Let’s Encrypt as a Startup Success Story

Security and Ops and Encrypting the web

Daniel Jeffery
Linux Foundation

Why HTTPS?

“As our dependency on the internet has grown, the risk to users' privacy and safety has grown along with it.

Every unencrypted HTTP request reveals information about a user’s behavior, and the interception and tracking of unencrypted browsing has become commonplace.

Today, **there is no such thing as non-sensitive web traffic**, and public services should not depend on the benevolence of network operators.”

(emphasis theirs) https://https.cio.gov/everything/

Why Let’s Encrypt

Achieving HTTPS is time-consuming, confusing and can be costly.

To achieve ubiquitous HTTPS the Web needed a CA that was:

- Automated - Must be able to remove human inefficiencies
- Free - Even charging $0.01 adds significant complexity
- Open - Transparency is needed for trust

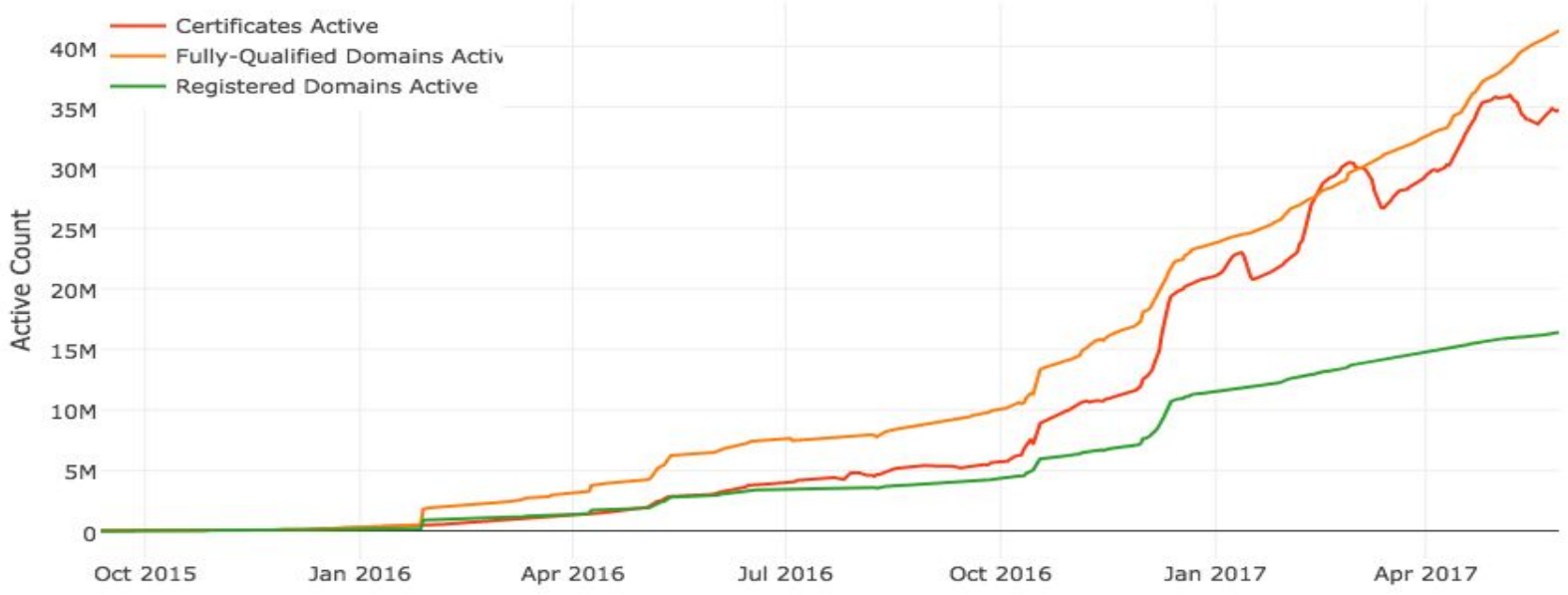
The Let’s Encrypt Mission

Encrypt the Unencrypted

Let’s Encrypt is seeking 100% encryption of all HTTP traffic.

> 90% of domains acquiring certificates from Let’s Encrypt **were previously using no encryption or not using publicly trusted certificates.**

Let’s Encrypt Numbers

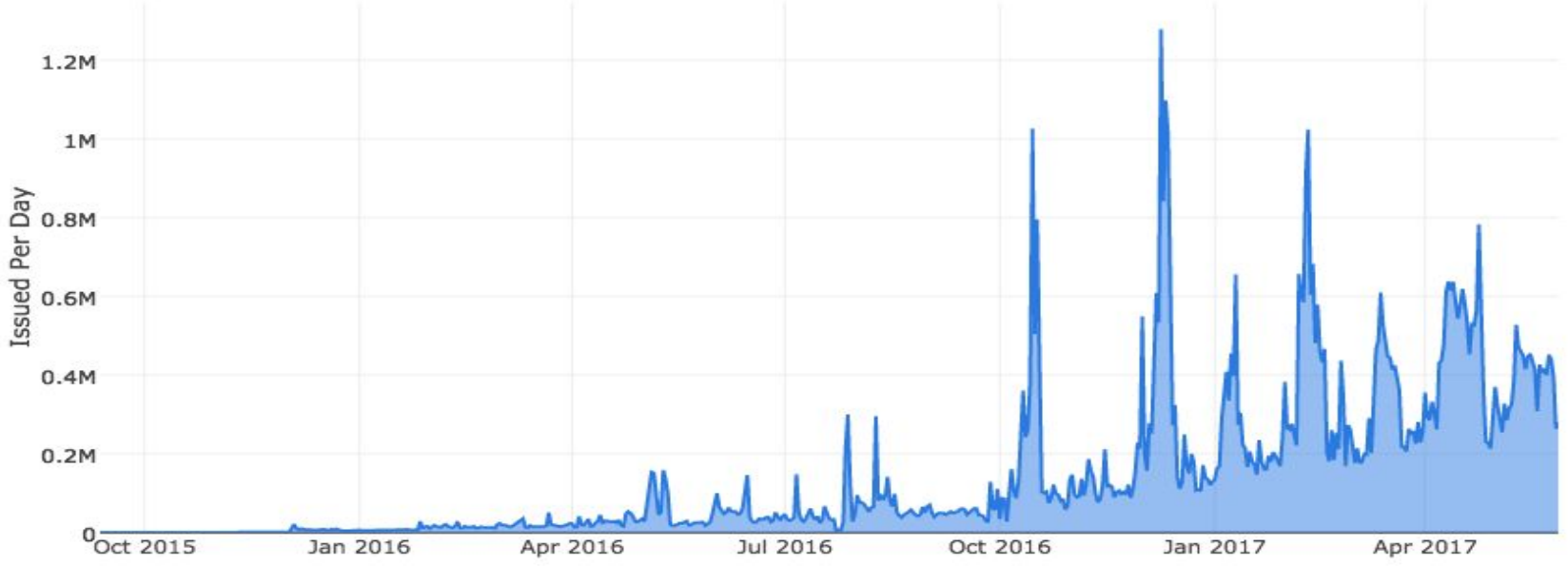


Let’s Encrypt Activity

Let’s Encrypt Numbers

Firefox Telemetry Average Encrypted Page Loads (14-day average)

Let’s Encrypt Numbers



Let’s Encrypt Certificates Issued per Day

Let’s Encrypt Numbers

Let’s Encrypt currently has a full-time staff of 10 people.

- Very lean web PKI org
- 2 Management and Administration
- 2 Let’s Encrypt Developers
- 1 Full-time LE developer from EFF
- 5 Let’s Encrypt Sysadmins

Let’s Encrypt Numbers

Lean hardware as well:

- About 1.5 racks of hardware. Total.
- In datacenter secure rooms
- Compute, storage, network gear, hardware security modules
- Akamai caching and DDOS in front of OCSP and new requests
- Primarily Linux and OSS
- Configuration management, reproducibility of all components

Web PKI - What it is

First, TLS (SSL) provides two things:

- Encryption
  - Agree on a protocol and away you go
- Authentication
  - Trickier, hence CAs

Web PKI - What Domain Validation is

Using a valid DV certificate from a trusted CA provides:

- Confidentiality
  - Encryption from the server software to the browser
- Integrity
  - Able to verify that the transmitted data has not been tampered with
- Authenticity
  - The owner of the private key used on the other end demonstrated control of this FQDN

OV and EV certs can offer Identification of the domain owner as well

Web PKI - Where it fails

Common threats:

- MITM attacks (institutionalized and ‘independent’)
- Impersonating the intended site

Common causes:

- CA incompetence, sub CAs, Government controlled or influenced CAs

HSTS and HPKP can help.

Web PKI - Understanding DV Certs

DV certificates are good for:

- Achieving end to end encryption with the intended FQDN

DV certificates do not:

- Assert anything about the entity behind the FQDN
- https://letsencrypt.org/2015/10/29/phishing-and-malware.html
- Trust DV for what it can be trusted for, and no further

Let’s Encrypt Infrastructure: 0 to public in 8 months

With ‘boulder’ already mostly complete LE:

- Built redundant DCs with full configuration management, logging and monitoring
- Passed WebTrust Audits with zero findings
- Yes, this required a hellish amount of work
- And it was awesome

Let’s Encrypt Infrastructure: 0 to public in 8 months - How?

Prioritize

Perfection is the enemy of getting anything done

Make the right compromises

Engage cooperative, dedicated people

Security and Ops in Startups

Why Security in a Startup?

- Security processes done right should result in more reliable service
- Nature of attacks
  - Opportunistic, doesn't matter that you're small
- Losing trust is difficult to recover from
- Difficult to add security later
  - Sometimes requires fundamental overhaul (very expensive)

Today for your Startup

We'll discuss:

- System patching, frequency and automation
- Monitoring and centralized logging
- Configuration management, code review and change control
- Separation of duties

Ops Context: Patching Dread

- Patching can and does break production environments.
- Any multi-year ops veteran will have effective horror stories to block patching.
- Conclusion:

Stock up on Red Bull and patch once a year on a weekend when C-level is out of town

Why patch frequently?

- Patching is the simplest and most effective way to be protected from existing threats
- Frequent patching means less things break at once
- Less breaking at once means easier to troubleshoot and keep the cyber flowing

Strategies to Patch Servers Frequently

- Just Do It
  - -Time sink
  - -Will lag behind other ‘priority’ projects
  - +Operations team is eyes on, ready to fix it as it breaks
  - +Every update is individually approved

- Ephemeralize everything and roll out new, patched instances every week
  - -Ensure all instances are tested in prod-like environment
  - -Very difficult with stateful components (especially databases)
  - Most feasible in the cloud
  - +Redundancy allows a clean cutover

Strategies to Patch Servers Frequently (cont.)

- Automate unattended updates
  - -Have a truly production-like staging environment that is under load at all times
  - -Monitor staging just like production for failures and interruptions of service
  - +Consistently up-to-date, fewer rushes to patch
  - +With staging, issues tested prior to production deploy
- Design your software with patching, config management and updates in mind

How to Patch Hardware

It still sucks.

Centralized systems exist, but most are prohibitively expensive for startups.

As automation is rarely an option at startup scale we need to closely watch security and update alerts, schedule time and Just Do It.

Consider options like RANCID to capture config changes.

Just as critical, not easy to automate.

Automation and Patching

Automation is how we fix human error. It is why we have computers.

Automation is just as important in patching.

Consistent, reliable processes mean when something goes wrong, it can be fixed.

Automated testing of your systems should happen immediately following patching (or constantly).

Really, Patch Stuff

Patching production systems must be accepted as a business priority and it must be done frequently to be meaningful.

Work to define a process that ensures stability while patching often and plan it into your production environment from day 1.

Monitoring and Centralized Logging

Monitoring of service performance and uptime typically occurs early, but security monitoring may not occur at all.

Centralized logging is often overlooked, even though it has great value for developers and general operations tasks.

Prevention is ideal, detection is a must.

Centralized Logging

- Plan for it from the beginning
  - Ensure sanitary and unsanitary logs (if needed)
  - Log in a way it can be made available *safely* to developers
  - Allows rapid and effective developer and operations troubleshooting
- More difficult for attacker to alter logs
- Central point for monitoring of logs

What to Monitor

Many lists and varies from system to system, but:

- Privilege usage
- New processes
- Logins and failures
- Errors, changes in volume of errors or traffic
- Blocked or denied connection attempts
- Changes to system files
- Advanced - capture memory and schedule memory forensics scans

Configuration Management, Code Review and Change Control

Use of a configuration management system is not typically explicitly required in regulations, but uniformity of the systems is.

Code review of configuration management changes, operational scripts and hardware config changes all result in accountability and far fewer ‘oopsies’ in production.

Change control in large organizations can be a nightmarish stalemate. In a startup it can be a lean and very effective method of ensuring code review, adequate testing and that the team is aware of each other’s work.

Configuration Management

Solutions such as Puppet, Chef, SaltStack and Ansible provide a framework

- Modules available for most common open source packages
- Enforce consistent configuration across environment and easily and quickly change it
- Rapid provisioning of new instances
- Ability to review configuration changes and approve them exactly as they will be deployed
- Management of local accounts or integration with centralized authentication system

Containers still benefit from being managed with a config management system so they are not locally altered.

Plan for separation of secrets such that config management can be shared with developers.

Code Review and Change Control

- Code review in systems like Gerrit, GitHub, Crucible, Bitbucket, etc.
  - Review of changes by operations and development staff possible
  - Enforce best practices and coding standards
  - Everything gets reviewed and formally approved
- Change Control enforces process, such as:
  - Code review before each stage
  - Everything goes through a full staging environment
  - Everything gets tested, not just baked, in staging

Fewer production issues and minimal downtime

>

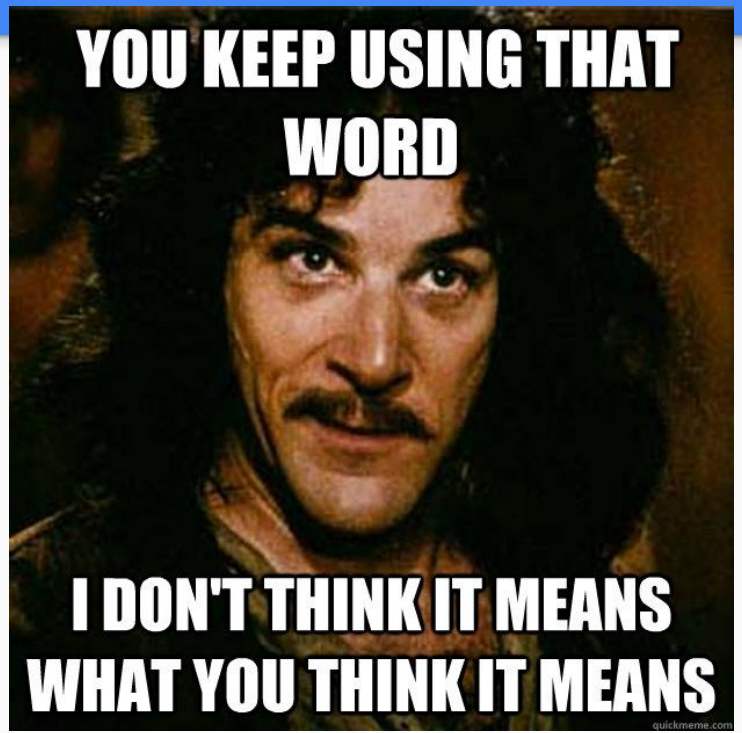
fastest possible deployment of a change

Why Separation of Duties

- This is teamwork not isolation
- Defines responsibilities
- Allows different priorities among team members
- Restrict risk of insider actions (malicious, rushed, incompetent, etc)
- Operations is not just the job for your worst developer

DevOps as I see it

- Can be the glue of separation of duties
- DevOps is not:
  - Developers doing WTF they feel like in production
  - Operations writing unmaintained, poorly thought-out code so they can be developers too
  - Everybody does everything
- DevOps IS:
  - Clearly defined roles
  - Collaboration and trust
  - Which means communication
  - Maximum visibility into dev and ops



Separation of Duties in a Lean Startup

Share as much between Dev and Ops as possible, collaboration not silos:

- Logs
- Configs
- Ticketing (at a minimum visibility)
- Production-like development environment
- Fully production-parallel staging environment
    - Including Hardware redundancy if at all possible

Separation of Duties in a Lean Startup

Plan for data to be shareable from the beginning.

- Separate secrets in config management
- Protect logs, share sanitary logs
- Share monitoring servers

Separation of Duties in a Lean Startup

Everything developed or forked in-house must be maintained and tested in-house (security, performance, stability, etc).

- This requires management buy-in
- Commit long-term resources or betray the future
- Rely on the community or the vendor, open bugs, commit upstream patches
- DevOps is not a license to write a bunch of crappy pet project code

Conclusion

By Implementing these few components commonly required in regulated environments, startups can be better protected from threats internal and external to the company.

Simultaneously they are creating an environment that is less volatile, yet up-to-date, collaborative and scalable.

Get Involved!

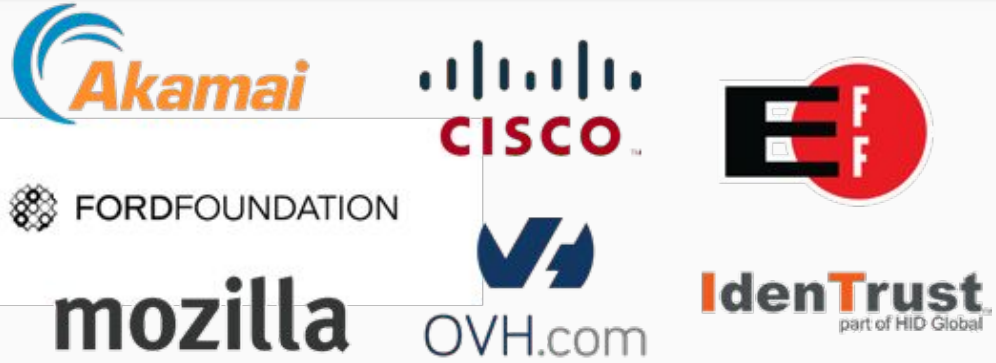
We are still in the early days of encrypting the entire Web. As geeks we need to reach out to facilitate change. Advocate, educate and encourage.

Additionally, Let’s Encrypt depends on industry and community support. Please consider getting involved with the project itself or a community client.

If your company or organization would like to sponsor Let’s Encrypt please email us at sponsor@letsencrypt.org.

Sponsors

Platinum & Gold



Silver

Q & A

Thanks for all the members of this great community who have so energetically supported Let’s Encrypt. We are changing the internet, together.