CEN
Connecticut's Trusted Internet Partner

Let's Encrypt: Introduction and Background

CEN CONNECT
2026 Member Conference

About Me

Jason Healy

Suffield Academy

- 420 Students
- High School (grades 9-12)
- Boarding and Day students

Director of Technology:

- Manage IT department
- Core network and wireless
- Teach Computer Science

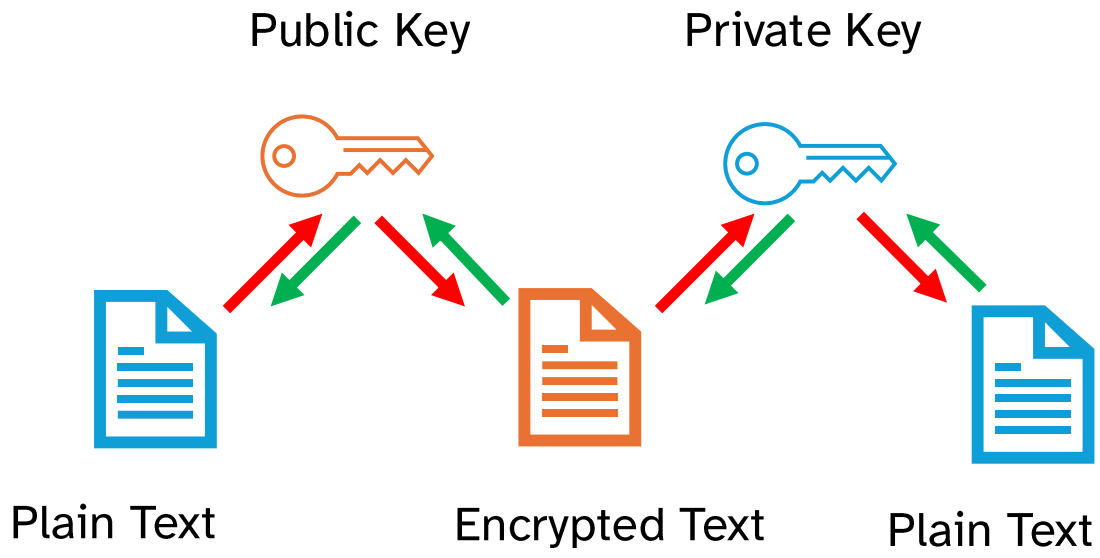
Motivation

Why We Need Encryption

- HTTP is plain-text
- Data are not protected against snooping/spoofing
- Many web browsers will warn users about plain HTTP

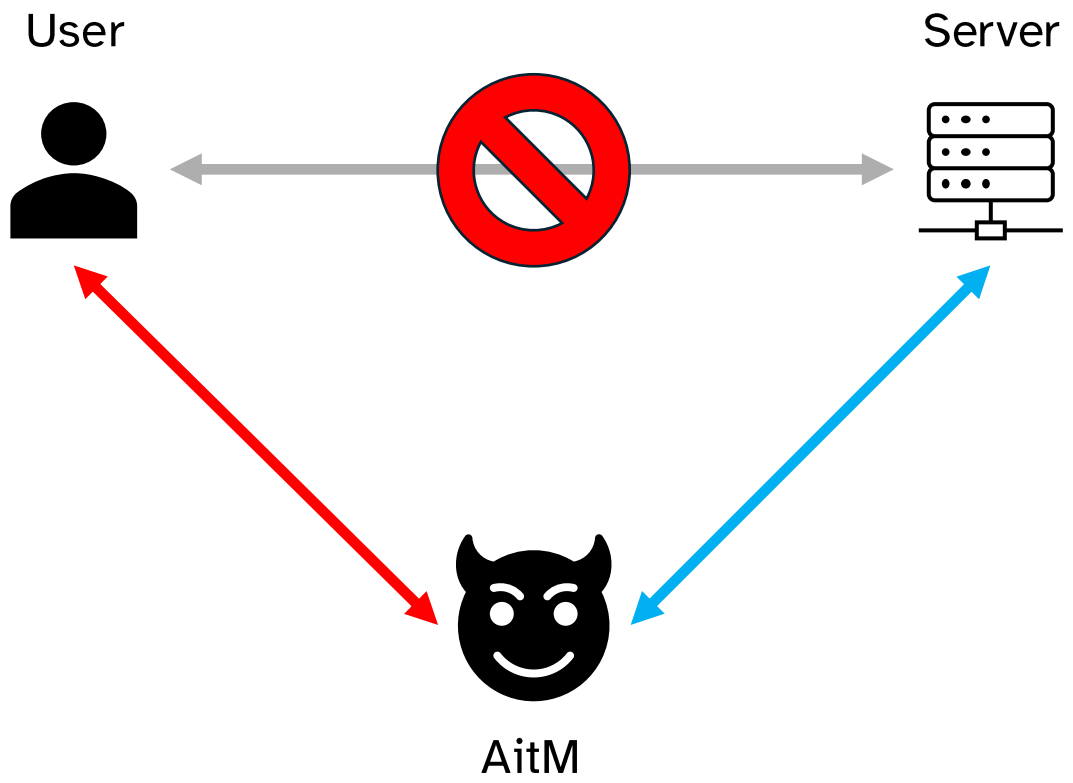
How TLS Encryption Works

- Use Public Key Cryptography
- Server sends a public key to clients
- Client can use this key to encrypt data that only the server can read
- Client can verify that data sent from server hasn't been tampered with
- TLS embeds the public key in a **certificate**



Why We Need Certificate Authorities

- Adversary-in-the-Middle Attack
- Adversary captures all traffic from the user, terminates session locally
- Opens a separate session to the server
- To defeat this attack, client needs to be able to verify that the public key used in the connection actually belongs to the server.
- A Certificate Authority (CA) performs this verification



Why We Need Let's Encrypt

- Most CAs require payments to "vouch" for your TLS cert
- EFF wanted to encourage TLS usage globally (post-Snowden)
- Let's Encrypt was founded as a free CA so anyone can sign certs
- Service is automated so it is cheap(er) to run (and easier to use!)

https://xkcd.com/364

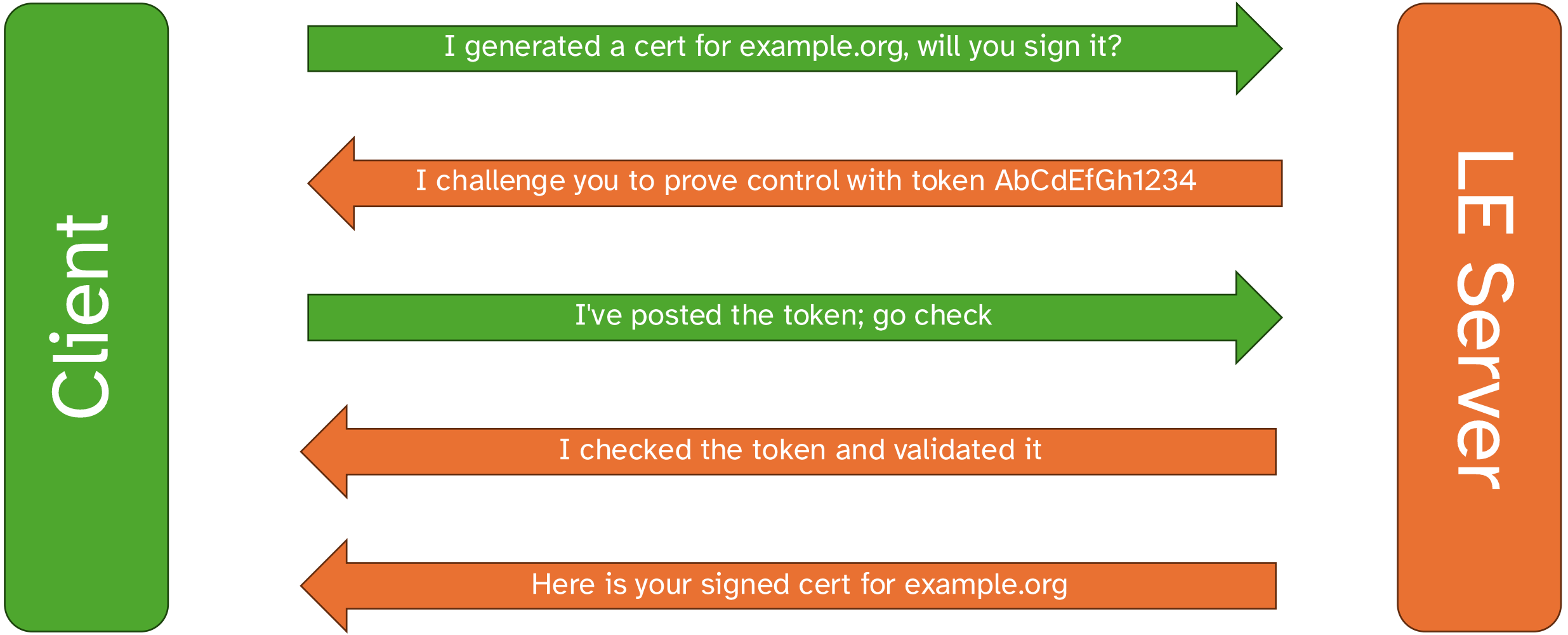
Protocol Overview

Automated Certificate Management Environment (ACME)

ACME Overview

- Let's Encrypt implements an **API** (ACME, RFC 8555)
- Users make an account to interact with the ACME servers
- The ACME servers issue **challenges**, **validate** them, and then **sign** certificates
- Challenges prove control/ownership of domain
- There are multiple challenge types:
  - **HTTP-01**: Serve a token via HTTP
  - **DNS-01**: Serve a token via DNS configuration
  - **DNS-PERSIST-01**: Grant blanket permission in DNS
  - **TLS-APLN-01**: Modify TLS handshake (experts only!)

ACME API Protocol Diagram



Client Setup

Client Selection

- ACME is just an API
- Multiple client implementations exist: https://letsencrypt.org/docs/client-options/
  - Windows / Linux
  - Kubernetes / Docker
  - Embedded in web server
  - C / C++ / Rust / Java / Bash / PHP / Perl …
- We will use Certbot for this talk (well-supported, good features)

Certbot Installation

Go to https://certbot.eff.org and choose your Software/OS combo:

My HTTP website is running Software **on** System

- ✓ Software
- Apache
- Nginx
- HAProxy
- Plesk
- Other
- Web Hosting Product

Help, I'm not sure!

My HTTP website is running Software **on** System

- ✓ System
- Bitnami
- FreeBSD
- Windows
- OpenBSD
- macOS
- Web Hosting Service
- Linux (snap)
- Linux (pip)

I'm not sure!

Certbot First Run: Auth Type

```
# certbot certonly
Saving debug log to /var/log/letsencrypt/letsencrypt.log

How would you like to authenticate with the ACME CA?
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1: Runs an HTTP server locally which serves the necessary validation files under
the /.well-known/acme-challenge/ request path. Suitable if there is no HTTP
server already running. HTTP challenge only (wildcards not supported).
(standalone)
2: Saves the necessary validation files to a .well-known/acme-challenge/
directory within the nominated webroot path. A separate HTTP server must be
running and serving files from the webroot path. HTTP challenge only (wildcards
not supported). (webroot)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Select the appropriate number [1-2] then [enter] (press 'c' to cancel): 2
```

Certbot First Run: Email and Terms

```
Enter email address or hit Enter to skip.
 (Enter 'c' to cancel): root@example.org

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Please read the Terms of Service at:
data:text/plain,Do%20what%20thou%20wilt

You must agree in order to register with the ACME server. Do you
agree?

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

(Y)es/(N)o: Y
```

Certbot First Run: Email Preference

- -

Would you be willing, once your first certificate is successfully issued, to share your email address with the Electronic Frontier Foundation, a founding partner of the Let's Encrypt project and the non-profit organization that develops Certbot? We'd like to send you email about our work encrypting the web, EFF news, campaigns, and ways to support digital freedom.

- -

(Y)es/(N)o: N

Account registered.

Certbot First Run: Cert Name and Root

```
Please enter the domain name(s) you would like on your
certificate (comma and/or space separated):
www.example.org example.org
Requesting a certificate for www.example.org and example.org

Input the webroot for example.org: /var/www

Select the webroot for www.example.org:
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1: Enter a new webroot
2: /var/www
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Select the appropriate number [1-2] then [enter]: 2
```

Certbot First Run: Complete

```
Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/www.example.org/fullchain.pem
Key is saved at:         /etc/letsencrypt/live/www.example.org/privkey.pem
This certificate expires on 2026-07-17.
These files will be updated when the certificate renews.


NEXT STEPS:
- The certificate will need to be renewed before it expires. Certbot can automatically
renew the certificate in the background, but you may need to take steps to enable that
functionality. See https://certbot.org/renewal-setup for instructions.


- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
If you like Certbot, please consider supporting our work by:
 * Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
 * Donating to EFF:                    https://eff.org/donate-le
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Certbot Renewal

- Once certificate is issued, we can automatically renew
- Default is to renew all certificates
- Renewal automatically checks date; safe to run "early"

```
# certbot renew
Saving debug log to /var/log/letsencrypt/letsencrypt.log

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Processing /etc/letsencrypt/renewal/www.example.org.conf
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Renewing an existing certificate for www.example.org and example.org

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Congratulations, all renewals succeeded:
  /etc/letsencrypt/live/www.example.org/fullchain.pem (success)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Certbot Hooks

- Certbot contains "hooks" that allow additional commands to be run at different points in the renewal process:

```
certbot renew --pre-hook <command> \
              --post-hook <command> \
              --deploy-hook <command> \
              --manual-auth-hook <command> \
              --manual-cleanup-hook <command>
```

- "deploy" is after each cert renewal
- "pre"/"post" are global before/after full run
- Hooks are saved to configuration after a successful run

Certbot Hook Examples

```
# certbot renew --post-hook \
      "systemctl restart postfix"
```

```
# certbot renew --deploy-hook \
      "nginx -s reload"
```

```
C:\> certbot renew --post-hook ^
      "iisreset /restart"
```

```
PS > certbot renew --post-hook `
      "powershell -NoProfile
         -ExecutionPolicy Bypass
         -Command iisreset /restart"
```

Certbot Automation

We now have:

- Single-command issuance
- Single-command renewal
- Deployment hooks for service refresh

Automation simply requires running the renewal command on a periodic schedule:

- Can use cron on Unix-like systems
- Windows certbot includes a scheduled renewal

Beyond the Basics

Staging Environment

- Let's Encrypt supports a staging environment
- Test workflow without issuing real certs
- Avoid hitting API limits when testing repeatedly
- Accessed using CLI switches in most clients, e.g.:

```
certbot --dry-run
certbot --test-cert
```

HTTP-01

- So far we've talked about HTTP challenges:

```
http://example.org/.well-known/acme-challenge/{token}
```

- Prove control of exact hostname(s)
- If you're already running a web server, no other software or access is needed
- Decentralized (run locally on each server)
- Easy to automate
- **Will not work for wildcards** (*.example.org)

DNS-01

- Assumes that if you control DNS, you effectively control the name the server is using (access to server itself not needed)
- Writes challenge token to DNS entry based on server name:

```
_acme-challenge.example.org. 300 IN TXT "{token}"
_acme-challenge.www.example.org. 300 IN TXT "{token}"
```

- Allows generation of wildcards (*.example.org)
- Can be centralized (one machine makes all cert requests)
- Requires access to DNS infrastructure (API/TSIG)

DNS-01 (Manual)

```
# certbot certonly --manual --preferred-challenges dns-01 -d '*.example.org'
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Requesting a certificate for *.example.org

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Please deploy a DNS TXT record under the name:

_acme-challenge.example.org.

with the following value:

Zz6Ka7RSjeSvmUh6Do17v6C31aByyI6hTA8Pu0xh2Kg

Before continuing, verify the TXT record has been deployed. Depending on the
DNS provider, this may take some time, from a few seconds to multiple minutes.
...

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Press Enter to Continue
```

DNS-01 Automation

- Don't want to have to manually update DNS for each host name every time we renew.
- Certbot has plugins to automatically update DNS for certain large providers:

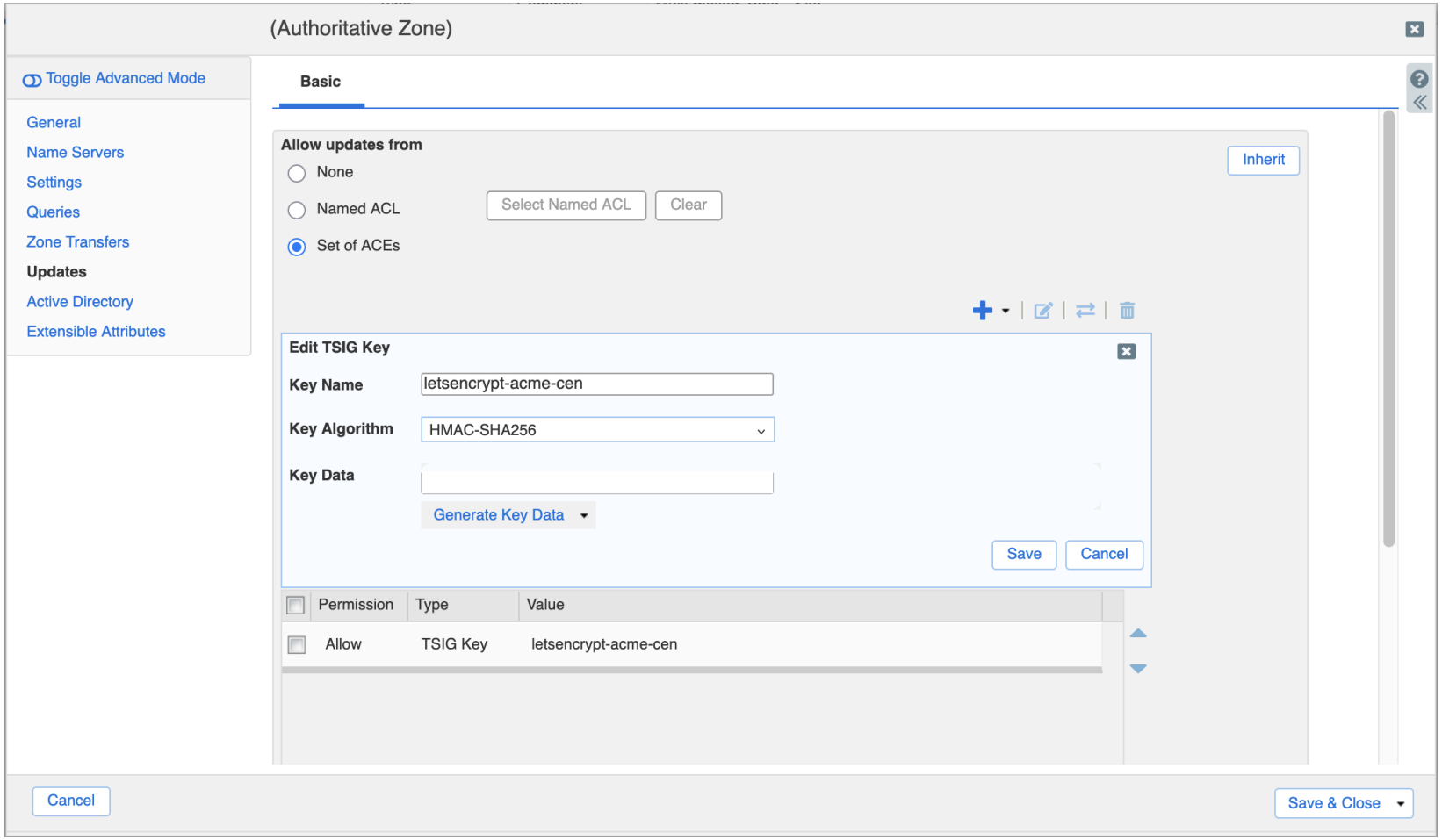
  https://eff-certbot.readthedocs.io/en/latest/using.html#dns-plugins
- They also have an RFC 2136 plugin, which works for standards-compliant DNS (including CEN!)

  https://certbot-dns-rfc2136.readthedocs.io/en/stable/
- Certbot will automatically update DNS as part of the challenge response, allowing renewals to be fully automated.

DNS-01 (RFC 2136 Key Creation)

Create a TSIG Key:

```
tsig-keygen -a HMAC-SHA512 keyname.
```



DNS-01 (RFC 2136)

```
# rfc2136.ini
dns_rfc2136_server = 2001:db8::53:0
dns_rfc2136_port = 53
dns_rfc2136_name = keyname.
dns_rfc2136_secret = WW91IG5lZWQgdG8gZ2V0IGEgbGlmZSE=
dns_rfc2136_algorithm = HMAC-SHA512
```

```
# certbot certonly \
  --dns-rfc2136 \
  --dns-rfc2136-credentials ~/mumble/rfc2136.ini \
  --dns-rfc2136-propagation-seconds 30 \
  -d example.org
```

DNS-01 Concerns

Granting full access to your DNS is scary...

- Most DNS servers do not allow restricting operations to a specific pattern (`_acme_challenge`)
- Many won't allow granular rules by type
- Even type restrictions (TXT) are too broad (can affect SPF, etc)

Most common solution is to redirect queries to a sandbox domain:

- NS Delegation / ACME-DNS (self-hosted)
- Alias domain (CNAME)

Unfortunately, you still must manually set up each domain on first use.

DNS-01 (Delegation)

Set up an alternate DNS server under your control (either standard DNS or ACME DNS).

Delegate the "real" challenge to the alternate server (must do this for every domain you wish to respond to challenges for):

```
_acme_challenge.www.example.org. IN NS my-alt-server.example.org.
```

Grant full permissions to the script on the alternate nameserver.

Let's Encrypt will chase the NS delegation to your alternate server.

ACME DNS is a special-purpose DNS server that has an HTTP API to directly create records without RFC 2136.

DNS-01 (CNAME Redirect)

Set up a sandbox domain or subdomain: `sandbox.example.org`

Establish a CNAME from the actual challenge to the sandbox:

```
_acme_challenge.www.example.org. IN CNAME
                    www.example.org.sandbox.example.org.
```

Now you only need to give write permissions to the sandbox. Because it's not used by the public internet, less damage if the sandbox is compromised.

Your update script must know how to update the sandbox instead of the actual domain.

DNS-01 (Manual Auth Hook)

```
#!/bin/bash
set -e
DNS_SERVER="2001:db8::53:0"
ZONE="sandbox.example.org"
TSIG_KEY_NAME="keyname."
TSIG_KEY_SECRET="WW91IG5lZWQgdG8gZ2V0IGEgbGlmZSE="

nsupdate -y "${TSIG_KEY_NAME}:${TSIG_KEY_SECRET}" <<EOF
server ${DNS_SERVER}
zone ${ZONE}
update delete ${CERTBOT_IDENTIFIER} TXT
update add ${CERTBOT_IDENTIFIER} 60 TXT "${CERTBOT_VALIDATION}"
send
EOF

sleep 20
```

DNS-01 (Manual Auth)

```
# certbot certonly --manual --preferred-challenges=dns \
           --manual-auth-hook /mumble/my_auth_hook.sh \
           -d www.example.org
```

Certbot will use our script instead of the built-in plugins to set the challenge response.

This works around missing functionality in certbot.

Some clients (uacme, acme.sh) support aliasing domains natively.

DNS-PERSIST-01 Challenges

- Coming in 2026? draft-ietf-acme-dns-persist
- Uses single DNS record to grant authority to a LE account:

```
_validation-persist.example.org. IN TXT
      "letsencrypt.org;"
      " accounturi=https://acme-v02.api.letsencrypt.org/acme/acct/123…890"
      " policy=wildcard"
```

- Do not need to perform live DNS updates, nor establish records for every domain name
- Configuration is available to limit scope of accounts and changes
- Consider planning for use if your DNS is tricky to update

Wrapping Up

I hope this has given you a good starting point!

As you think about your own implementation, consider:

- Do I want centralized or per-server cert management?
- Do I need wildcards (and thus DNS auth)?
- Do I have a homogenous or heterogenous set of servers?
- What software platform(s) will I integrate with?
- What operating system(s) will I run on?
- If using DNS-01, what API will I use for updates?
- Am I a candidate for DNS-PERSIST-01?

Q & A

We greatly appreciate your time today and continued support!

Please take a moment to rate this session in the *CEN Events* mobile app.