IBM WAF

2. For the hosting platforms: you can have some bait IPs, these IPs won't be used for anything except analyzing what traffic arrives there, if any traffic arrives, apply warnings similar to Hetzner which I think is what they used to detect this attack.

3. Config your server to allow incoming traffic only from IP from your WAF, so let's say you are using:
    - Cloudflare, IP filter cloudflare
    - Imperva or akamai, don't have docs about this you can check the ASN IP for that companies and whitelisted them same as cloudflare documentation
4. Hire the right team to make the setup of your server, the one in charge of your server configuration is someone specialized on system administrator, so if you let a developer do that configuration, he will be able to setup the server but could fail on a single detail on the server configuration which can led to attacks on the server; vise versa is true too don't let a system admin to develop the app, it could lead to vulnerabilities on the backend/frontend
5. Hire security team experts to analyze your system, I'm not sure if this is a known bug but I've never seen it before, that's why it's important to understand every possible detail so system admin can take actions to prevent attacks.
6. Other recommendations can be given by a security expert, I'm not.

If you goal is to avoid your page been scraped by bots

Unfortunately, you won't be able to stop them. I've been working on creating bots for about 6 years and all of them have been successful, even if you think you will apply some highly advanced machine learning algorithm to track user movements, user interaction, GPU configuration, user agent.. Etc , still there will be a way to bypass it, however you can make the bot creation more painful.

You can:

- In order for a user to search on the page, that user must be logged in, so then you can apply the user's limitations on the backend.

- Create an api and charge for using it, this way you can make some revenue and the demand for web scraper developers like myself will decrease.

- Delay and make the process of creating this bots painful and more expensive, if you like:
    1) Google recaptcha
    2) Cloudflare turnstile
    3) Hcaptcha: hcaptcha

    Even applying this to your website, web scraping developers like myself will be able to bypass those services, but the bots will be slow and will need to pay third party services in order to bypass it.
    However, using this will annoy your users too.

The Code

The code is made in Golang because there is no other language that can solve this problem as elegantly and efficiently as Golang.

The code is not made to be a library, you will need to download it and change as many variables as you want.

Installing:

You need to install Golang https://go.dev/dl

The setup:

You need to create a Input{} variable and fill the values for that domain
For example:

```
var Secure_state = Input{
   URL:         utils.ParseURL("https://secure.state.co.nz/car"),
   Keyworkds:   []string{"State Insurance",
"secure.state.co.nz/car/favicon.ico"},
   TCPTimeout: time.Millisecond * 150,
   BufferSize: 2048,
   Asn: asn.Asn{
       PrioritiesNames: []string{"IAG New Zealand"},
       ForbiddenNames:  ForbidenASN,
   },
}
```

Keywords: if what keywords you find on the body, try these keywords been at the start of the body
URL: the url you are trying to get the IP from, the code will extrat the host and add the path to the request
And Asn.Priorities: the ASN names you want to check on
Buffer Size: the size of the buffer, you can adjust this but its recommend to be low not to waste bandwidth

You can use the default code, which will print the progress each 1 minute, it will save all the bodies from requests for the IPs found, save the IPs found as soon as it gets it, and at the end the save the result on a json file

```
package main

func main() {
   input := Nike
   input.stopOnEachAsnFound()
   //input.stopOnIPCertificateFound()
}
```

Running the code

you just run the command:

```
$ go run .
```

Highlightings on the code:

- The code will span **n** numbers of workers to process all this concurrently https://www.youtube.com/watch?v=oV9rvDllKEg

- The code will send progress updates each period of time through a channel
- The code will send the IP found as soon as it finds it
- The code will make a http request, verify if there is any redirection, if there is a redirection it will make an HTTPs request.

One of the optimizations made on the code is for each ASN IP range to search, the code will search one by one but across all IP range, not individually, this explanations sounds like shit but I don't know how to explain rather with examples, let's say the IP is just a number and we will check this IP range:

1-20
50-80
100-200

The code will send the IPs to the workers in this order:

1. 1
2. 50
3. 100
4. 2
5. 51
6. 101
7. 3
8. 52
9. 102

Etc..
Previous code was sending it like this:

1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10
11. 11

Etc..

But it turns out i wasn't optimal if you want to find an IP as fast as possible, the total amount for searching it will be almost the same, but this optimization will find the IPs faster, so if with the first code we get to find it on the iteration 1000, on the second code we probably will find it on iteration 8000.
Check docs about column major approach

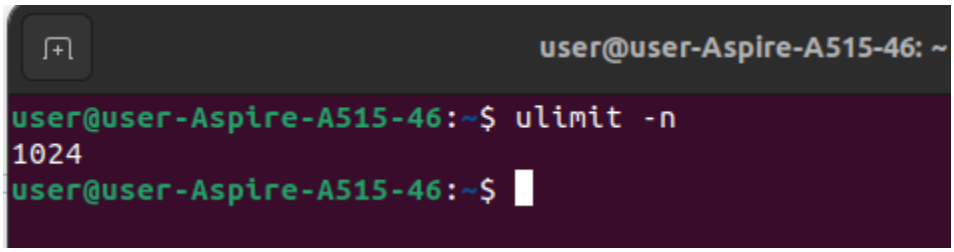
The hardware and software:

You need to verify how many HTTP requests your network can handle at the same time, you will need to adjust the code for your setup, if you just put a high workers number, you won't find anything, this is because there is not a way to verify if the error on a request was because your network is saturated or there was actually a timeout issue with the server.
You can try finding the IPs from the examples, if you can't find any IP, then just make the worker number smaller

You need to adjust the system to handle multiple request, your OS can handle few request by default, if you use linux try this command:

```
$ ulimit  -n
```

You will probably find something like this:



Basically each request needs to open a socket, and each socket uses the file descriptor, so you will need to increase this number.

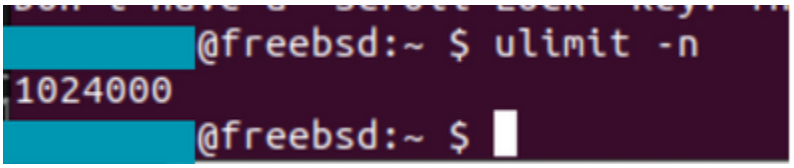
FreeBSD

1) Replace the file /etc/security/limits.conf

With:

```
*               hard    nofile          1048576
*               soft    nofile          1048576
```

Then you will need to logout and login again to the vm to apply the changes, then you can verify again and you will get the update:



2) Replace the file /etc/sysctl.conf

With:

```
net.inet.tcp.msl=2000
net.inet.tcp.finwait2_timeout=15000
net.inet.ip.portrange.first=1024
net.inet.ip.portrange.last=65535
kern.maxfiles=2048000
kern.maxfilesperproc=1024000
```

Then run:

```
$ sysctl -f /etc/sysctl.conf
```

Debian

1) Replace the file /etc/security/limits.conf

With:

```
*                hard    nofile          1048576
*                soft    nofile          1048576
```

2) Replace the file /etc/sysctl.conf

With:

```
net.ipv4.ip_local_port_range = 1024 65535
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_fin_timeout = 15
net.ipv4.tcp_keepalive_time = 300
net.ipv4.tcp_max_syn_backlog = 4096
net.ipv4.tcp_max_tw_buckets = 400000
net.core.somaxconn = 4096
net.core.netdev_max_backlog = 250000
net.ipv4.tcp_timestamps = 0
net.ipv4.neigh.default.gc_thresh1 = 2048
net.ipv4.neigh.default.gc_thresh2 = 4096
net.ipv4.neigh.default.gc_thresh3 = 8192
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.all.send_redirects = 0
net.core.wmem_max = 16777216
net.core.rmem_max = 16777216
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216
net.netfilter.nf_conntrack_max = 131072
net.netfilter.nf_conntrack_tcp_timeout_established = 60
```

Then run:

```
$ sysctl -f /etc/sysctl.conf
```