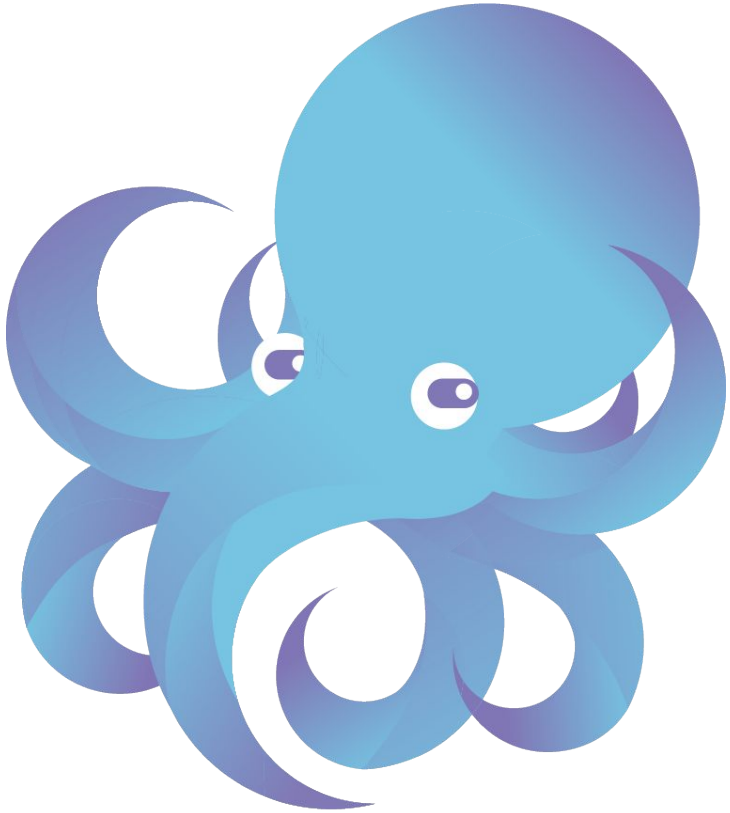
CLOUDFLARE



# Cloudflare and RPKI at scale

Louis Poinsignon

Martin J. Levy

# Introduction

Louis Poinsignon:

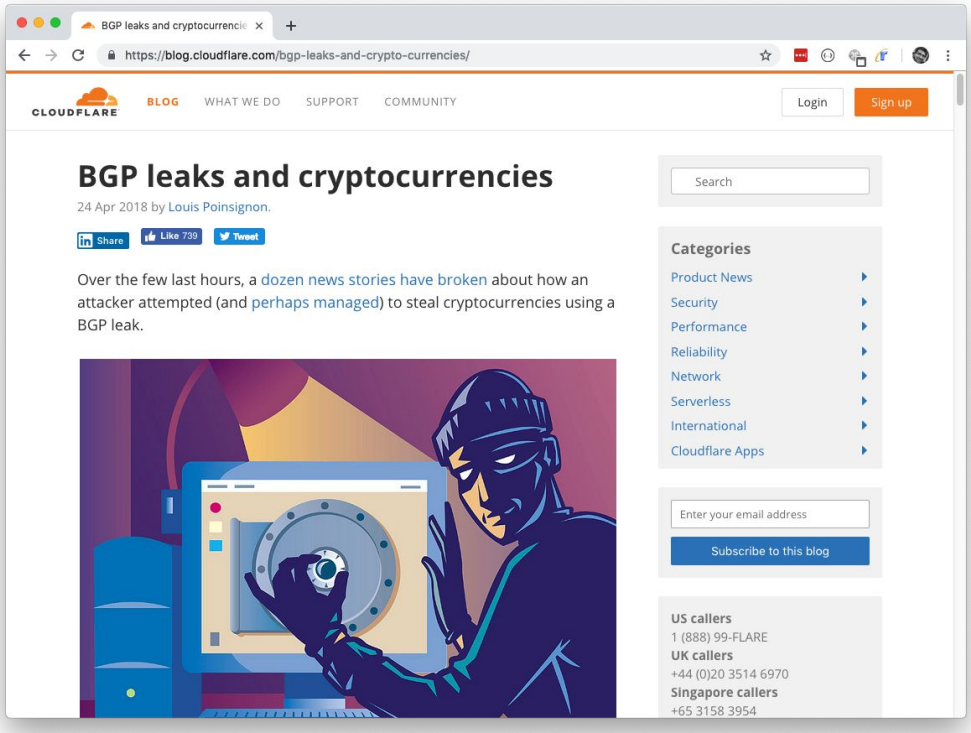
- Network Engineer at Cloudflare in San Francisco
- Open-source projects including flows and RPKI
- Network data collection (BGP, flows, peering-portal)


Featured in RIPE's video!
https://www.youtube.com/watch?v=Y9vbbxr-GbI

CLOUDFLARE

https://blog.cloudflare.com/rpki-details/
https://blog.cloudflare.com/rpki/

# How did it start?




CLOUDFLARE

https://blog.cloudflare.com/bgp-leaks-and-crypto-currencies/

# The Initial Story

Authoritative DNS route hijack in April 2018

DNS route announced via peering session (in Chicago)

This affected our network, hence our DNS Resolver

What should we do?

CLOUDFLARE

https://blog.cloudflare.com/bgp-leaks-and-crypto-currencies/

# The Initial Story

At the time...

- 150+ PoPs
- 26,000 BGP sessions
- IP space from five RIRs

Just the RIPE Validator [1]

How to distribute a prefix list efficiently?

CLOUDFLARE

[1] Cloudflare is very grateful for the RIPE Validator s/w

# The Initial Story

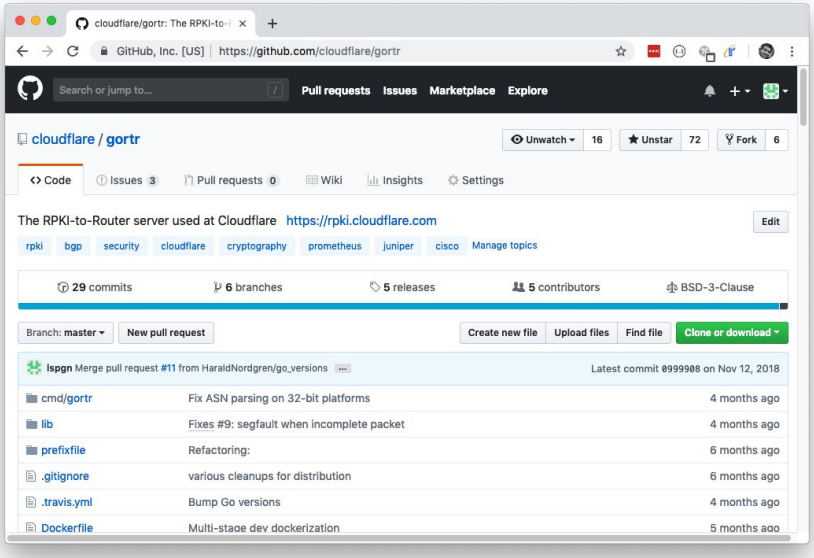
July: started deploying internally GoRTR.

August: open-source release.

`https://github.com/cloudflare/gortr`

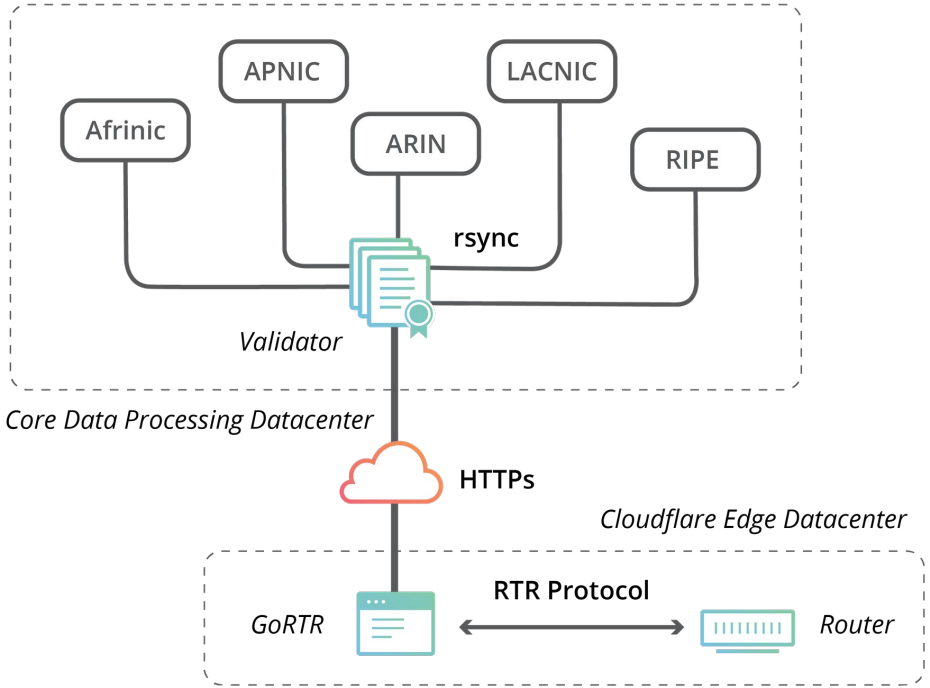
September → December:

- Turn up RTR sessions
- Signing prefixes`




https://github.com/cloudflare/gortr

# Diagram




CLOUDFLARE

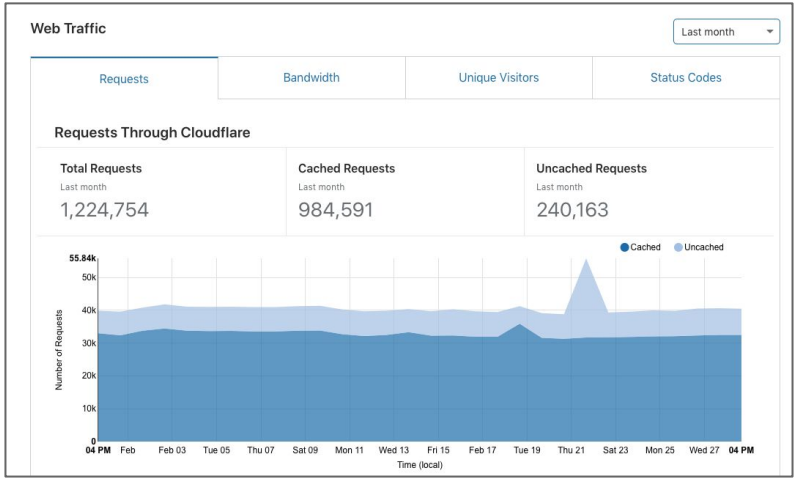
# Behind the scene (until January 2019)

**RIPE Validator** providing list of prefixes

Running in a Mesos cluster

With a cronjob:

- Fetching the data
- Filtering (remove > /24 and > /48 and duplicates)
- Signing it
- Making it available to our edge




`https://rpki.cloudflare.com/rpki.json` was born.

CLOUDFLARE

# Effects


https://www.flickr.com/photos/thure/6287816628/

The question everyone asked us:

**How much traffic was affected?**

Many invalids. Little traffic in practice

(we had a default or valid less specific)

Except in one place: Few gigabits per seconds displaced due to geographical more specific

CLOUDFLARE

# Accounting

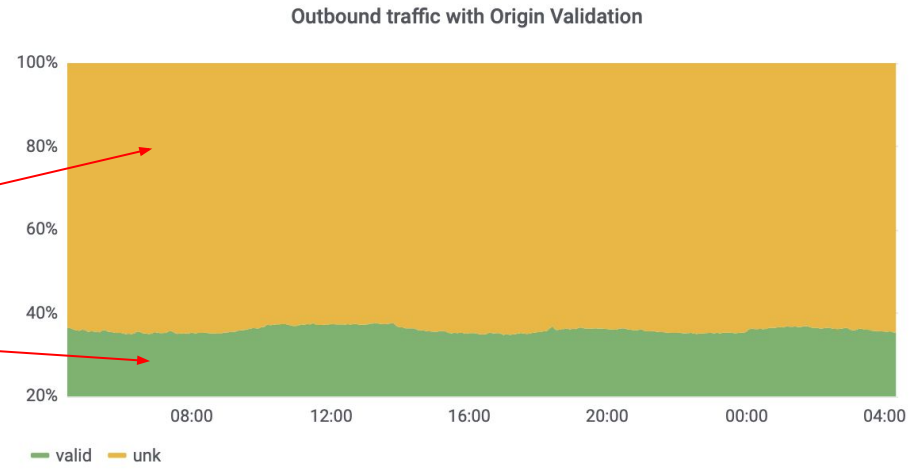
Using flows, we see **at least 30%** of our traffic being **valid**

Very **little/none invalid**

We use **GoFlow** for accounting. (Other tools compatible with flows: pmacct, Kentik, etc)

Traffic with a no ROA

Traffic with a valid ROA




CLOUDFLARE

# Signing the routes

# Signing the routes

Cloudflare has IP space from five RIRs

(*no space from twnic/jpnic/cnnic*)

Not a unified experience!

| RIR | Features | Ease of use | API |
|---|---|---|---|
| AFRINIC | ⭐ | ⭐ | ⭐ |
| APNIC | ⭐⭐ | ⭐⭐ | ⭐ |
| ARIN | ⭐⭐ | ⭐⭐ | ⭐⭐ |
| LACNIC | ⭐ | ⭐⭐⭐ | ⭐ |
| RIPE | ⭐⭐⭐ | ⭐⭐⭐ | ⭐⭐⭐ |

CLOUDFLARE

# Rankings

Features: RRDP, 2 factors, extra info, CA

Ease of use: steps to sign a ROA, multi user

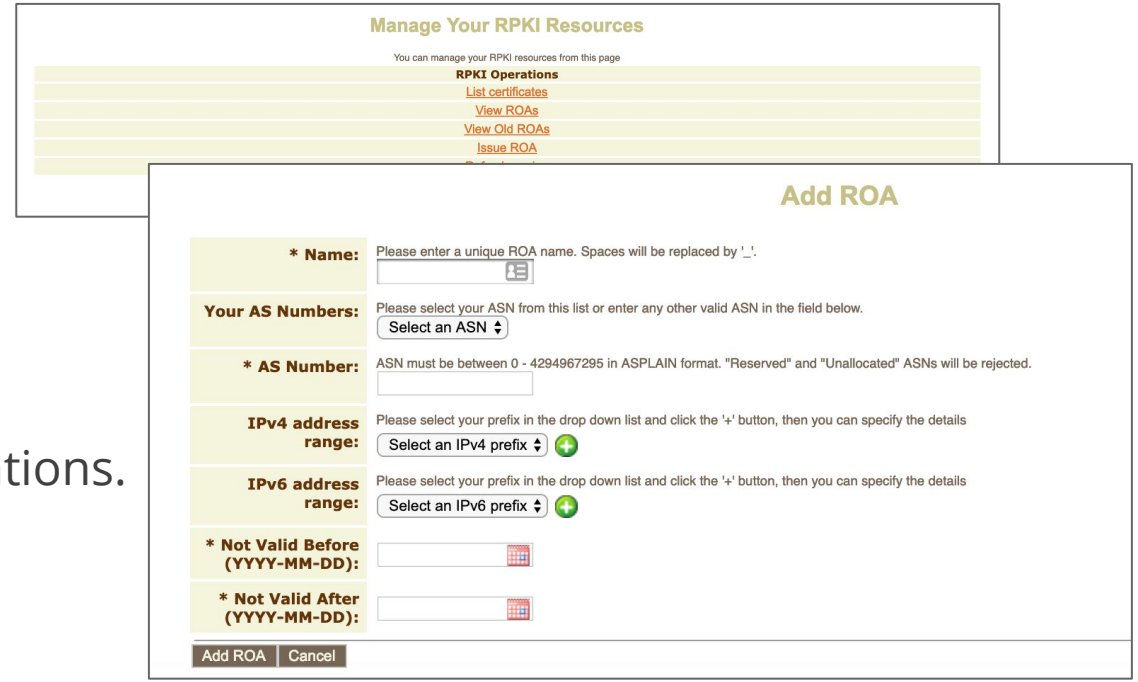
API: functional, complete and accessible

# Comparison - AFRINIC

Hard to set up: client TLS certificate to create (BPKI) in order to do RPKI.

- Buggy.
- No RRDP.
- No API.
- No auto-renew.
- Hosted CA possible.

Extensive certificate informations.




CLOUDFLARE

# Comparison - APNIC

Two factors or client certificate.

RRDP.

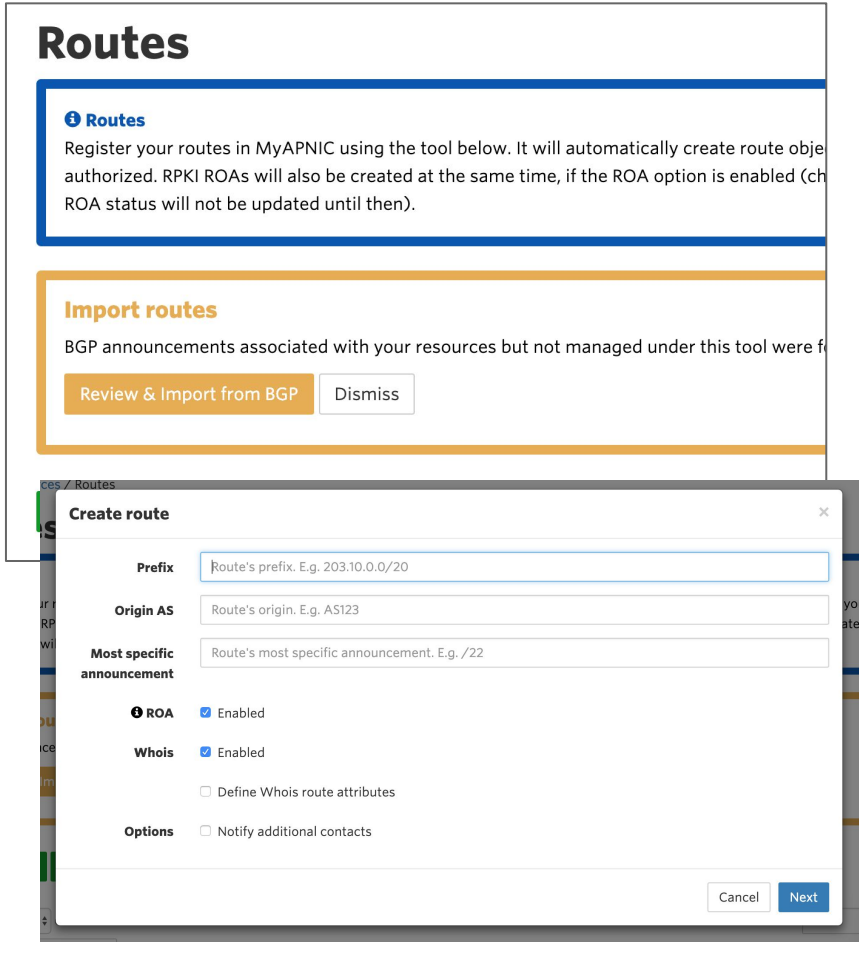
Auto-renew.

Allow BGP batch signing.

(slight bugs with large amount of prefixes).

Hosted CA possible.

Draft for API:
https://www.apnic.net/manage-ip/apnic-services/services-roadmap/public-api-draft-for-members/



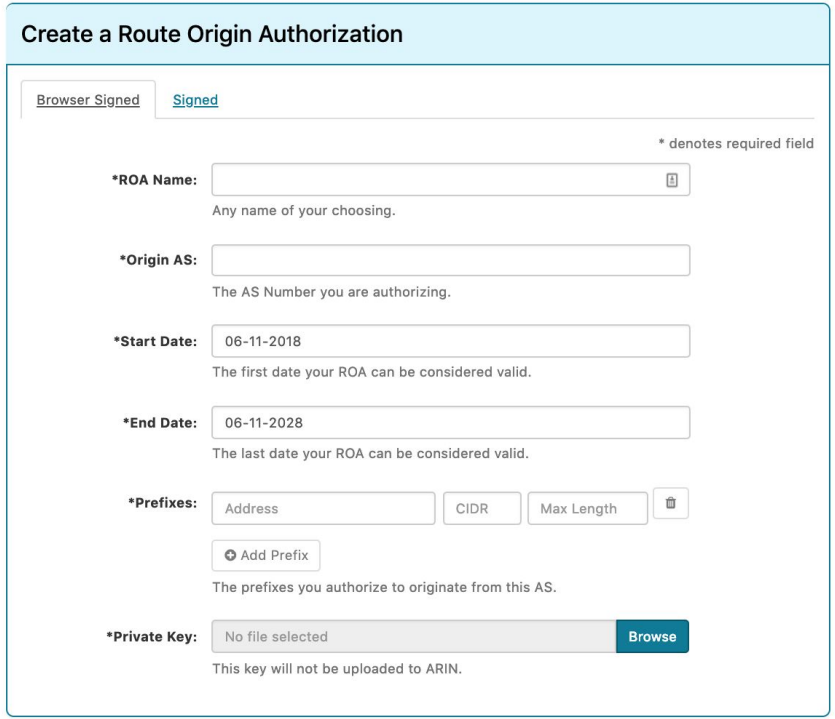

CLOUDFLARE

# Comparison - ARIN

Two factors. Separate signing key.

- No RRDP.
- No auto-renew.
- Semi-functional API (add).
- Dashboard not easy to find.
- Hosted CA possible.
- Slow rsync update (4 times a day).

Some certificate information.

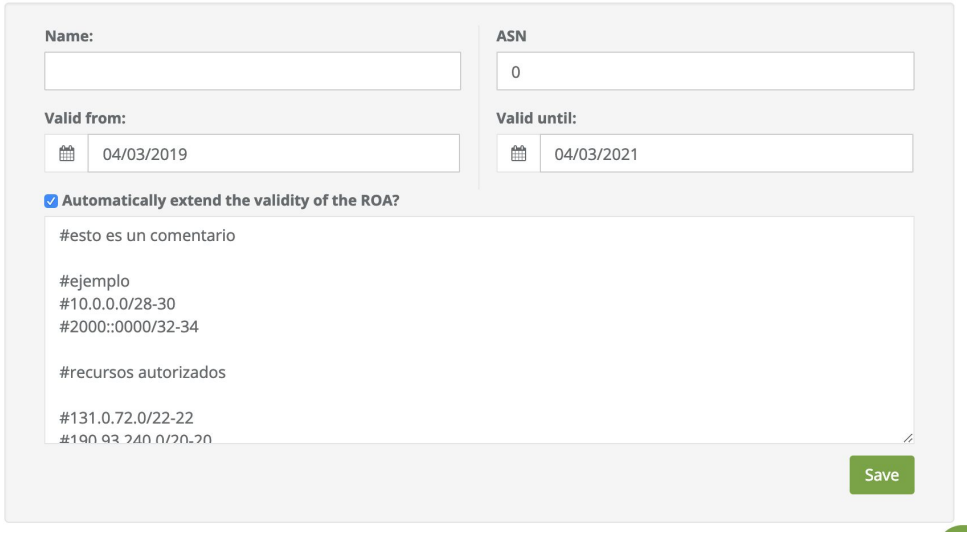



CLOUDFLARE

# Comparison - LACNIC

No two factors. Single user.

- No RRDP.
- No API.
- Auto-renew opt-in.
- Allow BGP batch signing.
- Based off RIPE.
- No Hosted CA.




Some extra info (revoked, path).

Incorrect certificate encoding (BER). High turnover of certificate (few days).

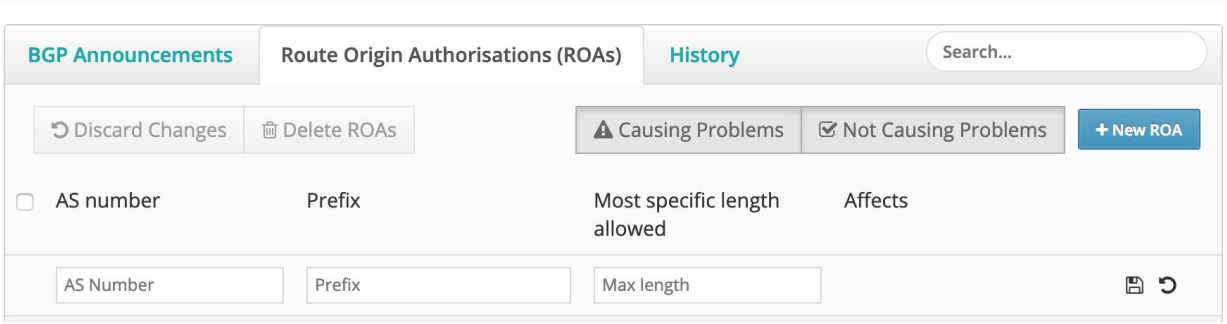
CLOUDFLARE

# Comparison - RIPE

Two factors.

- RRDP.
- Auto-renew.
- Nice API.
- Allow BGP batch signing.
- No Hosted CA (theoretically).
- No extra information. But history.

Incorrect certificate encoding (BER).




CLOUDFLARE

# Automation

We automated prefixes adding on **ARIN and RIPE** with a **Salt state**.

Two secrets to store (API key and signing key).

Cannot delete or list via API for ARIN: very prone to mistakes if user wants to reduce the amount of ROA files.

```
def _format_payload(roas, signature):
    template = """-----BEGIN ROA REQUEST-----
{roas}
-----END ROA REQUEST-----
-----BEGIN SIGNATURE-----
{signature}
-----END SIGNATURE-----
    """
    payload = template.format(
        roas=roas, signature="\n".join(textwrap.wrap(signature, width=64))
    )
    return payload


def _make_roa(name, asn, t, start_val, end_val, prefix, length, maxlength):
    template = (
        '1|{time}|{name}|{asn}|{start_val}|{end_val}|{prefix}|{length}|{maxlength}|'
    )
    time_str = calendar.timegm(t.timetuple())
    start_val_str = start_val.strftime(_TIME_FORMAT)
    end_val_str = end_val.strftime(_TIME_FORMAT)
    roa = template.format(
        time=time_str,
        name=name,
        asn=asn,
        start_val=start_val_str,
        end_val=end_val_str,
        prefix=prefix,
        length=length,
        maxlength=maxlength,
    )
    return roa


def _sign(pkey, roas):
    signature = pkey.sign(roas.encode('utf-8'), padding.PKCS1v15(), hashes.SHA256())
    return base64.b64encode(signature).decode('utf-8')
```

CLOUDFLARE

# Validator

# Why write a new validator?

November 2018: First release of NLnet Labs Routinator 3000 [1]

We were still using RIPE Validator

We wanted something more custom: with monitoring and RRDP

By building it in Golang:

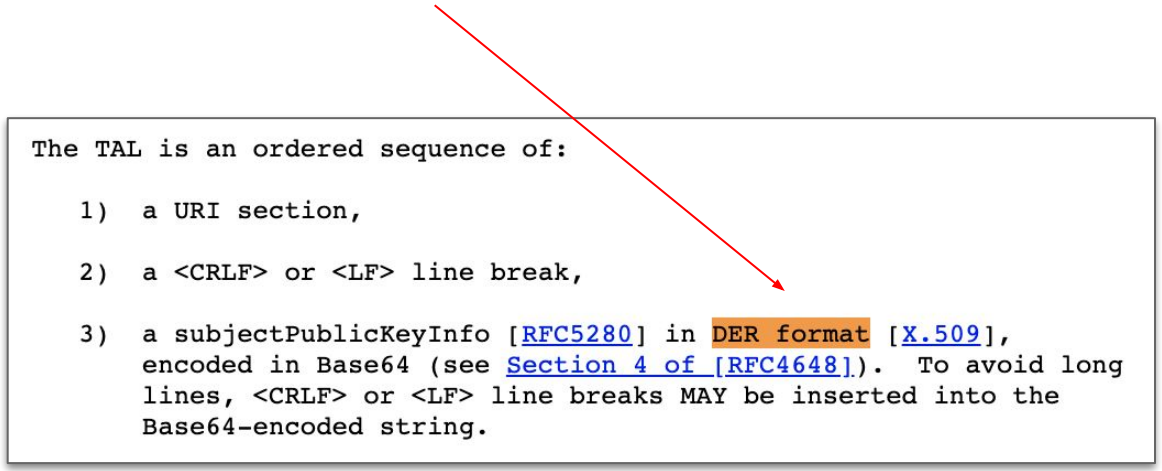
- Many APIs and easy concurrency
- Community doing cryptography
- Cloudflare uses Golang a lot (cfssl, sidh, etc.)


[1] https://github.com/NLnetLabs/routinator

# Challenges

Juniper bugs: Routing Validation disabled

Difficulties: rsync, BER encoded instead of DER, conditions in cryptography



The TAL is an ordered sequence of:

1) a URI section,

2) a <CRLF> or <LF> line break,

3) a subjectPublicKeyInfo [RFC5280] in DER format [X.509], encoded in Base64 (see Section 4 of [RFC4648]). To avoid long lines, <CRLF> or <LF> line breaks MAY be inserted into the Base64-encoded string.

CLOUDFLARE

https://tools.ietf.org/html/rfc7730

# Cloudflare's RPKI Toolkit

Sets of libraries and tools written in Go

Including ***OctoRPKI*** 

**https://github.com/cloudflare/cfrpki**

CLOUDFLARE

https://blog.cloudflare.com/cloudflares-rpki-toolkit/

# Cloudflare’s RPKI Toolkit


Libraries

- CER/ROA/MFT decoder
- PKI manager (exploring, validating)
- RRDP/rsync fetcher
- Validation of prefixes

Software

- Local validator (without RRDP/rsync)
- API tools for a distributed version without filesystem
- OctoRPKI
- Certificate Transparency tool

CLOUDFLARE

# OctoRPKI - Features (1/2)

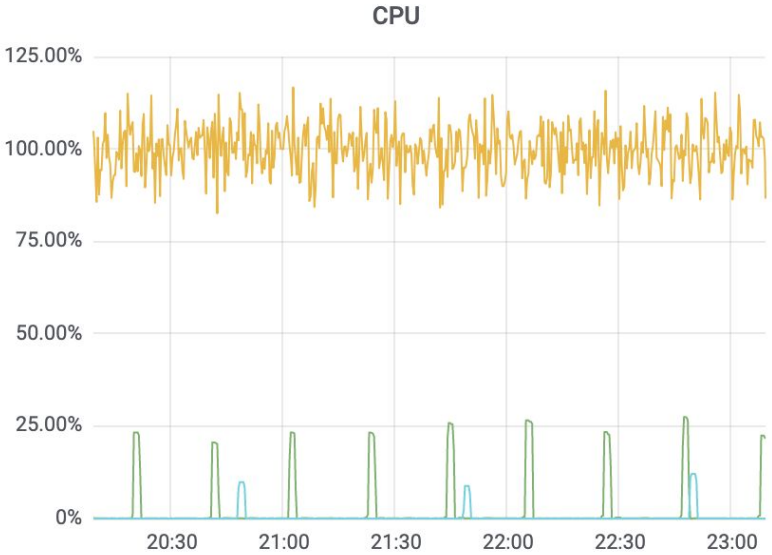
- Decodes TAL/CER/ROA/MFT
- Explore via Manifest or directory.
- RRDP support (and failover to rsync)
- Monitoring (Prometheus and JSON API which includes logs)
- Dockerizeable
- Handle stability (generate file when done)

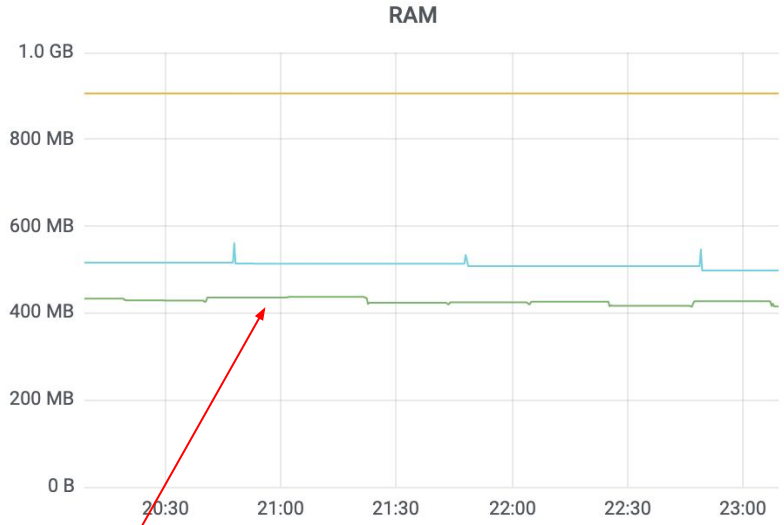
# OctoRPKI - Features (2/2)

- Full compatibility with GoRTR (including signing the JSON file)
- Server + caching options for generated file (CDN friendly)
- Configuration options
  - Disable/Enable components
  - Modes (server, one-off)
- ~5-15 minutes for a full cold-start sync

CLOUDFLARE

# OctoRPKI - Compute footprint



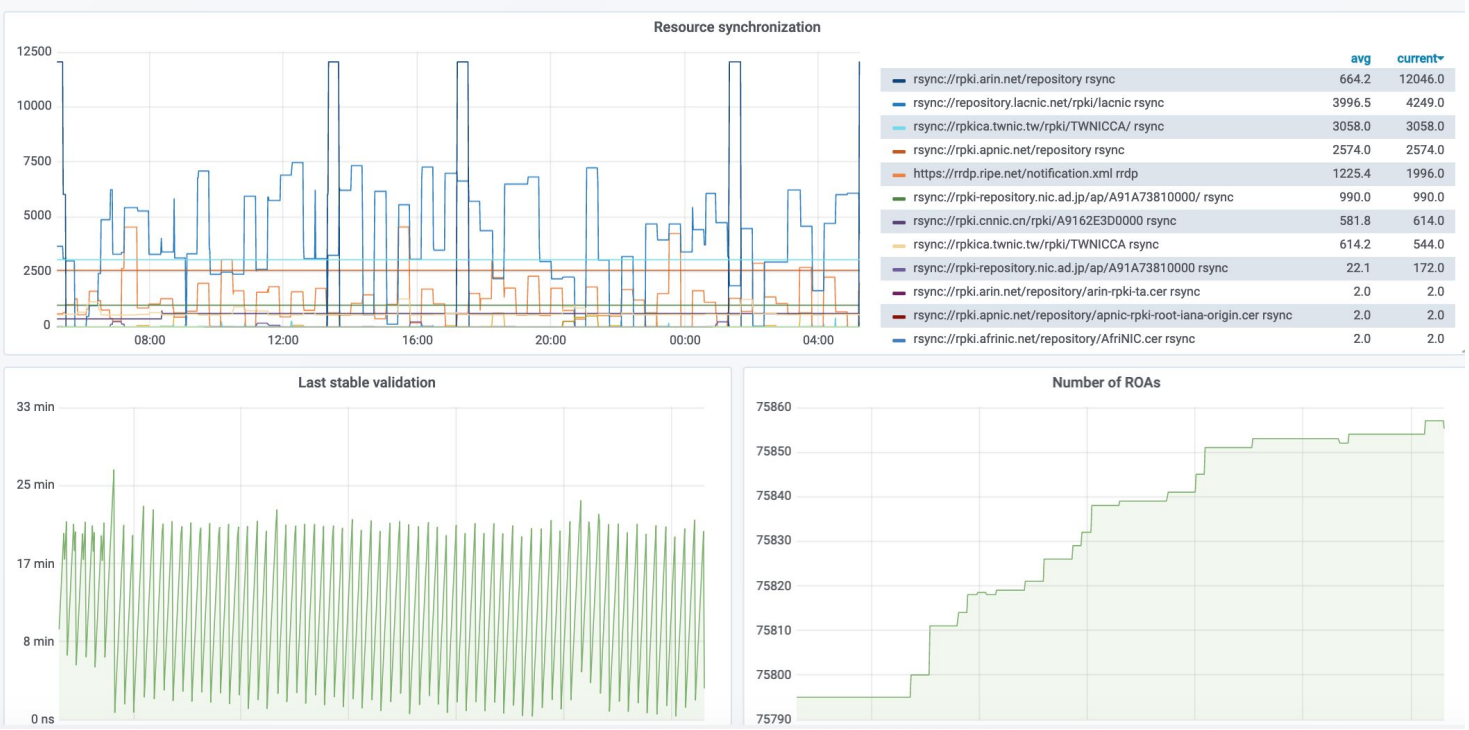




| | avg |
|---|---|
| rpki-benchmark-octorpki | 428 MB |
| rpki-benchmark-ripe | 906 MB |
| rpki-benchmark-routinator | 511 MB |

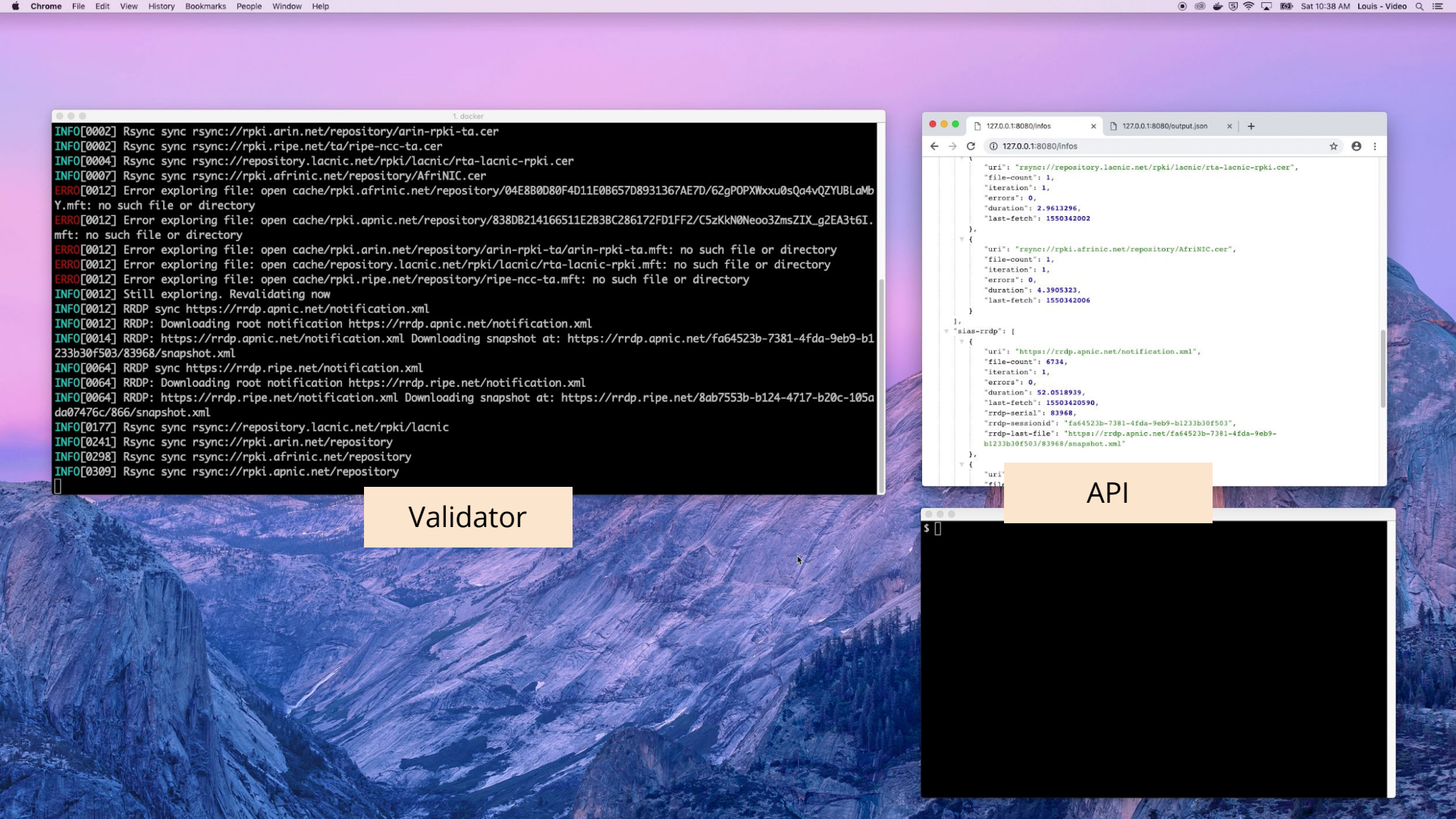
OctoRPKI v1.1.3
RIPE Validator v2.25
Routinator v3.3.0

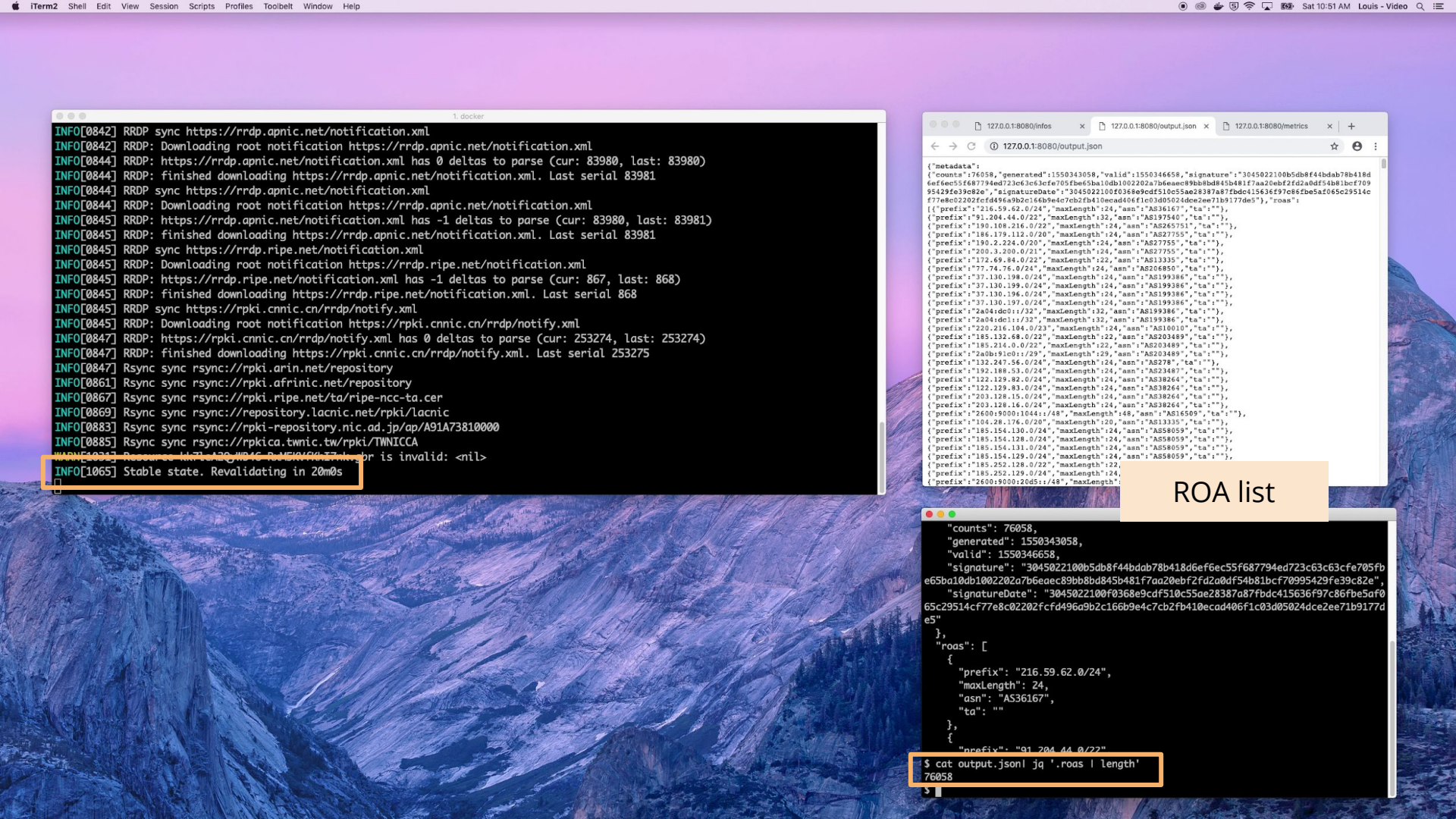
CLOUDFLARE

# Monitoring



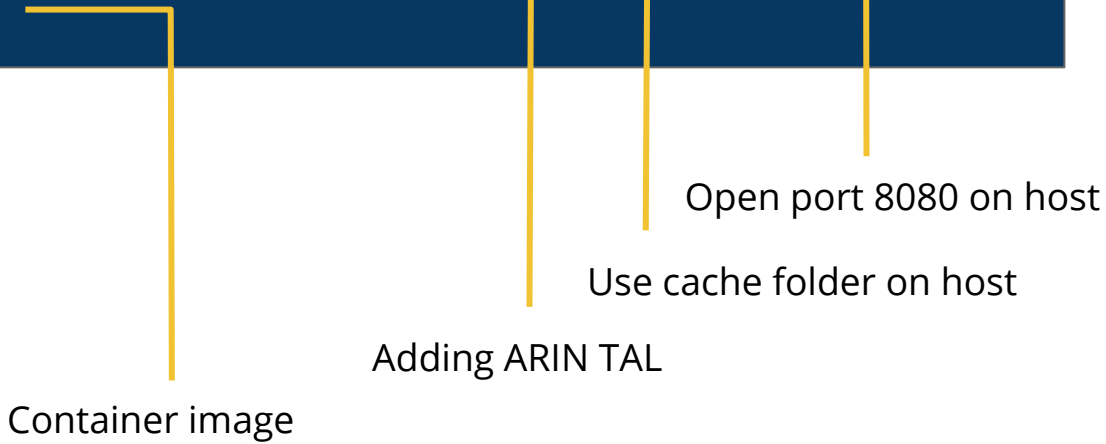

CLOUDFLARE


Chrome File Edit View History Bookmarks People Window Help
Sat 10:38 AM Louis - Video
1. docker
INFO[0002] Rsync sync rsync://rpki.arin.net/repository/arin-rpki-ta.cer
INFO[0002] Rsync sync rsync://rpki.ripe.net/ta/ripe-ncc-ta.cer
INFO[0004] Rsync sync rsync://repository.lacnic.net/rpki/lacnic/rta-lacnic-rpki.cer
INFO[0007] Rsync sync rsync://rpki.afrinic.net/repository/AfriNIC.cer
ERRO[0012] Error exploring file: open cache/rpki.afrinic.net/repository/04E8B0D80F4D11E0B657D8931367AE7D/6ZgPOPXWxxu0sQa4vQZYUBLaMbY.mft: no such file or directory
ERRO[0012] Error exploring file: open cache/rpki.apnic.net/repository/838DB214166511E2B3BC286172FD1FF2/C5zKkN0Neoo3ZmsZIX_g2EA3t6I.mft: no such file or directory
ERRO[0012] Error exploring file: open cache/rpki.arin.net/repository/arin-rpki-ta/arin-rpki-ta.mft: no such file or directory
ERRO[0012] Error exploring file: open cache/repository.lacnic.net/rpki/lacnic/rta-lacnic-rpki.mft: no such file or directory
ERRO[0012] Error exploring file: open cache/rpki.ripe.net/repository/ripe-ncc-ta.mft: no such file or directory
INFO[0012] Still exploring. Revalidating now
INFO[0012] RRDP sync https://rrdp.apnic.net/notification.xml
INFO[0012] RRDP: Downloading root notification https://rrdp.apnic.net/notification.xml
INFO[0014] RRDP: https://rrdp.apnic.net/notification.xml Downloading snapshot at: https://rrdp.apnic.net/fa64523b-7381-4fda-9eb9-b1233b30f503/83968/snapshot.xml
INFO[0064] RRDP sync https://rrdp.ripe.net/notification.xml
INFO[0064] RRDP: Downloading root notification https://rrdp.ripe.net/notification.xml
INFO[0064] RRDP: https://rrdp.ripe.net/notification.xml Downloading snapshot at: https://rrdp.ripe.net/8ab7553b-b124-4717-b20c-105ada07476c/866/snapshot.xml
INFO[0177] Rsync sync rsync://repository.lacnic.net/rpki/lacnic
INFO[0241] Rsync sync rsync://rpki.arin.net/repository
INFO[0298] Rsync sync rsync://rpki.afrinic.net/repository
INFO[0309] Rsync sync rsync://rpki.apnic.net/repository
Validator
127.0.0.1:8080/infos
127.0.0.1:8080/output.json
127.0.0.1:8080/infos
"uri": "rsync://repository.lacnic.net/rpki/lacnic/rta-lacnic-rpki.cer",
"file-count": 1,
"iteration": 1,
"errors": 0,
"duration": 2.9613296,
"last-fetch": 1550342002
"uri": "rsync://rpki.afrinic.net/repository/AfriNIC.cer",
"file-count": 1,
"iteration": 1,
"errors": 0,
"duration": 4.3905323,
"last-fetch": 1550342006
"sias-rrdp": [
"uri": "https://rrdp.apnic.net/notification.xml",
"file-count": 6734,
"iteration": 1,
"errors": 0,
"duration": 52.0518939,
"last-fetch": 15503420590,
"rrdp-serial": 83968,
"rrdp-sessionid": "fa64523b-7381-4fda-9eb9-b1233b30f503",
"rrdp-last-file": "https://rrdp.apnic.net/fa64523b-7381-4fda-9eb9-b1233b30f503/83968/snapshot.xml"
API
$


iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help
Sat 10:51 AM Louis - Video
1. docker
INFO[0842] RRDP sync https://rrdp.apnic.net/notification.xml
INFO[0842] RRDP: Downloading root notification https://rrdp.apnic.net/notification.xml
INFO[0844] RRDP: https://rrdp.apnic.net/notification.xml has 0 deltas to parse (cur: 83980, last: 83980)
INFO[0844] RRDP: finished downloading https://rrdp.apnic.net/notification.xml. Last serial 83981
INFO[0844] RRDP sync https://rrdp.apnic.net/notification.xml
INFO[0844] RRDP: Downloading root notification https://rrdp.apnic.net/notification.xml
INFO[0845] RRDP: https://rrdp.apnic.net/notification.xml has -1 deltas to parse (cur: 83980, last: 83981)
INFO[0845] RRDP: finished downloading https://rrdp.apnic.net/notification.xml. Last serial 83981
INFO[0845] RRDP sync https://rrdp.ripe.net/notification.xml
INFO[0845] RRDP: Downloading root notification https://rrdp.ripe.net/notification.xml
INFO[0845] RRDP: https://rrdp.ripe.net/notification.xml has -1 deltas to parse (cur: 867, last: 868)
INFO[0845] RRDP: finished downloading https://rrdp.ripe.net/notification.xml. Last serial 868
INFO[0845] RRDP sync https://rpki.cnnic.cn/rrdp/notify.xml
INFO[0845] RRDP: Downloading root notification https://rpki.cnnic.cn/rrdp/notify.xml
INFO[0847] RRDP: https://rpki.cnnic.cn/rrdp/notify.xml has 0 deltas to parse (cur: 253274, last: 253274)
INFO[0847] RRDP: finished downloading https://rpki.cnnic.cn/rrdp/notify.xml. Last serial 253275
INFO[0847] Rsync sync rsync://rpki.arin.net/repository
INFO[0861] Rsync sync rsync://rpki.afrinic.net/repository
INFO[0867] Rsync sync rsync://rpki.ripe.net/ta/ripe-ncc-ta.cer
INFO[0869] Rsync sync rsync://repository.lacnic.net/rpki/lacnic
INFO[0883] Rsync sync rsync://rpki-repository.nic.ad.jp/ap/ap/A91A73810000
INFO[0885] Rsync sync rsync://rpkica.twnic.tw/rpki/TWNICCA
is invalid: <nil>
INFO[1065] Stable state. Revalidating in 20m0s
127.0.0.1:8080/infos
127.0.0.1:8080/output.json
127.0.0.1:8080/metrics
127.0.0.1:8080/output.json
{"metadata":
{"counts":76058,"generated":1550343058,"valid":1550346658,"signature":"3045022100b5db8f44bdab78b418d6ef6ec55f687794ed723c63c63cfe705fbe65ba10db1002202a7b6eaec89bb8bd845b481f7aa20ebf2fd2a0df54b81bcf70995429fe39c82e","signatureDate":"3045022100f0368e9cdf510c55ae28387a87fbdc415636f97c86fbe5af065c29514cf77e8c02202fcfd496a9b2c166b9e4c7cb2fb410ecad406f1c03d05024dce2ee71b9177de5"},"roas":
[{"prefix":"216.59.62.0/24","maxLength":24,"asn":"AS36167","ta":""},
{"prefix":"91.204.44.0/22","maxLength":32,"asn":"AS197540","ta":""},
{"prefix":"190.108.216.0/22","maxLength":24,"asn":"AS265751","ta":""},
{"prefix":"186.179.112.0/20","maxLength":24,"asn":"AS27755","ta":""},
{"prefix":"190.2.224.0/20","maxLength":24,"asn":"AS27755","ta":""},
{"prefix":"200.3.200.0/21","maxLength":24,"asn":"AS27755","ta":""},
{"prefix":"172.69.84.0/22","maxLength":22,"asn":"AS13335","ta":""},
{"prefix":"77.74.76.0/24","maxLength":24,"asn":"AS206850","ta":""},
{"prefix":"37.130.198.0/24","maxLength":24,"asn":"AS199386","ta":""},
{"prefix":"37.130.199.0/24","maxLength":24,"asn":"AS199386","ta":""},
{"prefix":"37.130.196.0/24","maxLength":24,"asn":"AS199386","ta":""},
{"prefix":"37.130.197.0/24","maxLength":24,"asn":"AS199386","ta":""},
{"prefix":"2a04:dc0::/32","maxLength":32,"asn":"AS199386","ta":""},
{"prefix":"2a04:dc1::/32","maxLength":32,"asn":"AS199386","ta":""},
{"prefix":"220.216.104.0/23","maxLength":24,"asn":"AS10010","ta":""},
{"prefix":"185.132.68.0/22","maxLength":22,"asn":"AS203489","ta":""},
{"prefix":"185.214.0.0/22","maxLength":22,"asn":"AS203489","ta":""},
{"prefix":"2a0b:91c0::/29","maxLength":29,"asn":"AS203489","ta":""},
{"prefix":"132.247.56.0/24","maxLength":24,"asn":"AS278","ta":""},
{"prefix":"192.188.53.0/24","maxLength":24,"asn":"AS23487","ta":""},
{"prefix":"122.129.82.0/24","maxLength":24,"asn":"AS38264","ta":""},
{"prefix":"122.129.83.0/24","maxLength":24,"asn":"AS38264","ta":""},
{"prefix":"203.128.15.0/24","maxLength":24,"asn":"AS38264","ta":""},
{"prefix":"203.128.16.0/24","maxLength":24,"asn":"AS38264","ta":""},
{"prefix":"2600:9000:1044::/48","maxLength":48,"asn":"AS16509","ta":""},
{"prefix":"104.28.176.0/20","maxLength":20,"asn":"AS13335","ta":""},
{"prefix":"185.154.130.0/24","maxLength":24,"asn":"AS58059","ta":""},
{"prefix":"185.154.128.0/24","maxLength":24,"asn":"AS58059","ta":""},
{"prefix":"185.154.131.0/24","maxLength":24,"asn":"AS58059","ta":""},
{"prefix":"185.154.129.0/24","maxLength":24,"asn":"AS58059","ta":""},
{"prefix":"185.252.128.0/22","maxLength":22,
{"prefix":"185.252.129.0/24","maxLength":24,
{"prefix":"2600:9000:20d5::/48","maxLength"
ROA list
"counts": 76058,
"generated": 1550343058,
"valid": 1550346658,
"signature": "3045022100b5db8f44bdab78b418d6ef6ec55f687794ed723c63c63cfe705fbe65ba10db1002202a7b6eaec89bb8bd845b481f7aa20ebf2fd2a0df54b81bcf70995429fe39c82e",
"signatureDate": "3045022100f0368e9cdf510c55ae28387a87fbdc415636f97c86fbe5af065c29514cf77e8c02202fcfd496a9b2c166b9e4c7cb2fb410ecad406f1c03d05024dce2ee71b9177de5"
},
"roas": [
{
"prefix": "216.59.62.0/24",
"maxLength": 24,
"asn": "AS36167",
"ta": ""
},
{
"prefix": "91.204.44.0/22"
$ cat output.json| jq '.roas | length'
76058
$

# OctoRPKI - Run it yourself

```
$ docker run -ti \
    -p 8080:8080 \
    -v $PWD/cache:/cache \
    -v $PWD/tals/arin.tal:/tals/arin.tal \
    cloudflare/octorpki
```




CLOUDFLARE

https://hub.docker.com/r/cloudflare/octorpki

# GoRTR

```
$ docker run -ti \
    -p 8082:8082 \
    -v $PWD/example.pub:/example.pub \
    cloudflare/gortr \
      -verify.key /example.pub \
      -cache https://YOUR_ROA_URL
```

**OctoRPKI** does not embed a RTR server. Modular and independence!

Fully compatible with **GoRTR** **https://github.com/cloudflare/gortr**

Signs the prefix list to ensure a safe distribution of the file.

**Can run natively on Juniper!**

CLOUDFLARE

# GoRTR

The only software to support **plaintext**, **SSH** and **TLS** as transports

### Compatibility matrix

A simple comparison between software and devices. Implementations on versions may vary.

| Device/software | Plaintext | TLS | SSH | Notes |
|---|---|---|---|---|
| RTRdump | Yes | Yes | Yes | |
| Juniper | Yes | No | No | |
| Cisco | Yes | No | Yes | Only SSH password |
| Alcatel | Yes | No | No | |
| Arista | No | No | No | |
| FRRouting | Yes | No | Yes | Only SSH password |
| Bird | Yes | No | Yes | Only SSH key |
| Quagga | Yes | No | No | |

CLOUDFLARE

https://github.com/cloudflare/gortr#configurations

# GoRTR without installing anything

**SSH:**

`rtr.rpki.cloudflare.com:8283` (user: `rpki` / pass: `rpki`)

**Plaintext:**

`rtr.rpki.cloudflare.com:8282`

Just configure into your router and go!

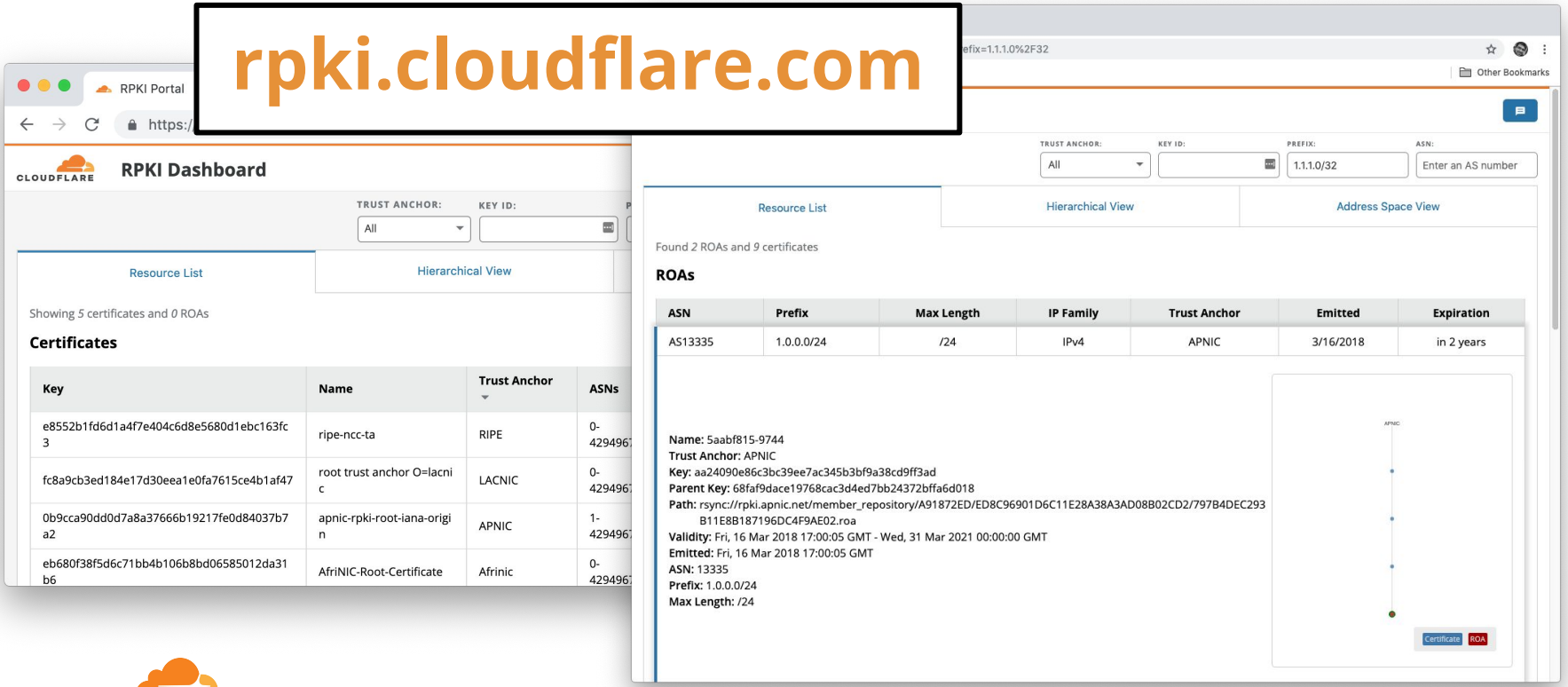
```
router bgp 65001
 rpki server 192.168.1.100
  transport tcp port 8282
 !
!
```

CLOUDFLARE

https://github.com/cloudflare/gortr#configurations

# Cloudflare's Internal Version

**rpki.cloudflare.com**

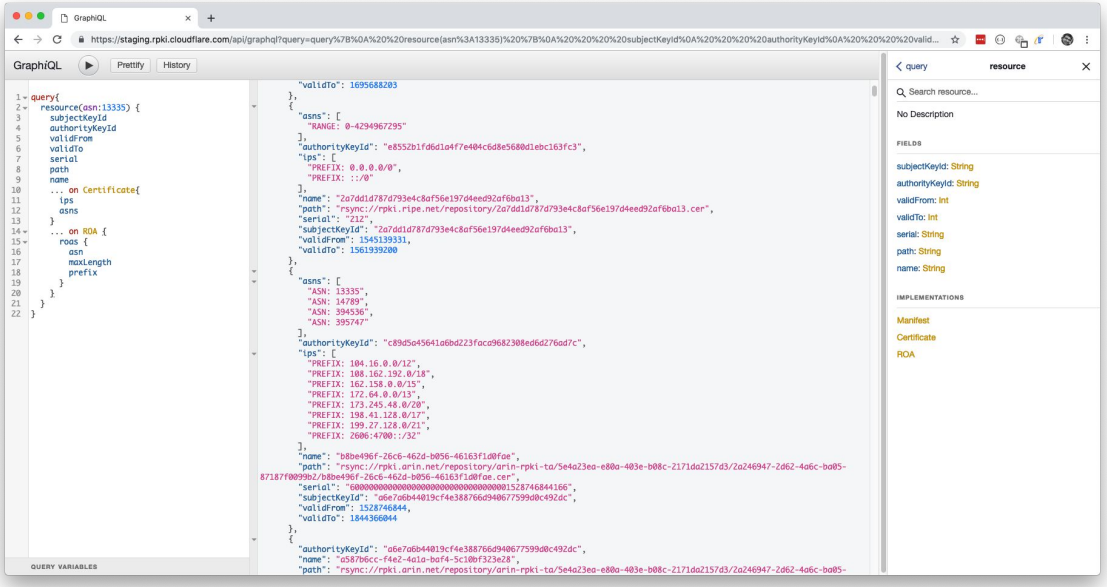



CLOUDFLARE

# Cloudflare's Internal Version

Provides `https://rpki.cloudflare.com/rpki.json`

Also a **GraphQL** API for the dashboard



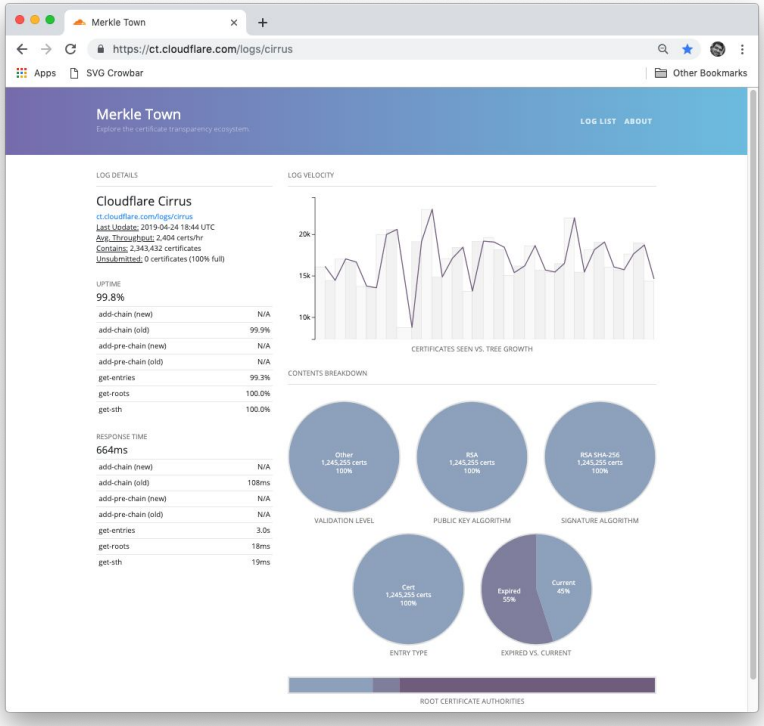

CLOUDFLARE

# Certificate Transparency

Historical records of certificates

Contains a chain (root → ROA)

Sent by our validator




CLOUDFLARE

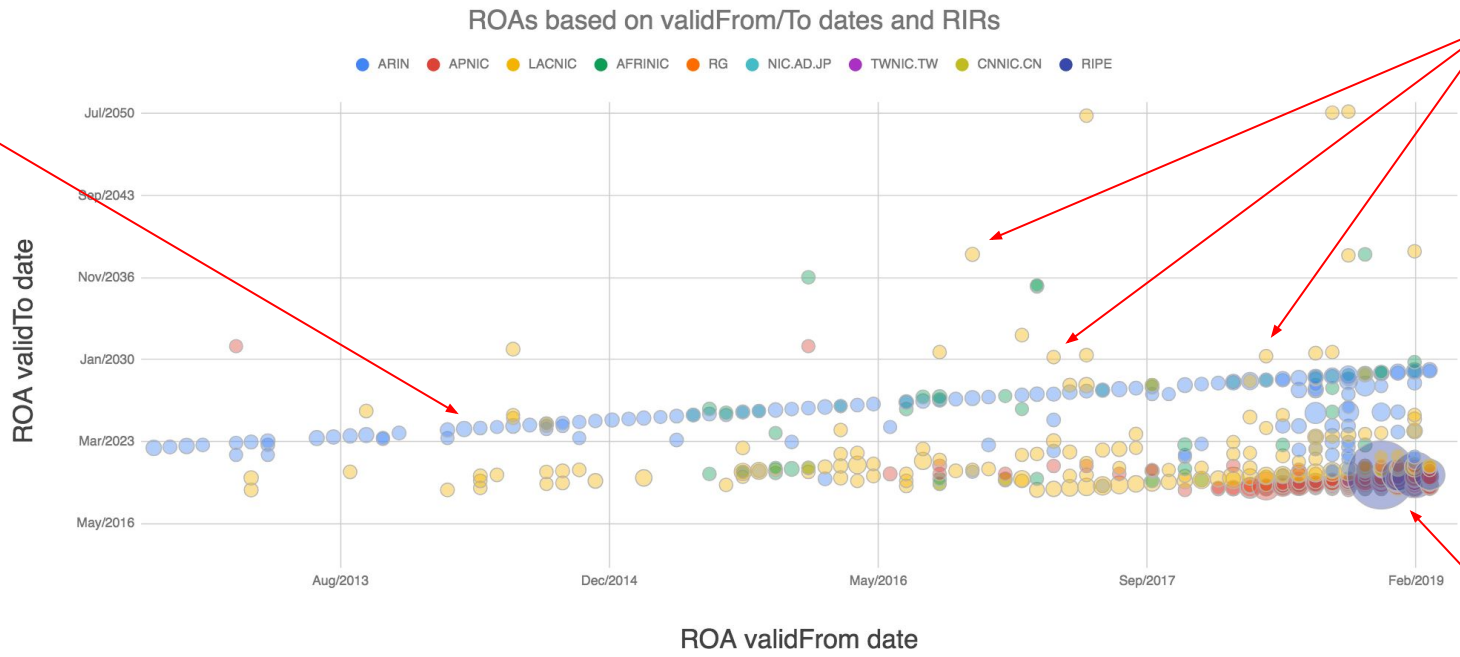
https://ct.cloudflare.com/logs/cirrus

# Other data

# Other data - so how fresh are those ROAs?

ARIN uses ten year expire

LACNIC random expires




RIPE regenerates certificates!

CLOUDFLARE

# Future projects

# Future projects or ideas

Certificate encoder, ASPA.

More toolings and visualizations around RPKI (BGP collection):

- Integration in our portal `https://peering.cloudflare.com/` *(ask for your free access)*

# Who validates?

# The Probing Project

Could we probe the entire Internet [1] to see who is doing validation?

CLOUDFLARE

[1] v4 IPs that are public and open to probing, around 3.9 billions

# Who?

Involved in this project and special thanks:

- Job Snijders: NTT, NLNOG
- Jérôme Fleury, Vasco Asturiano: Cloudflare

# Methodology

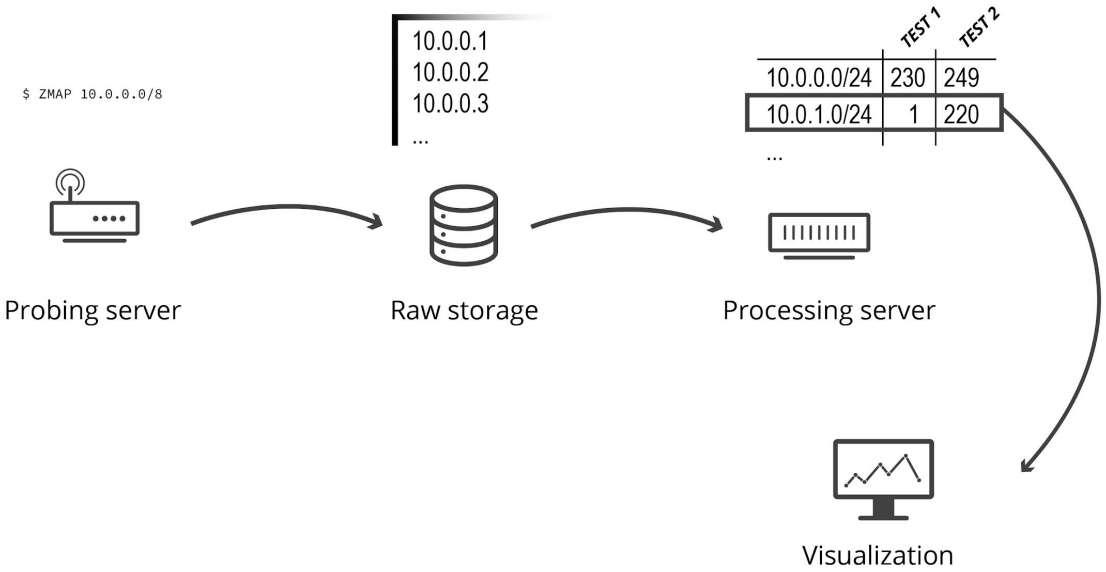
1. Run two tests with zmap (~2hrs/test):

   one test with behind an RPKI valid prefix and ;

   one test behind an RPKI invalid prefix

2. Sum the IPs that replied by range for each test
3. Visualize the ratio of replies between the two tests per prefix

# Methodology




CLOUDFLARE

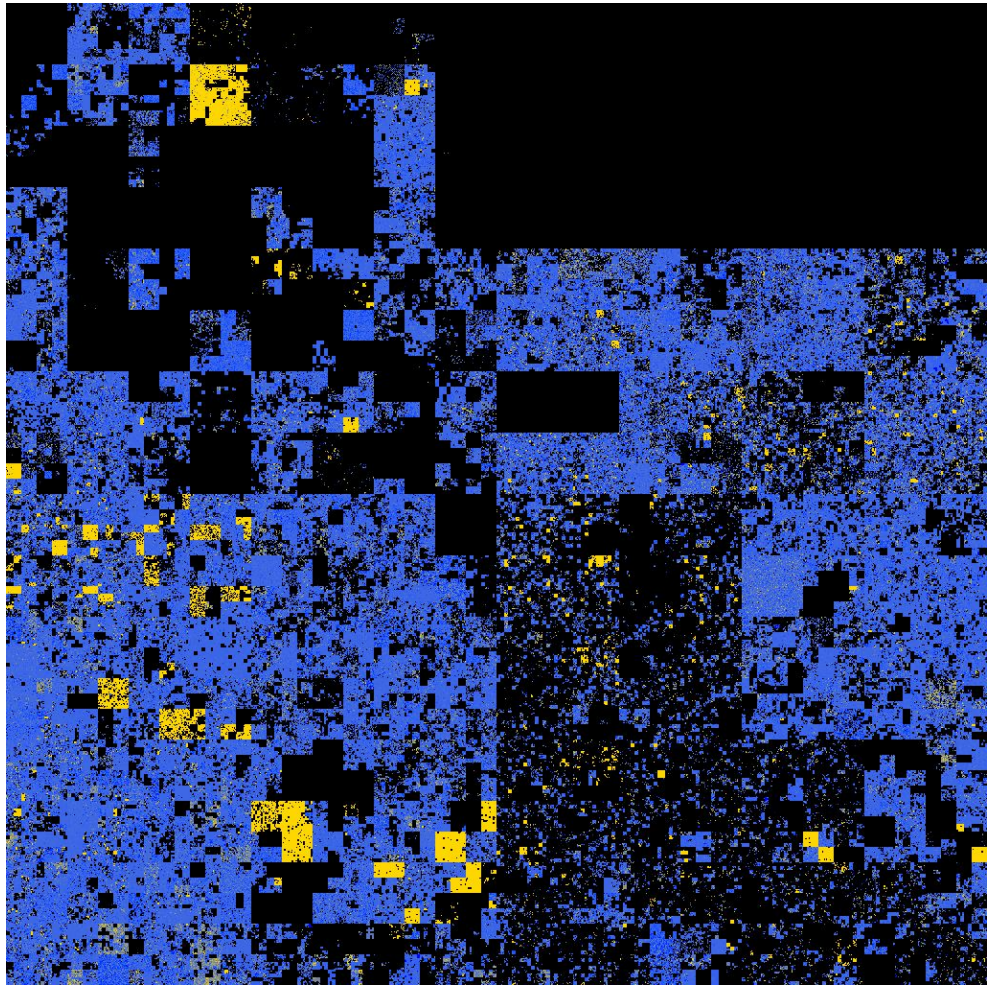
# Hilbert

*1024x1024 → 1 pixel per /20*

Ratio

$$\frac{\text{Received with source as invalid prefix}}{\text{Received with source as valid prefix}}$$

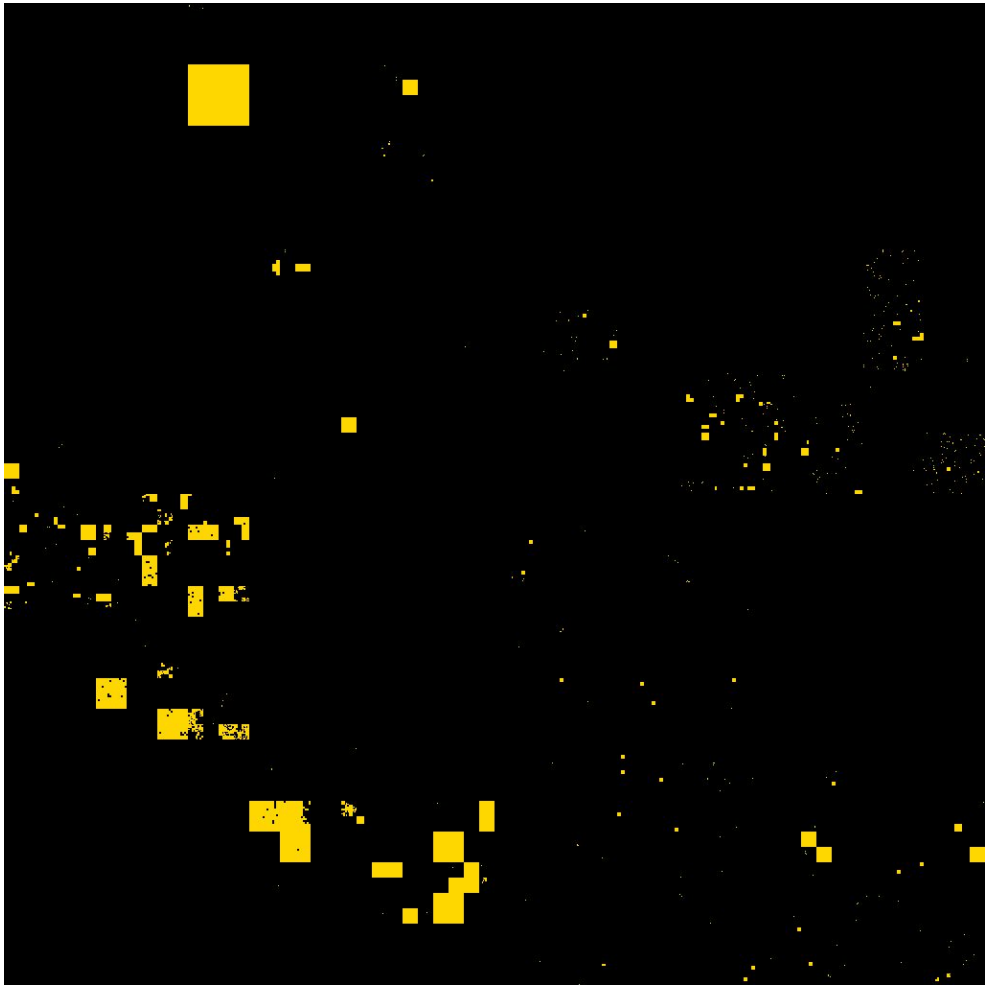
Good!




CLOUDFLARE

# Hilbert

**AT&T**
**Announced IP addresses**



CLOUDFLARE

# Learnings

AT&T has ~400k /24.

Easiest to visualize because big blocks.

Could we **automatically** identify networks?

*Going from “0 to 1” to “0 and 1” (0 good and 1 bad)*

# Machine learning

Simplify the identification of “validating prefixes”.

Requires **training** instead of defining specific ratios.

https://www.pexels.com/photo/man-pouring-condiment-on-plate-of-food-2494658/

*Little particles of artificial intelligence*

*Your network*

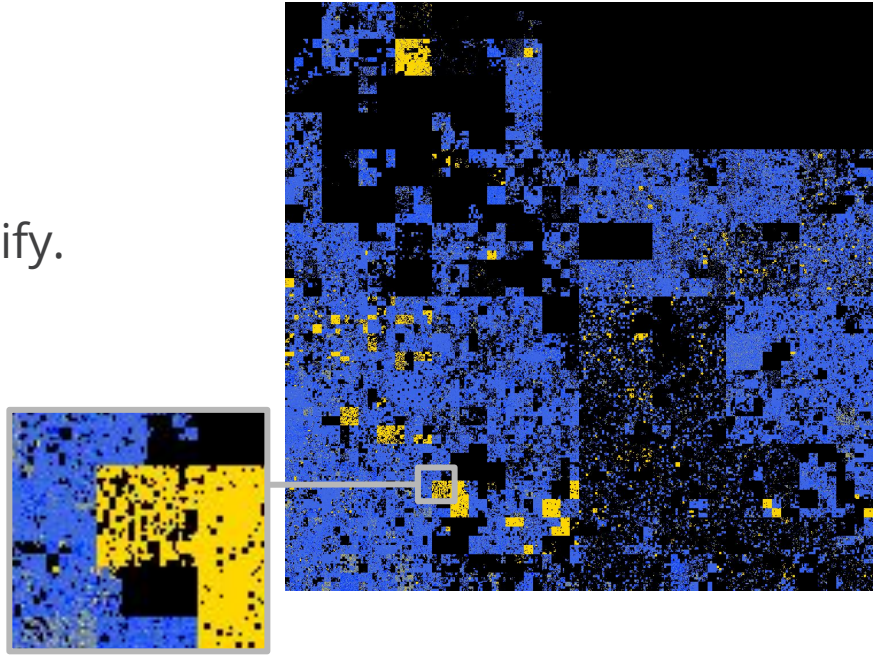
CLOUDFLARE

# Training

Take an easy sample to manually identify.

~6,500 /20 prefixes

Assisted by script: (with ASN mapping)



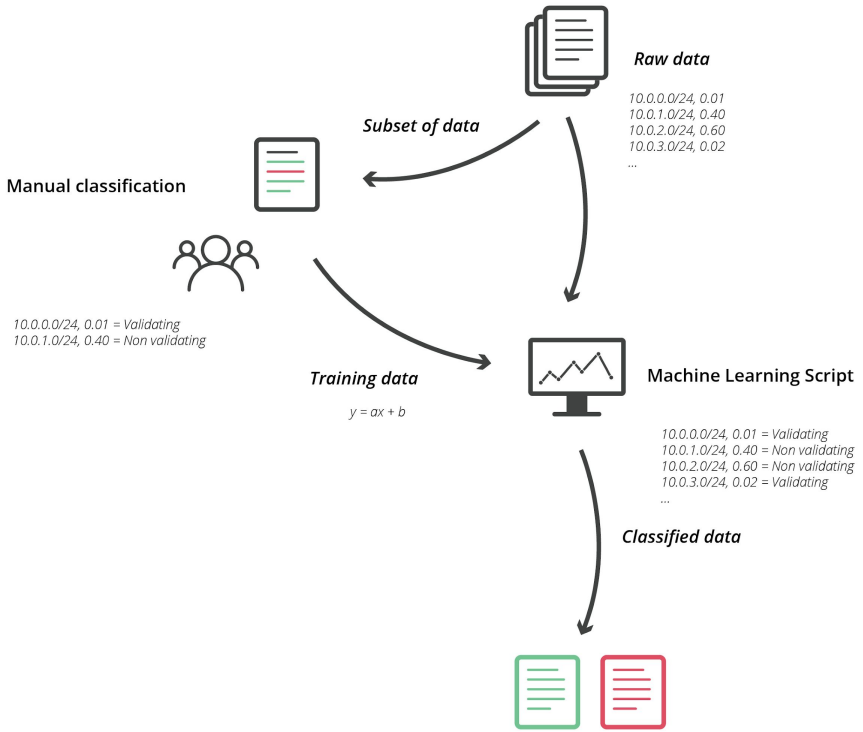
```
$ ./create_training.py \
    --input stripped-data.csv \
    --asn-valids 7018 \
    --asn-invalids 701 \
    --subnets 96.0.0.0/8 97.0.0.0/8 98.0.0.0/8 \
    --output training_data.csv
```

CLOUDFLARE

# Classification

The SVM model will take a few minutes to run on 433k /20's.

Model improvements: adds more variables (proximity to other low-ratio prefixes...).




CLOUDFLARE

# Results

Detected validating:

/20 detected: 28,199

/24 detected: 320k

less than AT&T total prefixes assumed validating:

not everything is responding to ICMP

Work in progress!

# Recent Leaks And Conclusions

# Summary of Amazon Route Hijack

An attacker announces Amazon Authority DNS prefixes.

Cloudflare and Google accept them in specific locations.

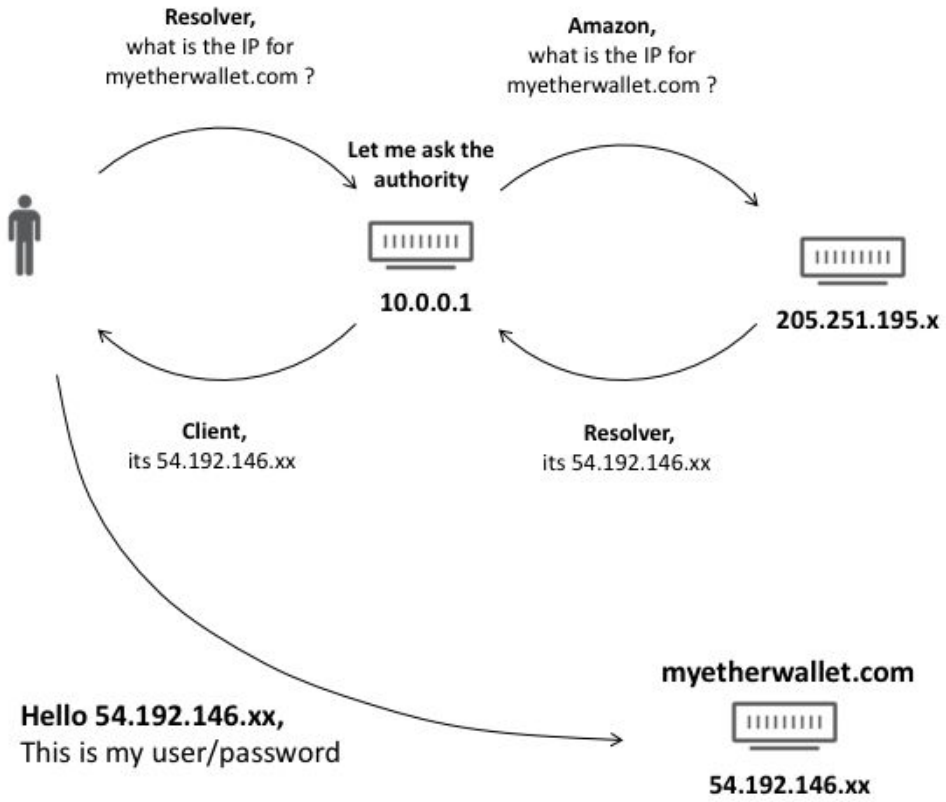
Cloudflare and Google DNS resolvers use this route when clients request the website, the attacker's server is returned.

The server has a phishing website for the client.

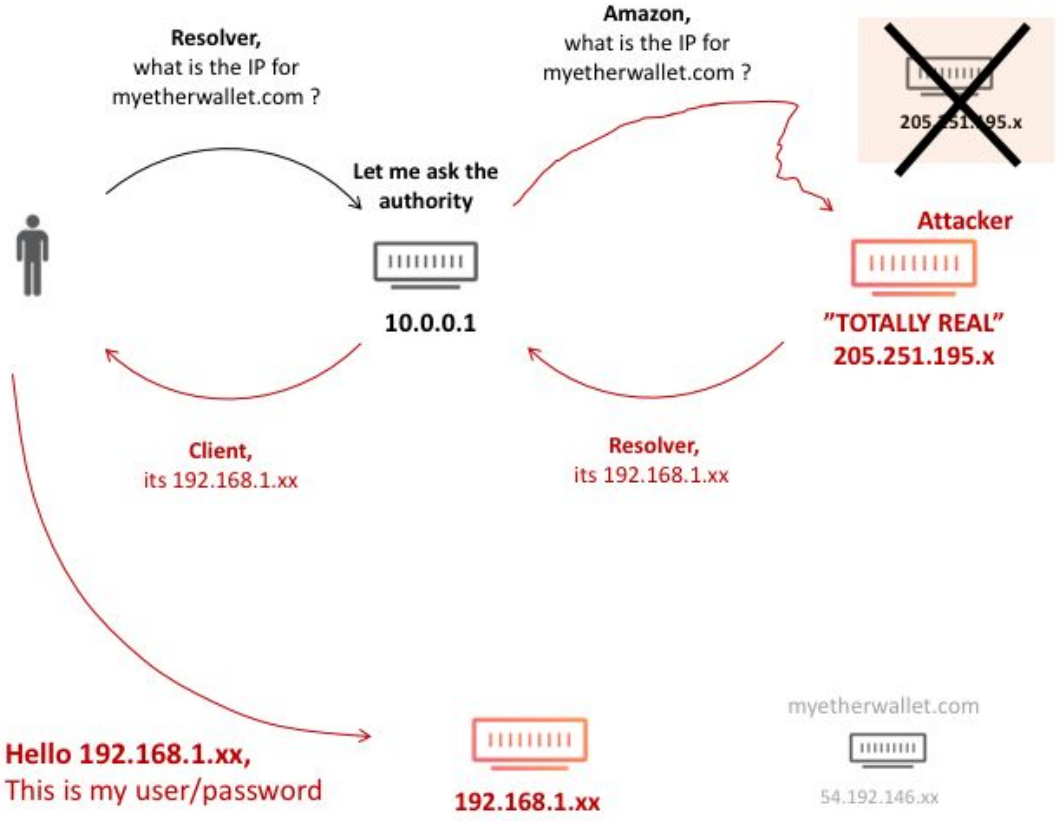
Attacker gather credentials and steals Bitcoins.

CLOUDFLARE

https://blog.cloudflare.com/bgp-leaks-and-crypto-currencies/


Resolver,
what is the IP for
myetherwallet.com ?
Amazon,
what is the IP for
myetherwallet.com ?
Let me ask the
authority
10.0.0.1
205.251.195.x
Client,
its 54.192.146.xx
Resolver,
its 54.192.146.xx
Hello 54.192.146.xx,
This is my user/password
myetherwallet.com
54.192.146.xx

CLOUDFLARE


Resolver,
what is the IP for
myetherwallet.com ?
Amazon,
what is the IP for
myetherwallet.com ?
205.251.195.x
Let me ask the
authority
Attacker
10.0.0.1
"TOTALLY REAL"
205.251.195.x
Client,
its 192.168.1.xx
Resolver,
its 192.168.1.xx
Hello 192.168.1.xx,
This is my user/password
192.168.1.xx
myetherwallet.com
54.192.146.xx

CLOUDFLARE

# Summary of Amazon Route Hijack

Amazon did not have signed routes

Cloudflare did not do RPKI validation + route filtering

**If RPKI was deployed:**

Route would have been rejected because wrong origin

CLOUDFLARE

https://blog.cloudflare.com/bgp-leaks-and-crypto-currencies/

# Summary of Verizon Route Leak

A company has two Internet accesses: Verizon and another ISP.

The ISP has a BGP optimizer which feeds more-specific routes.

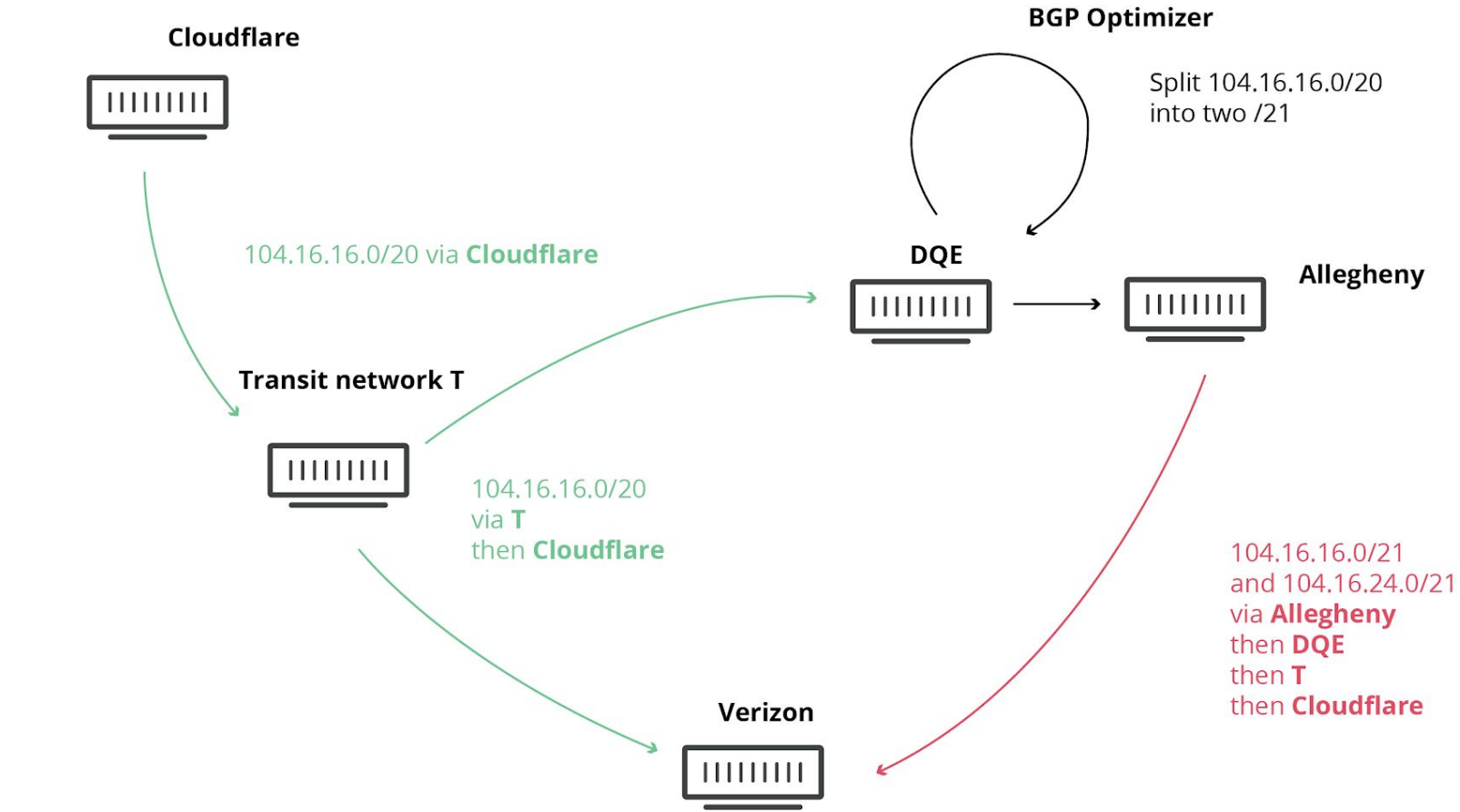
Unfortunately, the ISP sends the routes to the company which end up being sent to Verizon.

Verizon did not filter them and re-announces them to its peers and clients.

Cloudflare loses traffic.

CLOUDFLARE

https://blog.cloudflare.com/how-verizon-and-a-bgp-optimizer-knocked-large-parts-of-the-internet-offline-today/
https://blog.cloudflare.com/the-deep-dive-into-how-verizon-and-a-bgp-optimizer-knocked-large-parts-of-the-internet-offline-monday/


Cloudflare
BGP Optimizer
Split 104.16.16.0/20
into two /21
104.16.16.0/20 via Cloudflare
DQE
Allegheny
Transit network T
104.16.16.0/20
via T
then Cloudflare
104.16.16.0/21
and 104.16.24.0/21
via Allegheny
then DQE
then T
then Cloudflare
Verizon

CLOUDFLARE

# Summary of Verizon Route leak

Cloudflare had signed routes.

Verizon did not filter. Many networks accepted the leak.

*Cloudflare filtering routes did not matter here.*

**If basic filtering was deployed:**

Peering sessions would have been removed when going above prefix threshold.

AS-Path filtering could have avoided accepting routes.

**If RPKI was deployed:**

Routes would have been rejected because wrong length.

CLOUDFLARE

https://blog.cloudflare.com/how-verizon-and-a-bgp-optimizer-knocked-large-parts-of-the-internet-offline-today/
https://blog.cloudflare.com/the-deep-dive-into-how-verizon-and-a-bgp-optimizer-knocked-large-parts-of-the-internet-offline-monday/

# What we learned

*RPKI will not be the solution to everything. But in our stories…*

**Filtering** solves Amazon being hijacked

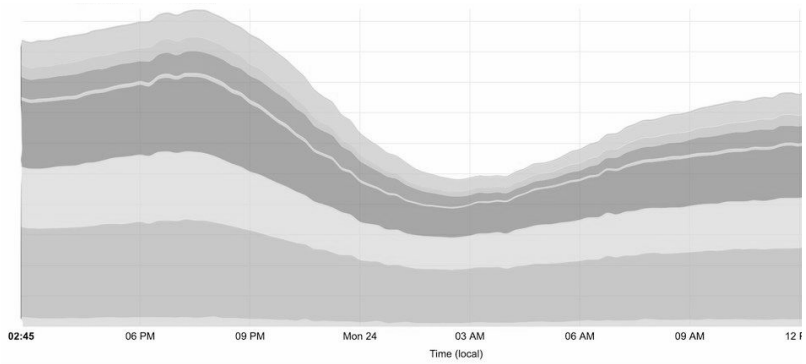
**Signing** helps your network not being leaked

CLOUDFLARE

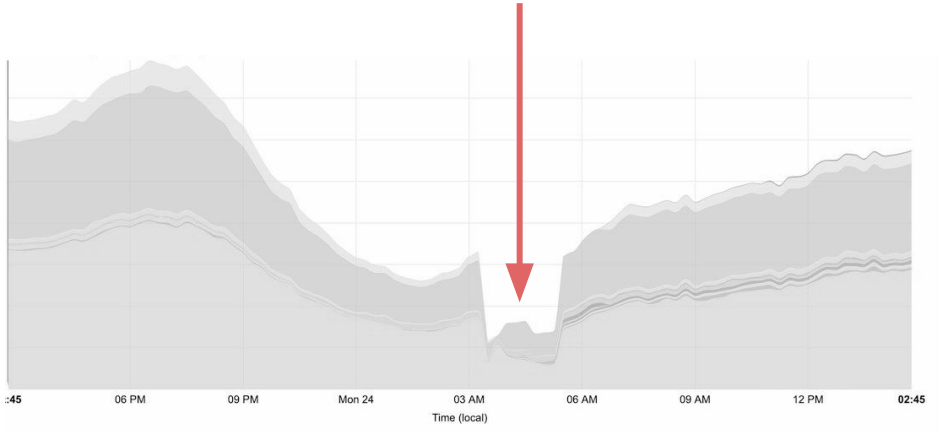
# Deploy RPKI now

*Because tomorrow is already too late*

With filtering




Without filtering

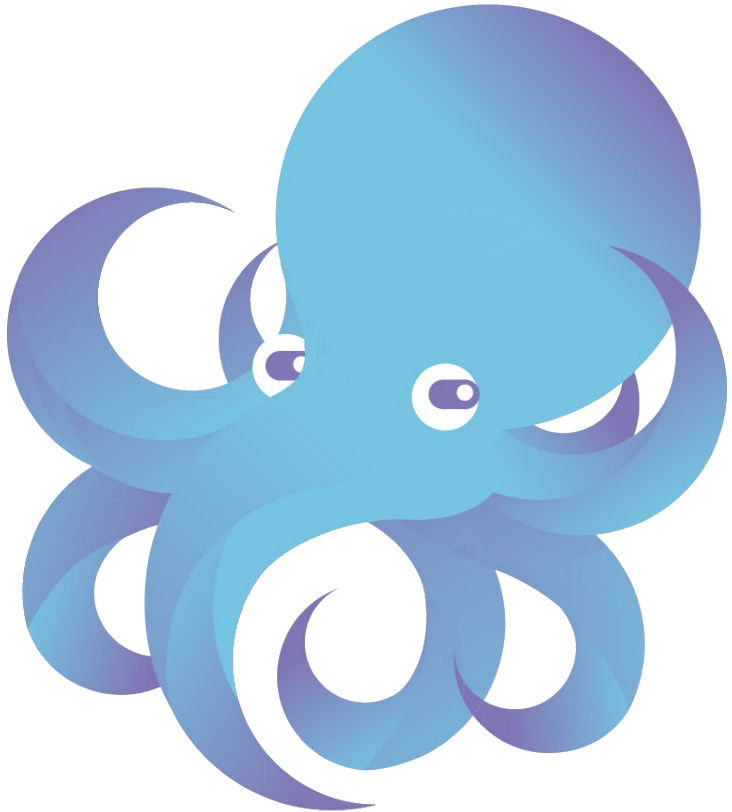



CLOUDFLARE

# Thank you

Questions?

louis@cloudflare.com
@lpoinsig (twitter)



CLOUDFLARE