Let’s Encrypt:

Automate all the things

Philip Paeps

SANOG 40 ~~tutorial~~ presentation

FreeBSD

Getting a certificate: common process

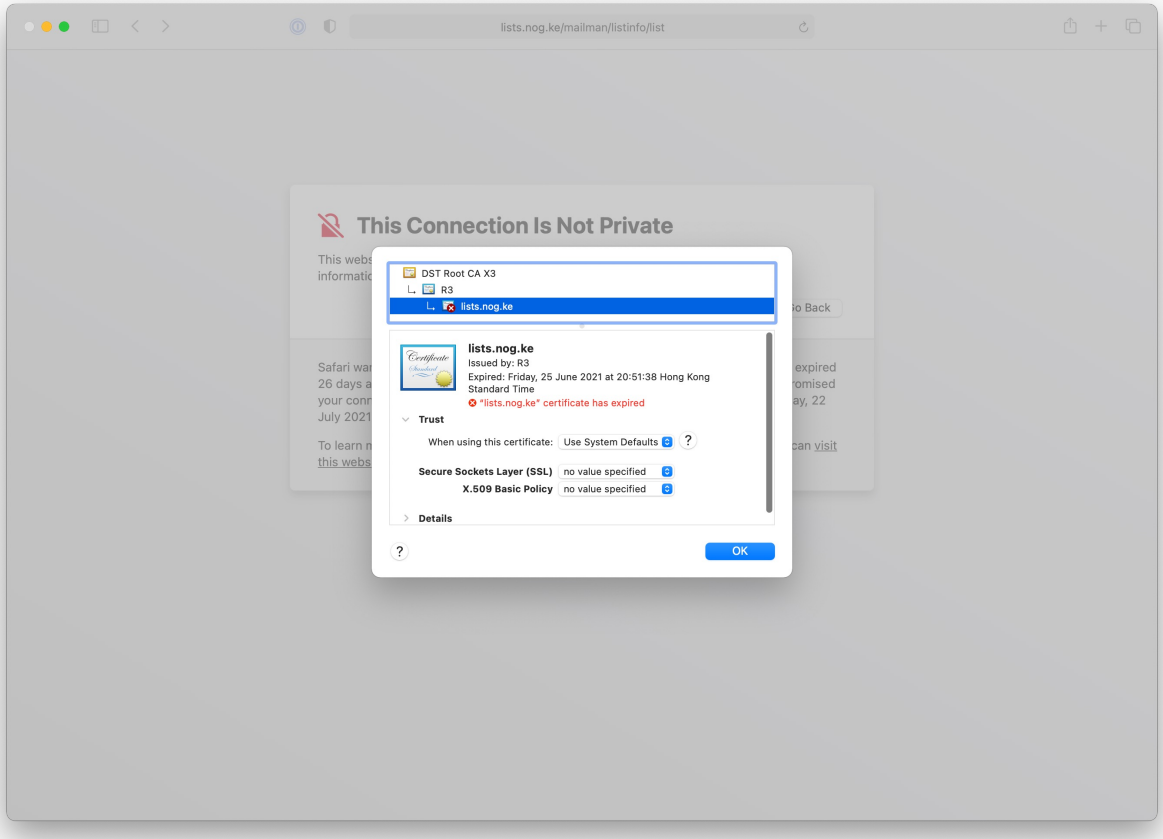
1. Generate certificate
2. Build certificate signing request (CSR)
3. Convince certification authority (CA) to sign certificate
4. Repeat annually

Certificate authorities

A fundamentally flawed system

- CAs are answerable first to their shareholders. Their primary objective is profit. Improving security on the internet is a secondary concern.
- Anything that happens *annually*, tends to happen *manually*.
  - Error prone
  - Easy to forget

A common example…



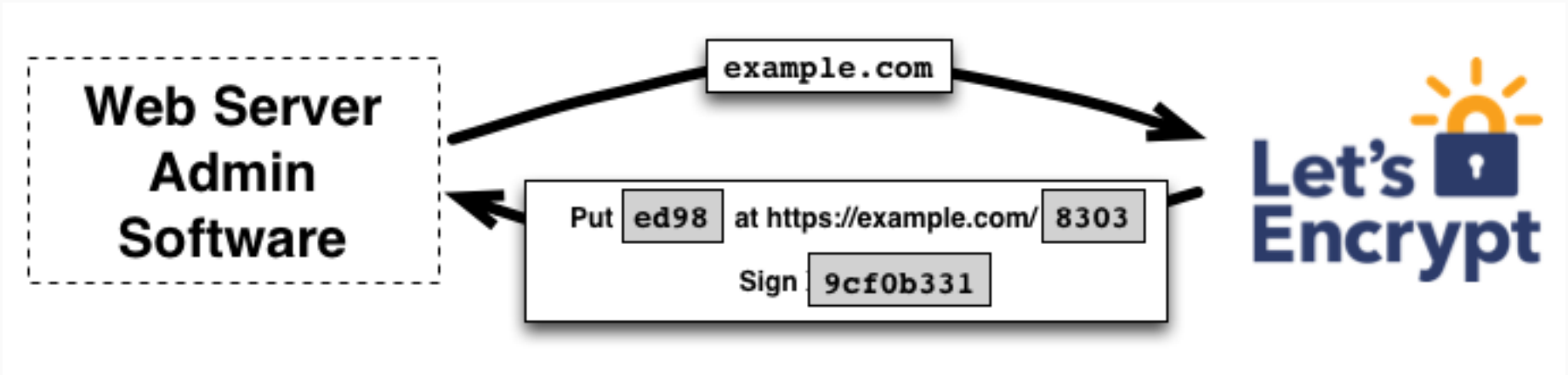
Let’s Encrypt: a not-for-profit CA

- Free
- Automatic
- Secure
- Transparent
- Open
- Cooperative

ZeroSSL: an alternative

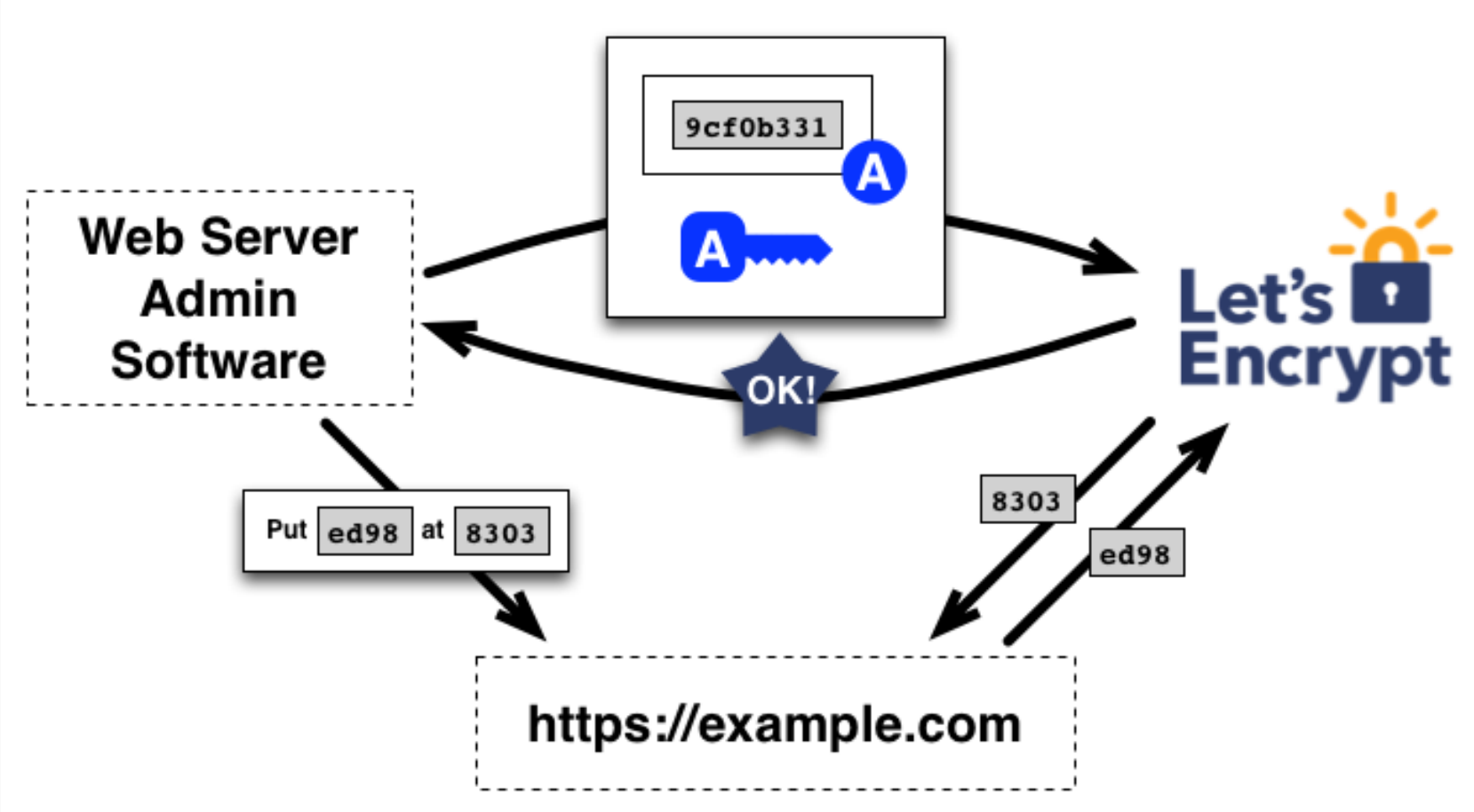
- Free plan substantially identical to Let's Encrypt
  - Unlimited ACME certificates
- More expensive plans available with a range of features
  - Monitoring
  - 1-year certificate validity
  - APIs
  - Technical support

How does this work?



FreeBSD

How does this work?



FreeBSD

Under the hood: ACME

- RFC 8555: Automated Certificate Management Environment
- JSON-based protocol over HTTPS
- Several implementations

ACME implementations

certbot

- Reference implementation
- Many dependencies
- Many assumptions

acme.sh

- Tiny shell script
- No awkward dependencies

Many many other clients exist:

https://letsencrypt.org/docs/client-options/

Lists over a hundred (as of November 2020)

acme.sh in a nutshell

Installation

The installer does 3 things:

1. Create and copy acme.sh in $HOME/.acme.sh
2. Alias for acme.sh
3. Daily cronjob to check and renew certificates

```
$ git clone \
https://github.com/acmesh-official/acme.sh.git
$ cd ./acme.sh
$ ./acme.sh --install

OR

$ curl https://get.acme.sh | sh
```

FreeBSD

acme.sh in a nutshell

Issue a certificate

Single domain:

```
$ acme.sh --issue -d example.com -w /home/wwwroot/example.com
$ acme.sh --issue -d example.com -w /home/username/public_html
$ acme.sh --issue -d example.com -w /var/www/html
```

Multiple domains:

```
$ acme.sh –issue                \
  -d example.com                \
  -d www.example.com            \
  -d cp.example.com                     \
  -w /home/wwwroot/example.com
```

Many many options

Opportunities for automation

- Standalone mode
  - No webserver needed
  - Requires privilege to bind to port 80
  - Still requires publicly reachable machine
- DNS mode
  - This is where the real magic is!

Rate limits

A brief digression

- Limits are generous but firm
- Moral of the story: don’t test in production
- Key limits of Let’s Encrypt:
  - Certificates per registered domain: 50 per week
    - Renewals don’t count
  - Duplicate certificates: 5 per week
  - Failed validation: 5 per account, per hostname, per hour
- More limits on https://letsencrypt.org/docs/rate-limits/

DNS mode

Demo

```
$ acme.sh --issue -k 4096 --dns dns_azure --challenge-alias certbot.trouble.is -d pacnog27.trouble.is
[Mon Nov 30 07:45:43 UTC 2020] Creating domain key
[Mon Nov 30 07:45:45 UTC 2020] The domain key is here: /home/certbot/.acme.sh/pacnog27.trouble.is/pacnog27.trouble.is.key
[Mon Nov 30 07:45:45 UTC 2020] Single domain='pacnog27.trouble.is'
[Mon Nov 30 07:45:45 UTC 2020] Getting domain auth token for each domain
[Mon Nov 30 07:45:47 UTC 2020] Getting webroot for domain='pacnog27.trouble.is'
[Mon Nov 30 07:45:47 UTC 2020] Adding txt value: QweUWz4_efyxJM9EfTlBQr9D11Kj0WJ_7DBpx19jH7g for domain:  _acme-challenge.certbot.trouble.is
[Mon Nov 30 07:45:52 UTC 2020] validation value added
[Mon Nov 30 07:45:52 UTC 2020] The txt record is added: Success.
[Mon Nov 30 07:45:52 UTC 2020] Let's check each dns records now. Sleep 20 seconds first.
[Mon Nov 30 07:46:14 UTC 2020] Checking pacnog27.trouble.is for _acme-challenge.certbot.trouble.is
[Mon Nov 30 07:46:14 UTC 2020] Domain pacnog27.trouble.is '_acme-challenge.certbot.trouble.is' success.
[Mon Nov 30 07:46:14 UTC 2020] All success, let's return
[Mon Nov 30 07:46:14 UTC 2020] Verifying: pacnog27.trouble.is
[Mon Nov 30 07:46:18 UTC 2020] Success
[Mon Nov 30 07:46:18 UTC 2020] Removing DNS records.
[Mon Nov 30 07:46:18 UTC 2020] Removing txt: QweUWz4_efyxJM9EfTlBQr9D11Kj0WJ_7DBpx19jH7g for domain: _acme-challenge.certbot.trouble.is
[Mon Nov 30 07:46:22 UTC 2020] validation record removed
[Mon Nov 30 07:46:22 UTC 2020] Removed: Success
[Mon Nov 30 07:46:22 UTC 2020] Verify finished, start to sign.
[Mon Nov 30 07:46:22 UTC 2020] Lets finalize the order, Le_OrderFinalize: https://acme-v02.api.letsencrypt.org/acme/finalize/67909898/6485905412
[Mon Nov 30 07:46:23 UTC 2020] Download cert, Le_LinkCert: https://acme-v02.api.letsencrypt.org/acme/cert/03896bb36534a837d3d47949ea2485987399
[Mon Nov 30 07:46:24 UTC 2020] Cert success.
-----BEGIN CERTIFICATE-----
[...]
-----END CERTIFICATE-----
[Mon Nov 30 07:46:24 UTC 2020] Your cert is in  /home/certbot/.acme.sh/pacnog27.trouble.is/pacnog27.trouble.is.cer
[Mon Nov 30 07:46:24 UTC 2020] Your cert key is in  /home/certbot/.acme.sh/pacnog27.trouble.is/pacnog27.trouble.is.key
[Mon Nov 30 07:46:24 UTC 2020] The intermediate CA cert is in  /home/certbot/.acme.sh/pacnog27.trouble.is/ca.cer
[Mon Nov 30 07:46:24 UTC 2020] And the full chain certs is there:  /home/certbot/.acme.sh/pacnog27.trouble.is/fullchain.cer
```

DNS mode

How does this work

- Verify ownership of domain by checking for TXT record
- Indirection: CNAME verification domain to Azure
  - For automation: Azure API is easy to use
  - For security: don’t expose API access to domain’s DNS
  - Could use ns-update to self-host, but Azure is easier
  - Could use any number of other DNS providers
- acme.sh will renew the certificate regularly (cron)

DNS hooks, deploy hooks…

A look under the hood

- Everything about acme.sh can be customised
- Hooks are simple shell scripts
- Plenty of examples in the repository

DNS mode

Using Azure

- Create a resource group
- Create a DNS zone
- Create a Service Principal with permission to modify the zone
- Add to acme.sh user’s environment
- Profit!

DNS mode

Using Azure (Cloud Shell)

```
PS /home/philip> $scope=az network dns zone list
  --resource-group kenog -o json --query '[0].id'
PS /home/philip> az ad sp create-for-rbac
  --name "KENOGAcmeDnsValidator"
  --role "DNS Zone Contributor"
  --scopes $scope
Creating 'DNS Zone Contributor' role assignment under scope '/subscriptions
/63271ae2-d0af-41d4-93d4-5059d4b80393/resourceGroups/kenog/providers/Microsof
t.Network/dnszones/kenog.trouble.is'
The output includes credentials that you must protect. Be sure that you do not
include these credentials in your code or check the credentials into your source
control. For more information, see https://aka.ms/azadsp-cli
'name' property in the output is deprecated and will be removed in the future.
Use 'appId' instead.
{
  "appId": "4131fea3-9749-455c-bd69-3e835e479f5b",
  "displayName": "KENOGAcmeDnsValidator",
  "name": "4131fea3-9749-455c-bd69-3e835e479f5b",
  "password": "cNML.zeI6b3-JDE7BwJVAKGO2ExBqQcDj6",
  "tenant": "76b15542-8c75-4020-b9b6-fe3d391c1aa4"
}
```

DNS mode

Using Cloudflare (or other services)

- Get an API token
- Restrict access as much as possible
- Set some environment variables
- Profit!

DNS mode

How does this work

```
trouble.is DNS:

; certbot keeps LetsEncrypt.org certificates fresh.
;   acme.sh on rincewind.trouble.is manages LetsEncrypt.org challenges
;   using the Azure API.  Domains we manage need a CNAME pointing
;   _acme-challenge.DOMAIN. to _acme-challenge.certbot.trouble.is.
certbot         NS      ns1-04.azure-dns.com.
                NS      ns2-04.azure-dns.net.
                NS      ns3-04.azure-dns.org.
                NS      ns4-04.azure-dns.info.

[...]

; PacNOG 27 Let's Encrypt talk
pacnog27A               127.0.0.1
                        AAAA    ::1
                        MX 0    .
_acme-challenge.pacnog27    CNAME _acme-challenge.certbot
```

The end

- Pointers to more resources:
  https://letsencrypt.org
  https://github.com/acmesh-official/acme.sh/wiki

- Contact me: philip@trouble.is