Encrypt ALL the things with **Let’s Encrypt**

Created by:

➔ Justin W. Flory

➔ Solomon Rubin

Introduction

What is SSL and why do I need it?

- SSL stands for Secure Sockets Layer
  - Difference between https and http
  - Encrypts your communications with a website on the fly
- Normally you need to purchase a SSL certificate from a **Certificate Authority**
  - Sometimes pricey, especially if you have multiple subdomains too
  - Let’s Encrypt offers a solution to this problem to help increase the overall security of the web
- Imagine a world where encryption is everywhere and your online communications are always secure (lol)

What is LetsEncrypt?!

- Problems with certificate issuance
  - Basic encryption is expensive
  - Most certificate authorities (CAs) focus on **identity** or **organization verification**
  - Most sites only need **domain verification**
- Free certificates
  - Providing only domain verification
    - At zero cost
  - To create a safer web

Key Principles

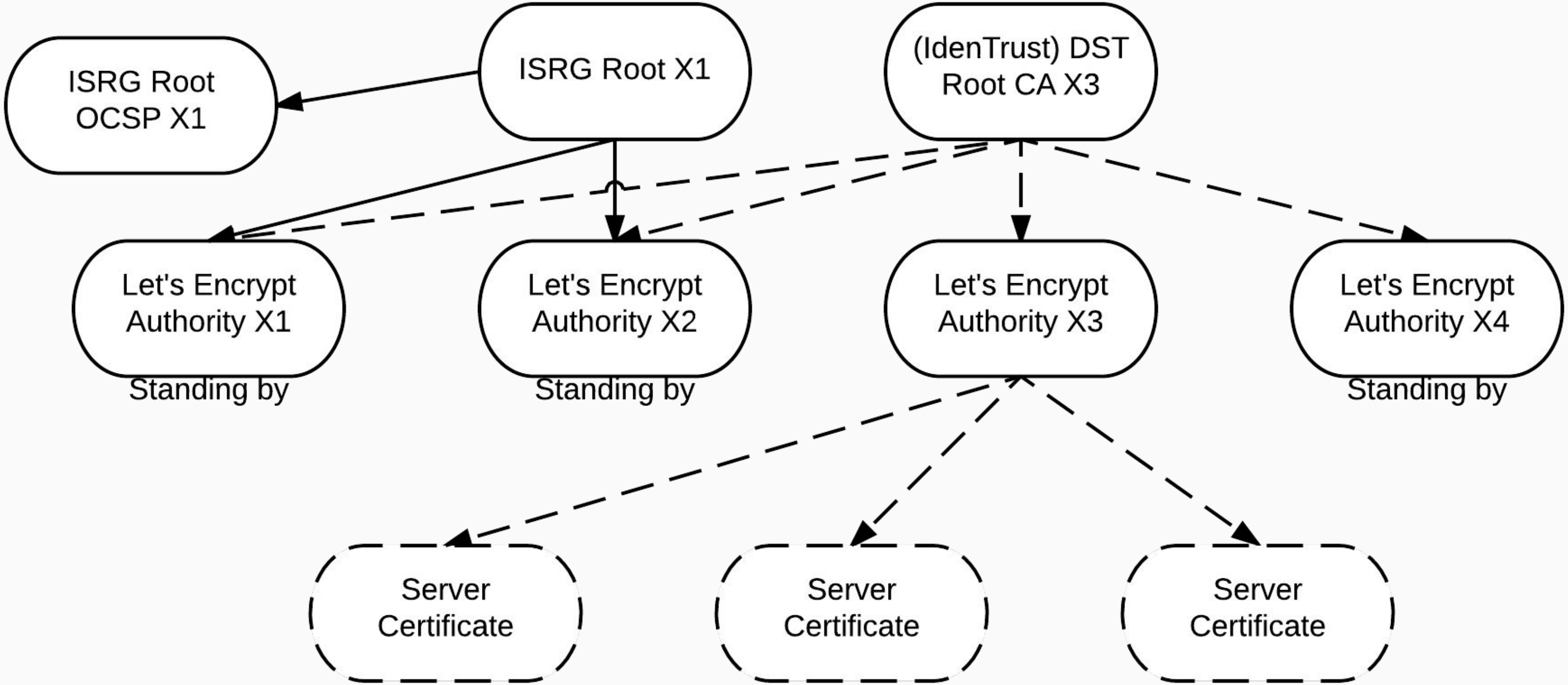
- **Free** for anyone who owns a domain
- **Automatic** cert issuance through client software located on the web-server
- **Secure**: “LE will serve as a platform for advancing TLS security...”
- **Transparent**: All certs issued and revoked get publicly logged
- **Open**: Cert management process is published as open source software.
- **Cooperative**: LE is a joint effort between multiple organizations and the community!

Who made this happen?
I want to see the proof!

- Linux Foundation
- Sponsored by many large organizations
    - Mozilla
    - Cisco
    - Facebook
    - IdenTrust
    - Electronic Frontier Foundation
    - Hewlett Packard
    - Many more

How does it work (Root Cert Propagation)

- LE Root Certificate (ISRG Root 1X)
    - Kept safely offline.
    - Propagated through Intermediates
- LE Intermediate Certificates (All IdentTrust cross-signed)
    - X1, X2 - Original Intermediates
    - X3 - Current generation Intermediate
    - X4 - Disaster Recovery Intermediate


ISRG Root OCSP X1
ISRG Root X1
(IdenTrust) DST Root CA X3
Let's Encrypt Authority X1
Standing by
Let's Encrypt Authority X2
Standing by
Let's Encrypt Authority X3
Let's Encrypt Authority X4
Standing by
Server Certificate
Server Certificate
Server Certificate

Crazy Diagram!

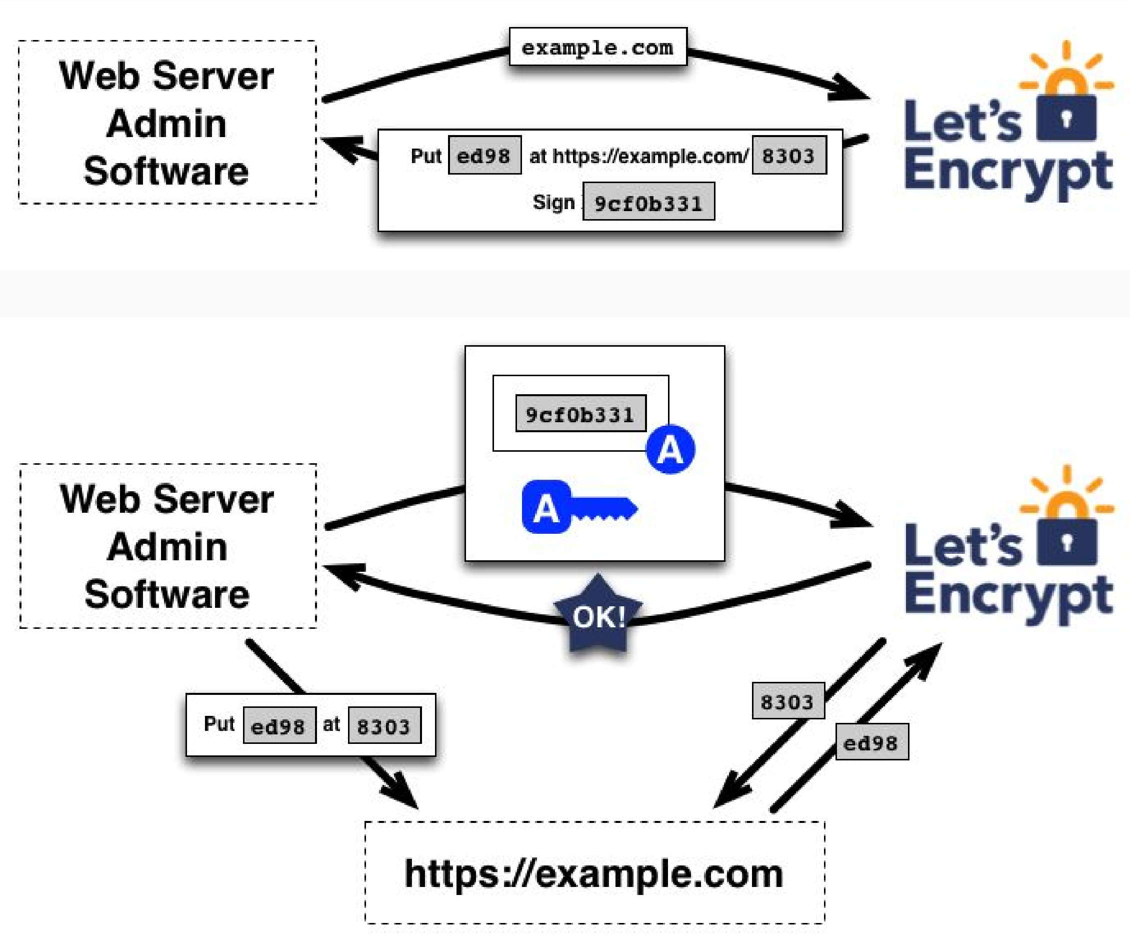
How does it work? (Domain Verification)

- Automatic DNS based verification
- Three Methods
  - Apache, Webroot, Standalone
  - NginX (experimental)
- Uses URL/Key Pairs

Verification Process

Challenge Sets

- Adding a key to a specific, random url
- Verify from LE servers



Getting your certificates

Installation

- Nowadays, available in most Linux distribution package managers
  - If not, it is still possible to compile from source and run it (it is all Python under the hood!)
- Debian / Ubuntu / Debian-based distributions

  -
    ```
    $ sudo apt-get install letsencrypt
    ```
- Red Hat Enterprise Linux / CentOS (via EPEL)

  -
    ```
    $ sudo yum install letsencrypt
    ```
- Fedora

  -
    ```
    $ sudo dnf install letsencrypt
    ```
- Arch Linux

  -
    ```
    $ sudo pacman -S letsencrypt
    ```

Issuing Certificates via standalone

- Standalone uses port 80 / 443 to verify the authenticity of the domain
  - Requires you not to be using port 80 or 443 already (if you have a web server running, you can temporarily stop it)
  - Most useful when setting up a **new** domain that does not already exist on your webserver
- Run the following command to get your certificate(s):

  -
    ```
    $ sudo letsencrypt certonly -m me@example.com --standalone -d example.com
    ```

Issuing Certificates via webroot

- Webroot uses the root directory of your domain to verify the authenticity of the domain
  - Places files in the root directory and LE servers will check if the files are present for the domain
  - Most useful when setting up an **existing** domain that you are migrating to https
- Run the following command to get your certificate(s):

  -
    ```
    $ sudo letsencrypt certonly -m me@example.com --webroot -w /var/www/example.com/public_html/ -d example.com
    ```

Run it in prod!

Writing an nginx conf file (pt. 1)

```
server {
    listen       443 ssl;
    server_name  ex.io;
    root         /var/www/ex.io/public_html;

    access_log   /var/www/ex.io/logs/ex.io_access.log;
    error_log    /var/www/ex.io/logs/ex.io_error.log error;
```

Writing an nginx conf file (pt. 2)

```
    ssl              on;
    Ssl_certificate /etc/ssl/certs/ex_io/ex_io-fullchain.
pem;
    ssl_certificate_key /etc/ssl/certs/ex_io/ex_io-privkey.
pem;
    ssl_protocols    TLSv1 TLSv1.1 TLSv1.2;
    ssl_prefer_server_ciphers on;
```

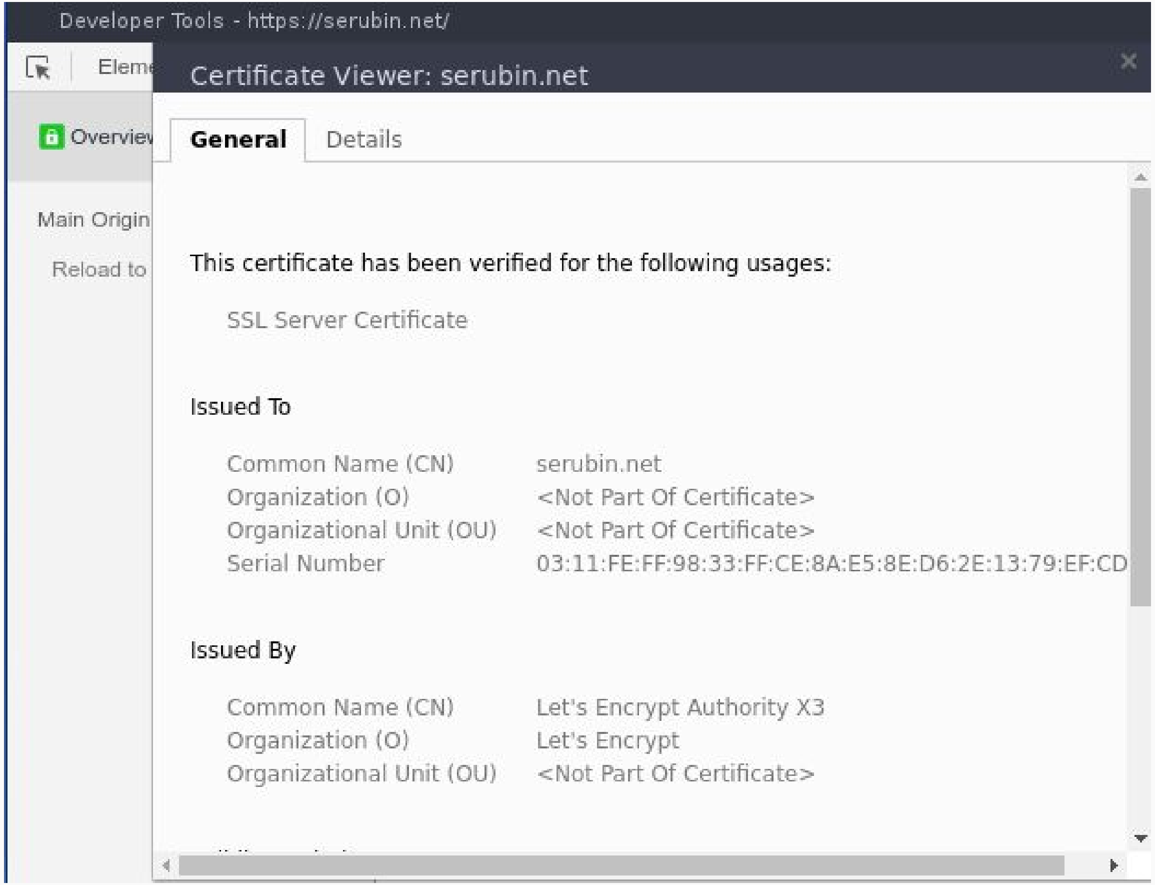
Writing an nginx conf file (pt. 3)

```
    ssl_ciphers "ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA:ECDHE-RSA-AES128-SHA:DHE-RSA-AES256-SHA256:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA:DHE-RSA-AES128-SHA:ECDHE-RSA-DES-CBC3-SHA:EDH-RSA-DES-CBC3-SHA:AES256-GCM-SHA384:AES128-GCM-SHA256:AES256-SHA256:AES128-SHA256:AES256-SHA:AES128-SHA:DES-CBC3-SHA:HIGH:!aNULL:!eNULL:!EXPORT:!DES:!MD5:!PSK:!RC4";
```

Writing an nginx conf file (pt. 4)

```
    location / {
        index  index.html index.htm;
        server_tokens off;
    }
}
server {
    listen      80;
    server_name ex.io;
    rewrite     ^   https://$server_name$request_uri?
permanent;
}
```

How ‘bout ‘dem apples?



Certificate Viewer: serubin.net

This certificate has been verified for the following usages:

SSL Server Certificate

Issued To

Common Name (CN)	serubin.net
Organization (O)	<Not Part Of Certificate>
Organizational Unit (OU)	<Not Part Of Certificate>
Serial Number	03:11:FE:FF:98:33:FF:CE:8A:E5:8E:D6:2E:13:79:EF:CD

Issued By

Common Name (CN)	Let's Encrypt Authority X3
Organization (O)	Let's Encrypt
Organizational Unit (OU)	<Not Part Of Certificate>

Apache Live Demo

Completely and totally unrehearsed.

Questions?
Comments?
Suggestions?

➔ Justin W. Flory

➔ Solomon Rubin