Automating DNS-01 The Less Lazy Way (with acme-dns)

davie @ LayerOne 2024

Who, me?

Network Engineer

Infosec Adjacent

ShellCon

Reverse Shell Corp

WRCCDC

3D Printing & Homelab

@ me on the LayerOne Discord: @_davie

Twitter: @daschu117

https://shellcon.io/

https://revshellcorp.org/

https://wrccdc.org/

Story Time: There's gotta be a better way

For the longest time, my homelab automated TLS certificates by exposing a NGINX Proxy Manager to the internet.

This was handy for accessing internal services, but it generally seemed like a bad idea.

About 2 years ago, I wanted to get away from this for obvious reasons. All of the automation solutions required a supported DNS provider, but I was using Google Domains that didn't have an API.

Even if it had been supported, the last thing I wanted to do was store my Google credentials unencrypted on disk.

What I needed was a DNS provider with per-record scoped API tokens. It would be a lot of work, but it would be worth it.

By a stroke of luck, I found exactly the solution I was looking for…

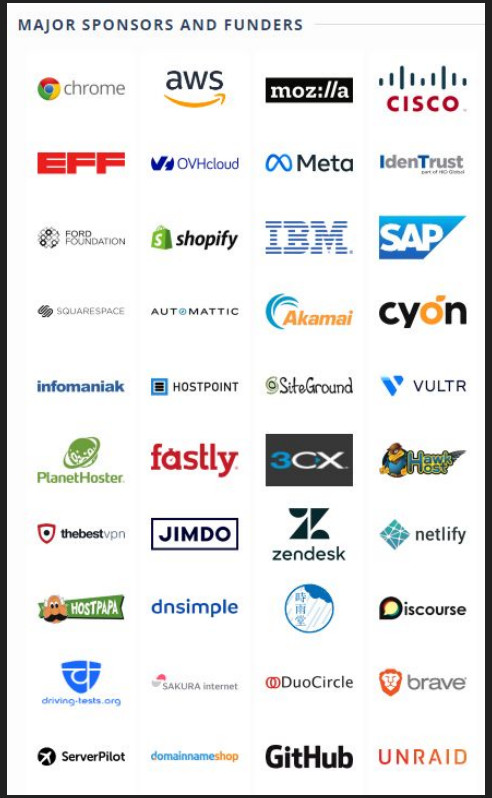
Let’s Encrypt & Internet Security Research Group

- Let's Encrypt is a non-profit Certificate Authority
- Provides free TLS certificates to over 363 million sites
- Founded in 2014
- Run by the Internet Security Research Group (ISRG)
- ISRG founded by Mozilla, the Electronic Frontier Foundation, the University of Michigan, Cisco, and Akamai

Let’s Encrypt

ISRG

https://letsencrypt.org/
https://www.abetterinternet.org/



Let's Encrypt in a nutshell

- Offers “domain validation” certificates based on control over the FQDN of a site or registered domain
- Certificates expire after 90 days to limit exposure of breaches and encourage automation
- Private key is only generated on the client side
- Let's Encrypt never sees the private key
- Was originally cross signed by the CA IdenTrust for widespread compatibility
- Now has its own widely trusted roots, ISRG Root X1 & ISRG Root X2

Vocabulary

ACME: Automated Certificate Management Environment is a protocol to securely validate domain ownership for issuing certificates

ACME Client: A webserver, proxy, loadbalancer, script, or other program that initiates the request and can answer the challenges correctly

ACME Challenge: A random token that must be put into a predetermined spot that proves domain ownership

A record: a DNS record that returns an IP address directly

CNAME record: a DNS record that returns another domain name

TXT record: a DNS record that contains free-form text, but has meaning to certain protocols (ACME, SPF, domain ownership)

NS record: a DNS record to delegate an entire subdomain to a different DNS nameserver

FQDN: Fully Qualified Domain Name; the hostname and domain name and top-level domain of a site or server

ACME Challenge Types

- HTTP-01
  - Easiest method, but only works on public HTTP servers on port 80.
- DNS-01
  - Sit tight, this is why we're here.
- TLS-ALPN-01
  - Requires webserver support for ALPN protocol. Not widely used.

- TLS-SNI-01
  - Not secure and deprecated. Forget I mentioned it.

HTTP-01 Challenge

- Does a DNS lookup of the site's A record
- Connects to your webserver on port 80
- Looks for a special file with a token
  `http://<YOUR_DOMAIN>/.well-known/acme-challenge/<TOKEN>`
- Needs port 80 open to the internet for LE to validate
- Very easy to automate and setup if server is publicly accessible
- Allows you to easily issue certificates for domains that are CNAME'd to a 3rd party web hosting provider
- Can't be used to get a wildcard certificate

DNS-01 Challenge Type

- Requires proof of control over a domain name by editing TXT records
- Need to put the token at: `_acme-challenge.www.example.com`
- Requires manual configuration on your DNS hosting provider
- Can be automated by providing API/Account credentials to your DNS provider
- Can be used to get a wildcard certificate for the entire domain
- Doesn’t require opening any ports to the internet

DNS-01 Automation Drawbacks

- Automated process requires credentials for DNS provider, which could be:
  - Username/password, bypassing 2FA
  - API Token, that may be functionally equivalent to your account password
- Hopefully API token can be limited to ONLY DNS records, or even better, specific DNS zones, or even moar better, specific records
  - But this only applies to providers with strong IAM implementations like AWS, GCP, etc…
- DNS credentials are required to be accessible on disk by the ACME client

Bad DNS Providers

Google Domains: Never had an API, now it's Squarespace. RIP

GoDaddy: Had an API, now it doesn't unless you spend enough money.

Namecheap: One API key for the whole account, can register domains, delete records, add users, change passwords, etc.

Linode: Can be limited to just Domains, but includes all domains and all records.

DigitalOcean: Can be limited to just Domains, but includes all domains and all records.

Cloudflare: Custom API Token can be created that can only edit a specific domain, but can edit any and all records in that domain.

AWS, GCP, etc: Probably similar to Cloudflare, but might be better if you like defining lots of custom roles.

(Almost) Good DNS Providers

Cloudflare Enterprise: Recently announced Foundation DNS that may support "per-record scoped API tokens and user permissions" at an undisclosed point in the future.

https://blog.cloudflare.com/foundation-dns-launch

(I really wish this list was longer, but I literally can't find any)

HTTP-01 vs DNS-01

	HTTP-01	DNS-01
Easy to automate	✅	❌
Doesn't require special support from DNS host	✅	❌
Doesn't require DNS API Credentials on disk	✅	❌
Works with servers not accessible from Internet	❌	✅
Works for services other than HTTP	❌	✅
Wildcard Certificates	❌	✅

Our Requirements

- Limit privileges to only DNS updates
- Limit DNS updates to TXT records only
- Unique credentials per subdomain
- Work with any DNS hosting provider regardless of their API
- Support getting wildcard certificates

acme-dns To The Rescue

https://github.com/joohoi/acme-dns (Joona Hoikkala)

A simplified DNS server written in Go with a RESTful HTTP API to provide a simple way to automate ACME DNS-01 challenges.

- Allows you to register for an anonymous, randomized subdomain
- CNAME the `_acme-challenge.site.domain.tld` to acme-dns
- At certificate request, the ACME client gets the challenge token, updates acme-dns with it
- LE does a look up for the domain, and follows the CNAME to the acme-dns server to get the challenge response

How to use acme-dns

```
# curl -sX POST https://auth.acme-dns.io/register | jq
{
  "username": "340ba7ec-7488-4497-a399-f4d76de60650",
  "password": "ATzMzZXSw6zduf6jQG2kWqpOF2Vx8_iflnei7RV9",
  "fulldomain": "e6896e5d-5b8e-44a9-87bc-1e0a16001c55.auth.acme-dns.io",
  "subdomain": "e6896e5d-5b8e-44a9-87bc-1e0a16001c55",
  "allowfrom": []
}
```

Setting up the CNAME DNS record

Point `_acme-challenge.www.example.com` to acme-dns "fulldomain" value

```
_acme-challenge.www.example.com. CNAME
xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx.auth.acme-dns.io.
```

The trailing . on these records is generally important, depending on your DNS provider

How acme-dns improves security

- The credentials configured in the ACME client are NOT account-equivalent
- The credentials don’t even provide control of the entire domain in your DNS Provider account
- The ACME client can only validate domains that you configure by CNAMEing to the acme-dns provided subdomain
- This allows delegating some domain records to one server, and other domain records to a different server, all without providing non-DNS access to your account
- The acme-dns credentials are only for updating the service’s specific subdomains, not anything like A, CNAME, or MX records on your domain
- Doesn’t require any internet facing ports (unless you run your own service)
- DNS admins can delegate automation to other teams

How to use acme-dns

Setup your own service

- https://github.com/joohoi/acme-dns
- https://hub.docker.com/r/joohoi/acme-dns/

Use a public service (maybe not the best idea, but I'm sure it's fine)

- `https://auth.acme-dns.io/` (Joona Hoikkala)
- `https://api.getlocalcert.net/api/v1/acme-dns-compat`

Compatible Clients

- acme.sh: https://github.com/acmesh-official/acme.sh
- Traefik: https://github.com/traefik/traefik

- Certify The Web
- cert-manager
- Lego
- Posh-ACME
- Sewer
- Windows ACME Simple (WACS)

- Certbot (with authentication hook)
- Caddy (with extra module)

Story Time: F*$% GoDaddy

Around the beginning of May 2024, GoDaddy changed their API with no announcement. Now they only allow customers with a monthly subscription or more than 10 domains to use the DNS API to update records.

This caught their customers unaware, causing confusion and anger. Seems like lots of people jumping ship to Cloudflare.

While migrating between DNS providers doesn't usually hurt too much due being able to import records, updating your DNS-01 automations across however many servers definitely does suck.

Using acme-dns to decouple your DNS hosting from certificate automation additionally makes DNS migrations simpler because you're changing less at once.

TL;DR: GoDaddy is a shitty company that doesn't deserve your, my, or anyone else's business.

Conclusion

In my opinion, there is only ONE way to properly automate DNS-01 challenges.

And that way is acme-dns 🥳

Additional resources:

Joona Hoikkala (@joohoi):
https://www.eff.org/deeplinks/2018/02/technical-deep-dive-securing-automation-acme-dns-challenge-validation
https://www.youtube.com/watch?v=Xu6SQLZ7gm8

Thank you!

https://github.com/daschu117/talk-acmedns

@_davie on the LayerOne Discord

Twitter: @daschu117