Let's Encrypt: An Automated Certificate Authority to Encrypt the Entire Web

Josh Aas*
Let's Encrypt

Richard Barnes*
Cisco

Benton Case
Stanford University

Zakir Durumeric
Stanford University

Peter Eckersley*
Electronic Frontier Foundation

Alan Flores-López
Stanford University

J. Alex Halderman*†
University of Michigan

Jacob Hoffman-Andrews*
Electronic Frontier Foundation

James Kasten*
University of Michigan

Eric Rescorla*
Mozilla

Seth Schoen*
Electronic Frontier Foundation

Brad Warren*
Electronic Frontier Foundation

ABSTRACT

Let's Encrypt is a free, open, and automated HTTPS certificate authority (CA) created to advance HTTPS adoption to the entire Web. Since its launch in late 2015, Let's Encrypt has grown to become the world's largest HTTPS CA, accounting for more currently valid certificates than all other browser-trusted CAs combined. By January 2019, it had issued over 538 million certificates for 223 million domain names. We describe how we built Let's Encrypt, including the architecture of the CA software system (Boulder) and the structure of the organization that operates it (ISRG), and we discuss lessons learned from the experience. We also describe the design of ACME, the IETF-standard protocol we created to automate CA–server interactions and certificate issuance, and survey the diverse ecosystem of ACME clients, including Certbot, a software agent we created to automate HTTPS deployment. Finally, we measure Let's Encrypt's impact on the Web and the CA ecosystem. We hope that the success of Let's Encrypt can provide a model for further enhancements to the Web PKI and for future Internet security infrastructure.

CCS CONCEPTS

• **Networks** → *Web protocol security*; • **Security and privacy** → *Security services*; *Usability in security and privacy*.

ACM Reference Format:
Josh Aas, Richard Barnes, Benton Case, Zakir Durumeric, Peter Eckersley, Alan Flores-López, J. Alex Halderman, Jacob Hoffman-Andrews, James Kasten, Eric Rescorla, Seth Schoen, and Brad Warren. 2019. Let's Encrypt: An Automated Certificate Authority to Encrypt the Entire Web. In *2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19), November 11–15, 2019, London, United Kingdom.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3319535.3363192

*Members of Let's Encrypt originating team.
†To whom correspondence may be addressed at jhalderm@eecs.umich.edu.

CCS '19, November 11–15, 2019, London, United Kingdom

ACM ISBN 978-1-4503-6747-9/19/11.
https://doi.org/10.1145/3319535.3363192

1 INTRODUCTION

HTTPS [78] is the cryptographic foundation of the Web, providing an encrypted and authenticated form of HTTP over the TLS transport [79]. When HTTPS was introduced by Netscape twenty-five years ago [51], the primary use cases were protecting financial transactions and login credentials, but users today face a growing range of threats from hostile networks—including mass surveillance and censorship by governments [99, 106], consumer profiling and ad injection by ISPs [30, 95], and insertion of malicious code by network devices [68]—which make HTTPS important for practically every Web request. Many cryptographic flaws in TLS have been discovered and mitigated (e.g., [11, 13, 17, 23, 37, 69]), but low adoption of HTTPS posed an even starker risk: as recently as 2015, 55–70% of browser page loads used *plaintext* HTTP [47].

A major barrier to wider HTTPS adoption was that deploying it was complicated, expensive, and error-prone for server operators [22, 57]. Most of the difficulty involved interactions with Certificate Authorities (CAs), entities trusted by Web browsers to validate a server's identity and issue a digitally signed certificate binding the identity to the server's public key. (Modern TLS implementations have negligible performance overhead in typical applications [48, 59].) To obtain and install a certificate, a server operator had to use arcane key generation software and configuration directives, follow manual steps to prove control of the domain name, and complete a payment transaction to a CA. The process was burdensome for smaller sites and difficult to integrate into large hosting platforms.

To reduce these barriers, facilitate broad adoption of HTTPS, and improve security for all Web users, we created Let's Encrypt, a CA that offers domain-validated certificates through a standard protocol at no cost to server operators. Let's Encrypt is the first browser-trusted CA designed for complete automation: identity validation and certificate issuance are fully robotic, and neither Web server operators nor CA staff need to take any manual steps.

Automation serves several goals for Let's Encrypt. On the Web server side, it greatly reduces the human effort required for HTTPS deployment, along with the concomitant risk of configuration errors that can lead to security problems [9, 14]. Automated support for Let's Encrypt has been integrated into Web server software [40, 67], IOT devices [16], large host platforms [71, 75], and CDNs [12].

On the CA side, automation bolsters security by reducing opportunities for human error, historically a frequent cause of misissuance events [86]. The *only* way for Let's Encrypt to validate a domain and issue a certificate is through the normal API; there is no manual override. Moreover, avoiding human intervention allows Let's Encrypt to keep the cost-per-certificate low and provide certificates at no charge. This eliminates two important impediments to HTTPS adoption: financial burdens and payment friction.

We designed Let's Encrypt to scale to the size of the entire Web. In just over three years of operation, it has issued more than 538 million certificates covering 223 million domain names. Today, there are more currently valid browser-trusted certificates issued by Let's Encrypt than issued by all other CAs combined.

This paper reports on our experiences building Let's Encrypt over the past seven years. We focus on three main results of that work:

ISRG Let's Encrypt is operated by an independent nonprofit we established called the Internet Security Research Group (ISRG). Running a CA requires round-the-clock operations staff, physically protected server infrastructure, and regular security and compliance audits, all of which ISRG oversees. Section 3 describes the organization's history and structure, its operating costs and funding model, and how it navigated becoming a trusted issuer and gaining acceptance in all major root programs.

ACME The key to Let's Encrypt's automation is ACME, a protocol for performing CA–server interactions, including certificate requests, domain validation, issuance, renewal, and revocation. Section 4 explains the principles behind ACME's design and operation, along with lessons learned while building it. ACME has recently been standardized by the IETF as RFC 8555 [20].

Boulder Let's Encrypt is powered by Boulder, an open-source ACME-based CA implementation designed for security, scalability, and high reliability. Section 5 describes Boulder's architecture, including design features motivated by past CA security failures, and discusses how Let's Encrypt operates Boulder in production.

Over the four years since Let's Encrypt launched, the fraction of browser page-loads that take place over HTTPS has approximately doubled, to 72–95%, according to telemetry from Google Chrome [47]. To shed light on how Let's Encrypt's has contributed to and helped shape this growth, we combine the CA's metrics with data from Internet-wide scans and Certificate Transparency logs (Sections 6 and 7). We find that more than a third of Alexa Top Million sites use Let's Encrypt, and Let's Encrypt is the fourth most popular CA in terms of handshakes from Firefox Beta users. We also survey the diverse ecosystem of ACME clients that Web servers and hosting providers use to acquire Let's Encrypt certificates, and describe the widely used ACME client we created, Certbot.

We hope that the success of Let's Encrypt can serve as a model for future efforts to create usable, scalable, and robust Internet security infrastructure. To help guide such efforts, Section 8 discusses some of the factors that we believe helped Let's Encrypt succeed where other proposals to provide universal server authentication have not. We also discuss major challenges facing the Web that Let's Encrypt has *not* solved—including combating phishing and securing domain validation against network attacks—and call for further security research to address them.

2 HTTPS BEFORE & AFTER LET'S ENCRYPT

HTTPS provides a simple experience for browser users: so long as the server presents a valid certificate that the browser trusts, users receive the benefits of an authenticated and encrypted connection without taking any action. For server operators, however, provisioning HTTPS had long been far more difficult. In this section, we review the state of affairs prior to Let's Encrypt's launch in late 2015, and we highlight several challenges inhibiting widespread HTTPS deployment that Let's Encrypt is designed to help overcome.

2.1 Certificate Costs and Marketplace

High prices for certificates were (and still are) common, as shown in Table 1. In 2015, the average price for a one-year single-domain certificate from the five largest CAs was $178, and for a wildcard certificate it was $766. (Prices for the same products from these CAs today are the same or higher.) Lower-cost alternatives did exist, but often came with significant limitations; because browsers treat all domain-validated certificates identically, they were the subject of extensive market segmentation by the CAs. One CA, StartCom offered free certificates for non-commercial use starting in 2011; however, it charged for revocation, which proved to be a source of problems when Heartbleed [37] forced many sites to revoke [108].

Beyond high prices, the certificate marketplace was complex and difficult for server operators to navigate, as evinced by the existence of third-party review sites such as SSL Shopper [91], which helped administrators compare prices and value for CA products. CAs often bundled certificates with additional services such as "vulnerability assessment scans" [92] and warranties covering losses to relying parties if the certificate was issued in error [45]. These add-ons could make it harder for customers to assess the true value of certificates. Some CAs even charged higher prices for certificates on keys that used more modern cryptographic algorithms; for example, Symantec charged $597 more per year to issue a certificate for an elliptic curve key [27] than for an RSA key [92].

Let's Encrypt offers domain-validated certificates for all domains at no cost. This eliminates financial barriers to HTTPS adoption and greatly lessens friction from confusion and transaction costs.

2.2 Obtaining and Installing a Certificate

Prior to our work, the process of obtaining a certificate and configuring an HTTPS server was often manual and tedious [22]. First, system administrators had to recognize that they needed a certificate and navigate the confusing marketplace. Then they had to

Certificate Authority	Mid-2015 Prices		Mid-2019 Prices	
	Single	Wildcard	Single	Wildcard
GoDaddy [45, 46]	$69	$332	$79	$369
Comodo/Sectigo [31, 85]	$76	$404	$92	$422
GeoTrust [43, 44]	$149	$499	$149	$688
DigiCert [32, 33]	$195	$595	$207	$653
Symantec [92, 93]	$399	$1999	$399	$1999

Table 1: Prices for a one-year certificate for non-free CAs with the largest market shares. Single domain offerings are domain-validated; wildcard offerings sometimes require organization validation. The 2015 prices are from shortly before Let's Encrypt began offering service to the public.

generate a private key, create a certificate signing request (CSR), and perform the verification steps required by the CA. After waiting (sometimes for days or more) for the CA to issue the certificate, they had to figure out how to configure their server with the certificate and the appropriate trust chain. All these steps typically involved esoteric commands, and in practice many users blindly followed a tutorial. Even when a CA issued a certificate immediately, usability studies show that it would often take administrators over an hour to install the certificate on their servers [57]. These tasks had to be repeated every year or so when the certificate expired.

Besides being difficult, manual HTTPS server setup carries the risk of introducing security vulnerabilities through misconfiguration. A study in 2013 found that only 45% of certificates on HTTPS servers were correctly configured, while about 13% were configured so poorly that their corresponding Web sites were completely inaccessible for some clients [36].

Certificate renewals present additional challenges. Expired certificates can cause browser warnings for site visitors, so administrators run the risk of driving away customer traffic if they fail to replace old certificates before they expire. Prior to Let's Encrypt, renewal lapse occurred for about 20% of trusted certificates [36].

Let's Encrypt seeks to overcome all these problems through automation. We designed the ACME protocol (Section 4) to allow software agents on HTTPS servers to obtain, provision, and renew certificates automatically, with no user interaction. Third-party contributors have created a rich ecosystem of ACME clients that automate HTTPS deployment on systems ranging from IoT devices to traditional Web servers to large hosting platforms (see Section 6).

To further promote automation, Let's Encrypt limits certificate lifetimes to 90 days, rather than one or multiple years as CAs have traditionally offered. This is short enough to strongly encourage operators to automate things (few people will want to manually renew certificates every quarter) but long enough to allow for manual renewal if necessary in a particular deployment. Shorter certificate lifetimes potentially enhance security, since they lessen reliance on certificate revocation in the event of key compromise [97], and some clients do not check revocation status [61].

2.3 CA Validation Practices

Validating the identity of certificate subjects is one of the main jobs of any CA. The CA/Browser Forum—a voluntary consortium of CAs and browser and platform vendors that works to standardize Web PKI ecosystem rules—defines several categories of validation [28]. For "Domain Validation" (DV), CAs confirm that the requester has control over the domain name to be listed in the certificate. For "Organization Validation" (OV), CAs verify public business registration documents for the requesting organization. "Extended Validation" (EV) takes this process even further, with more intensive checks of proper ownership of identifiers.

OV and EV require manual submission and verification of identity documents, which significantly slows the issuance process [29]. Some CAs limit useful certificate features (such as wildcard names) to these stricter and more expensive validation levels [43].

OV and EV were intended to provide greater protection for site visitors [29]. However, research has found that users do not notice or understand the distinction between validation levels [96], negating the potential security benefit. Consequently, Chrome [73], Firefox [94], and Safari [53] have or will soon drop UI distinctions between validation levels. Moreover, Web clients treat all valid certificates for a given domain, whatever the validation level, as the same security context [21], further undermining the security benefit of OV and EV certificates.

Let's Encrypt offers only domain validation, since only DV can be completely automated and the security benefits of OV and EV are unclear. The CA provides both regular and wildcard DV certificates, and supports several methods of verifying control of a domain (see Section 4) to cover a wide spectrum of use cases.

3 ISRG: THE HOME OF LET'S ENCRYPT

Creating an automated CA that could scale to the size of the Web required more than building software and protocols. We also needed to establish an organization to operate the CA, develop processes that would provide high security and high availability, find a sustainable funding model, and become a trusted issuer so that our certificates would be accepted by major browsers and platforms. In this section, we discuss how ISRG, the organization that runs Let's Encrypt, came to be, and how it accomplished these goals.

3.1 History and Organizational Structure

Let's Encrypt was created through the merging of two simultaneous efforts to build a fully automated certificate authority. In 2012, a group led by Alex Halderman at the University of Michigan and Peter Eckersley at EFF was developing a protocol for automatically issuing and renewing certificates. Simultaneously, a team at Mozilla led by Josh Aas and Eric Rescorla was working on creating a free and automated certificate authority. The groups learned of each other's efforts and joined forces in May 2013.

That month, they formed the Internet Security Research Group (ISRG), a nonprofit corporation, to be the legal entity operating Let's Encrypt. It was decided that ISRG should be a nonprofit because nonprofit governance requirements—such as no profit motive, no ownership, relatively high transparency, and a public service mission—would help ensure that the organization served the public in a stable and trustworthy manner over the long term. Josh Aas has served as ISRG's Executive Director since its founding.

Initially, ISRG had no full-time staff. Richard Barnes of Mozilla, Jacob Hoffman-Andrews of EFF, and Jeff Hodges (under contract with ISRG) began developing Let's Encrypt's CA software stack. Josh Aas and J.C. Jones, both with Mozilla at the time, led infrastructure development with assistance from Cisco and IdenTrust engineers. ISRG's first full-time employee, Dan Jeffery, joined in April 2015 to help prepare the CA's infrastructure for launch. Simultaneously, James Kasten, Peter Eckersley, and Seth Schoen worked on the initial ACME client (which would eventually become Certbot) while at the University of Michigan and EFF. Kevin Dick of Right Side Capital Management, John Hou of Hou & Villery, and Josh Aas constituted the team responsible for completing a trusted root partnership deal and signing initial sponsors.

Let's Encrypt was publicly announced on November 18, 2014 [1], issued its first browser-trusted certificate on September 14, 2015 [4], and began providing service to the public on December 3, 2015 [2].

As of May 2019, ISRG had 13 full-time staff: six site reliability engineers, three software developers, and four administrative staff.

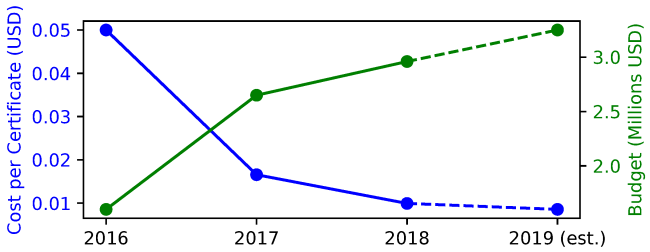


Figure 1: ISRG operating budget and effective cost per certificate. Automation has allowed Let's Encrypt to keep operating costs stable even in the face of tremendous growth in certificate issuance since its launch.

Site reliability engineers are responsible for the CA's infrastructure and are the only staff with access to systems that issue certificates. Administrative staff work on fundraising, finance, communication, management, legal, and other organizational administration tasks. Software developers work primarily on the CA's Boulder software, while also contributing back to various open source projects. Since launch, the CA has regularly introduced new features, including IPv6 support (July 2016), internationalized domain name support (October 2016), ACMEv2 support (March 2018), and wildcard certificate support (March 2018).

ISRG is governed by a board of directors, which meets quarterly. As of September 2019, there are ten directors, including representatives from public benefit entities, educational institutions, and major financial contributors, as well as the ISRG Executive Director [54].

3.2 Budget and Fundraising

ISRG spent approximately $3.0M in 2018 and is projected to spend approximately $3.3M in 2019. (These figures exclude money set aside as cash reserves and do not reflect in-kind contributions, which significantly reduce costs for third-party services and hosting.) Staffing is ISRG's largest category of expense, accounting for about two-thirds of the budget. Daily CA operations are heavily automated, so most staffing costs go towards software engineering and systems administration of the automated infrastructure. This has allowed ISRG to maintain stable operating costs despite enormous growth in certificate issuance, as shown in Figure 1.

Initially, ISRG was funded almost entirely through large donations from technology companies. In late 2014, it secured financial commitments from Akamai, Cisco, EFF, and Mozilla, allowing the organization to purchase equipment, secure hosting contracts, and pay initial staff. Today, ISRG has more diverse funding sources; in 2018 it received 83% of its funding from corporate sponsors, 14% from grants and major gifts, and 3% from individual giving.

3.3 Becoming a Trusted Issuer

One of the greatest hurdles to establishing Let's Encrypt was acquiring the ability to issue certificates that would be trusted by existing clients. New CAs can apply for inclusion in browser and OS root programs, but the time from acceptance of new roots to broad client deployment is prohibitively long. For example, years may elapse between when new roots are added to Google's root program and when they are available on the majority of Android devices [100].

For a new CAs to be widely trusted at launch, it either needs to purchase an established, already trusted root, or get cross-signed by a trusted authority. A "cross-signed" intermediate CA certificate is signed by *another* CA. If the other CA is already widely trusted by root programs, this makes the cross-signed intermediate, and any certificates it issues, immediately widely trusted as well.

ISRG initially investigated purchasing an existing root, as this would provide the greatest flexibility. The estimated price for a root at the time was $1–8M, depending on remaining lifetime, extent of trust, cryptographic algorithm type, and attached liabilities. We made a small number of offers, but never completed an acquisition.

In October 2014, ISRG executed a long-term cross-signing agreement with IdenTrust [3], a root authority trusted by Mozilla, Apple, and Microsoft [63]. IdenTrust was attractive as a partner because it had a sufficiently trusted root, offered cross-signing, and supported Let's Encrypt's mission. IdenTrust also offered WebTrust-compliant datacenter environments, which Let's Encrypt needed to host its servers and gained access to as part of the agreement.

ISRG has since established a root trust anchor, which was accepted into all major browser and platform root programs as of August 2018 [6] and is gradually becoming widely deployed. Starting in July 2020, Let's Encrypt will default to issuing certificates with a trust chain leading to the ISRG root instead of IdenTrust [7].

3.4 Legal Environment

As a U.S.-based entity, ISRG endeavors to craft policies that comply with U.S. law, including applicable sanctions. Although service is not provided to people or entities on the U.S. Treasury Department's Specially Designated Nationals (SDN) list [98], and service to a number of governments is limited as a result of sanctions, Let's Encrypt serves the vast majority of people and entities in every country.

ISRG occasionally receives requests from law enforcement and relies on legal counsel to determine whether or not the requests are legitimate, in whole or in part. It publishes a Legal Transparency Report every six months in order to provide information about the numbers of requests received [66]. Let's Encrypt has never been compelled by law enforcement, or any other entity, to issue a certificate (e.g., to facilitate HTTPS interception attacks). The Web security community generally recognizes this threat, along with the risk of CA compromise, as important motivations [88] for the adoption of Certificate Transparency (CT) [62]. Let's Encrypt has always fully participated in CT.

4 THE ACME PROTOCOL

A central element in the creation of Let's Encrypt was the Automated Certificate Management Environment (ACME) protocol. Unlike prior certificate management protocols (e.g., CMP, CMC, and EST [10, 77, 82]), ACME is tailored to the needs of the Web PKI and was designed with scalable, automated issuance as a core goal.

Let's Encrypt launched with an initial version of the protocol that our team designed internally [55]. This initial version became the basis for the IETF ACME working group, which published a final version as RFC 8555 [20]. The initial and final versions are sometimes referred to as "ACMEv1" and "ACMEv2", respectively. In this section we will use "ACME" to refer to the final protocol version (v2) unless otherwise specified.

4.1 Protocol Overview

Earlier certificate management protocols presume pre-existing relationships between the certificate applicant and the CA, such as pre-shared secrets. In contrast, ACME provides tools to automate the entire relationship between a CA and a certificate applicant. A typical interaction is illustrated in Figure 2 and includes the following tasks:

Account management. CAs need to track applicants' identities in order to know which applicants are allowed to request issuance for which identifiers. ACME provides a minimal notion of an *account*, which is essentially a key-pair that the applicant registers with the CA, together with some optional metadata. The focus of ACME on automation enables the use of digital signatures to authenticate applicants, as opposed to less secure means like pre-shared keys or passwords. In addition to basic functions like account registration and account key rotation, ACME also provides an external account binding feature, by which CAs can associate a bare-bones ACME account with a richer account in some other system.

Authorization and identifier validation. Before a CA can issue a certificate containing a given identifier to a given applicant, it must verify that applicant's authority to represent that identifier. In the Web PKI, this is typically done by the CA requiring the applicant to demonstrates control of the identifier. ACME has a notion of an *authorization* being bound to an account, by virtue of the account holder completing *challenges* that validate control of the identifier. It supports challenges that implement the most commonly used and easily automated validation methods, as discussed in Section 4.2.

Certificate request and issuance. Once an ACME client has registered an account with a CA's ACME server, it can request that a certificate be issued by sending the CA a description of the desired certificate: what identifiers, what lifetime, etc. Although a PKCS#10 Certificate Signing Request (CSR) [72] must still be sent by the client, the request description is encoded in a JSON object, since the CSR format is unable to express, for example, a request for specific `notBefore` and `notAfter` values. This request creates an *order* that is used to track the process of authorizing and issuing the certificate. The CA populates the order with the authorizations that the CA requires the client to complete before it will issue the certificate.

In most cases, authorizations will correspond to identifiers—issuing for `example.com` will require demonstrating control over `example.com`. However, this model also allows for more exotic cases such as wildcard certificates, where validation cannot be done directly on the identifier in the certificate. Let's Encrypt's policy, for example, is that a wildcard certificate (e.g., for `*.example.com`) requires proving control over the base domain name (`example.com`) using a specific challenge type.

ACME has no notion of renewal. Instead, a client "renews" a certificate by simply requesting a new certificate.

Certificate revocation. ACME provides a mechanism for revocation of a certificate to be requested by any of the following parties:

(1) The account that caused the certificate to be issued;
(2) Any other account that can prove control of the identifiers in the certificate; or
(3) Anyone who can prove control of the private key corresponding to the public key in the certificate.

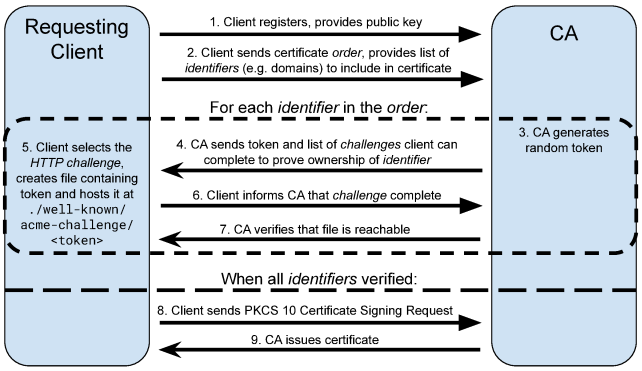


Figure 2: ACME protocol. This diagram illustrates how an ACME client can obtain a certificate without human interaction. In the dashed region, the client proves ownership of the domain using an HTTP-based challenge.

The last case is an especially important innovation over prior protocols, since it allows a good Samaritan who encounters a leaked key to request automated revocation of corresponding certificates.

4.2 Validation Methods

The ACME challenge mechanism for identifier validation is extensible so that new validation mechanisms can be added over time and so that CAs can implement their own mechanisms. (Acceptable methods for the Web PKI are defined in the CA/Browser Forum's Baseline Requirements [28].) There are currently three specified challenge types, all of which are supported by Let's Encrypt:

(1) The **HTTP** challenge requires the applicant to serve an object containing a CA-provided random value at a specific HTTP URL at the domain. The CA makes GET requests for the URL and verifies that the correct object is returned.

(2) The **DNS** challenge requires the applicant to provision a DNS record at `_acme-challenge.<domain>` containing a CA-provided random value. The CA fetches this record and verifies that its content is correct.

(3) The **TLS-ALPN** challenge requires the applicant to configure a TLS server to respond to a TLS ClientHello message containing a specific ALPN value and an ACME-specific TLS extension [42, 87]. The TLS server must then present a self-signed certificate containing a CA-provided random value and correctly complete the TLS handshake.

These three mechanisms allow ACME to be integrated into a variety of operational environments. For example, the HTTP challenge is easy to deploy on domains that have single Web servers, but it requires the response to be provisioned across all servers in load-balanced configurations. The TLS-ALPN challenge allows for certificate management to be done by a TLS entity with no HTTP logic of its own, e.g., by a TLS-terminating load balancer.

The various challenge types offer slightly different levels of assurance, and it is up to CAs to decide which are appropriate in a given situation. Let's Encrypt, for example, only allows a wildcard certificate to be issued if the applicant has proven control with the DNS challenge, since this challenge type proves direct control

over the DNS zone in question (and such control could be used to provision arbitrary subdomains and obtain certificates for them).

In the 90 days preceding September 22, 2019, Let's Encrypt completed 266M successful challenges with ACME clients. Of these, 86% used HTTP, 13% used DNS, and 0.3% used TLS-ALPN.

4.3 Security Improvements

While ACMEv1 and ACMEv2 share many attributes, several significant changes were made in ACMEv2 in response to security analyses performed during the IETF standardization process.

Bhargavan et al. formally verified the security of ACME [24]. They modeled the protocol in the applied pi calculus and demonstrated its security in an even more robust threat model than the one addressed by traditional CAs' issuance processes. However, their work also uncovered weaknesses involving some subtle authentication properties. ACMEv2 incorporates improvements based on their recommendations.

IETF security review also highlighted some areas for privacy improvements Most notably, all GET requests (which lacked any authentication) were replaced with authenticated POST requests, so that CAs can apply appropriate authorization policies.

ACME validation methods have also evolved in response to security findings. In general, all validation methods that rely on empirical validation of control (as all of the above do) are vulnerable to network-layer attacks on the validation process, such as BGP hijacking to reroute validation requests [25, 26, 84]. The ACME RFC discusses these risks in detail, and suggests some mitigations [20].

However, validation methods face a range of other, often subtle threats due to the diversity of server and hosting-provider behavior. During ACME's development, two additional challenges types were proposed (one of which was deployed by Let's Encrypt), but they were found to be insecure in common shared-hosting environments:

HTTPS-based HTTP challenge. Initially, HTTP challenges were allowed to be completed on port 80 (over HTTP) or port 443 (over HTTPS). In 2015, Kasten and Eckersley noted that Apache would serve the first HTTPS site defined in its configuration file if the requested domain did not have HTTPS enabled. In other words, if the same Apache instance served both `site1.com` and `site2.com` but only `site2.com` supported HTTPS, all HTTPS requests to the server—regardless of whether the client requested `site1.com` or `site2.com`—would be served with content from `site2.com`. In shared-hosting environments, this behavior could allow an attacker to acquire certificates for domains they did not control. In response, the HTTP challenge type was changed to prohibit HTTPS requests.

TLS-SNI challenge. In early 2018, Franz Rosén found that the TLS-SNI challenge supported by Let's Encrypt did not provide sufficient authentication in many shared-hosting environments [5, 81]. This challenge validated domain control by requiring the applicant to configure a TLS server at the domain to host a specific challenge certificate. This certificate had to contain a subject alternative name (SAN) with a CA-specified random value as a subdomain of `.acme.invalid`. The server was required to respond with the certificate if the subdomain was queried using the Server Name Indication (SNI) extension [38] in the TLS handshake.

The challenge design incorrectly assumed that hosting providers prevent clients from uploading certificates for domains they do not own (including `.invalid` domains). After Rosén demonstrated that this was often not the case, Let's Encrypt immediately disabled the challenge, then re-enabled it for a small set of whitelisted non-vulnerable providers. Let's Encrypt eventually phased it out and replaced it with the TLS-ALPN challenge described above, which represents the result of security lessons learned from this process.

5 CA SOFTWARE AND OPERATIONS

When we started to develop Let's Encrypt, available CA software was designed around the manual processes that CAs typically followed. ACME, with its focus on automation, differs substantially from these human-oriented workflows, and we needed to build entirely different software to operate an automated CA. We created Boulder, a new, open-source CA software stack that forms the core of Let's Encrypt's operations.

5.1 Security Principles

Let's Encrypt must achieve high security and availability in order to remain trustworthy and fulfill its role as Internet infrastructure. Both Boulder and the CA's operations reflect a set of design principles that support these goals:

Minimal logic. Boulder implements the minimal logic necessary to instantiate an ACME-based CA, in order to make it easier for developers and auditors to verify the correctness of the code. Boulder is implemented in a relatively high-level language, with structured intercomponent communications (Go and gRPC, respectively), which further increase the comprehensibility of the code.

Minimal data. To reduce the potential harm from data breaches, Let's Encrypt collects only the minimum subscriber data necessary to deliver the service in compliance with ACME and WebTrust [103] requirements: a public key, an optional contact email address, and various access logs [64].

Full automation. The system has a single issuance path, based on ACME, with no facility for human operators to manually create certificates or make one-off policy exceptions. Locking out humans prevents errors that have led to misissuance by other CAs [80].

Functional isolation. Boulder is composed of limited-purpose components that communicate only through well-defined APIs. Components with different risk levels are physically isolated from one another. This reduces the risk that compromise of more exposed components (e.g., the Internet-facing front end) will lead to compromise of more critical components (e.g., CA signing keys).

Operational isolation. To limit risk of physical compromise, access to Let's Encrypt data centers is strictly limited, even for our staff. Most administration is performed remotely, with engineers only entering the data centers to complete tasks that require physical access to hardware. Remote administration tasks are protected by multifactor authentication and strong monitoring. Staff with administrative access can only access administration functions via dedicated, restricted virtual machine environments within a security-focused operating system running on specific laptops.

Continuous availability. Boulder's componentized architecture allows multiple redundant instances of each function to be run in parallel. Let's Encrypt operates Boulder instances in physically

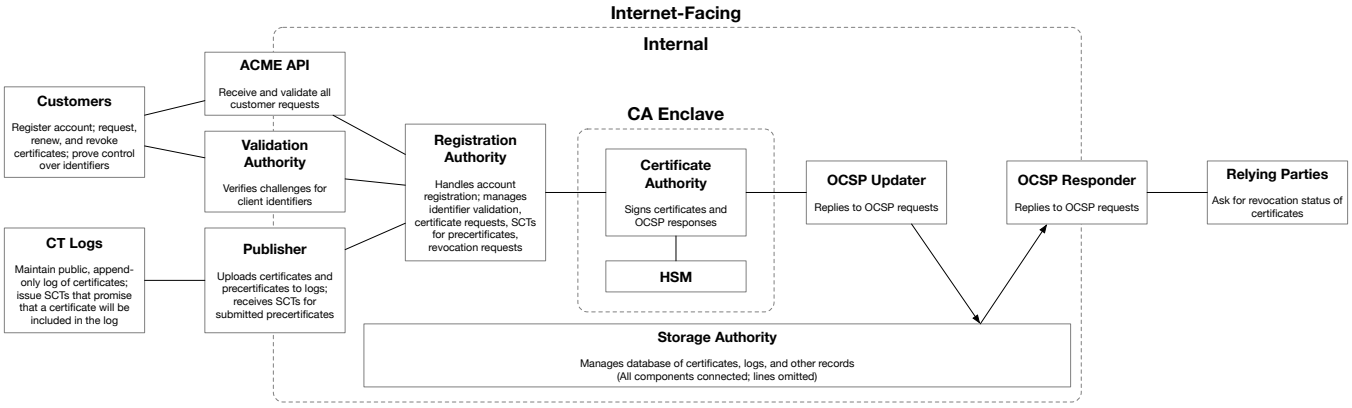


Figure 3: Boulder architecture. Let's Encrypt developed and operates a Go-based open-source CA software platform named Boulder, which is composed of single-purpose components that communicate over gRPC, as illustrated here. The certificate lifecycle unfolds roughly from left to right in the diagram.

secure, geographically distributed facilities, and divides production traffic between datacenters, with load balancing handled by a CDN.

In addition, security audits by outside firms are regularly commissioned. Physical access to core CA equipment, all HSM administrative operations, and certain other functions require multiple staff members to present authentication. For most other administrative operations, including software deployments, logging and monitoring systems provide strong accountability. These protections help ensure that a rogue engineer would have difficulty stealing a signing key, and although they might be able to cause misissuance, they would likely be caught after the fact.

To reduce the chances that bugs in Boulder could lead to misissuance or other problems, new versions are tested in a public staging environment prior to entering production. Boulder also applies ZLint [58] to perform automated conformance tests on every certificate, and any ZLint notices, warnings, or errors block issuance.

5.2 System Architecture

Boulder implements the three main functions required of a CA in the modern Web PKI:

(1) Issuance of certificates (via ACME);
(2) Submission of precertificates and certificates to Certificate Transparency (CT) logs [62]; and
(3) Publication of certificate revocation status via OCSP.

Figure 3 illustrates Boulder's components and their interactions. All instances of a given Boulder component run independently. For example, Let's Encrypt runs four instances of the *Certificate Authority* component, each of which can sign certificates and OCSP responses without communication with the other three instances.

ACME clients communicate with Let's Encrypt exclusively via the *ACME API* component, which validates client requests and relays them to the *Registration Authority* (RA). The RA orchestrates the authorization and issuance process. For example, when a client responds to an ACME challenge, the RA instructs the *Validation Authority* to validate that the client has completed the challenge. Once the RA has verified that the client has sufficient authorization to issue a requested certificate, it instructs the *Certificate Authority* to issue a precertificate, which is submitted to CT logs by the *Publisher*. When a CT log provides a Signed Certificate Timestamp (SCT), the RA instructs the Certificate Authority to issue the certificate and makes it available to the client.

Each Certificate Authority has access to a hardware security module (HSM) that stores the private key corresponding to one of the Let's Encrypt intermediate certificates. The private key corresponding to the ISRG root certificate is held in a separate, offline HSM. The online HSMs also hold a private key corresponding to an OCSP signer certificate, used to sign OCSP responses. As of September 2019, Let's Encrypt HSMs perform approximately 450 signatures per second, with the bulk of this work going towards OCSP responses and the remainder towards signing certificates.

OCSP responses are computed proactively, asynchronously from OCSP requests. The *OCSP Updater* monitors for new certificates or newly revoked certificates, and instructs the Certificate Authority to sign appropriate OCSP responses. As these responses expire, the OCSP Updater issues fresh ones, until the certificate expires. All of these OCSP responses are stored in a database, from which the *OCSP Responder* answers OCSP requests. OCSP responses are served and cached via Akamai, allowing Let's Encrypt to keep up with OCSP traffic even for very popular sites.

The architecture described here has allowed Let's Encrypt to scale up to issuing more than a million certificates per day at peak, running on modest infrastructure. The biggest scalability challenge has not been compute power or signing capacity, but rather the growth of the primary CA database and of log data. As of September 2019, Let's Encrypt produces about 150 GB of CA log data per day. All of this log data is kept online and searchable for 90 days (the lifetime of certificates), and, in compliance with WebTrust audit requirements, a subset is archived offline for a minimum of seven years.

6 CLIENT ECOSYSTEM

There is no official client for Let's Encrypt. Instead, by standardizing ACME, the project has fostered an ecosystem of third-party clients that support a wide range of platforms and use cases [65]. ACME client libraries are available in a variety of languages, including C, Go, and Java. There are a range of command-line clients, such as acme.sh and Certbot, as well as server software with built-in

provisioning, such as Caddy, Apache, and cPanel. Shared hosting providers, including Squarespace, Google, and OVH, also use Let's Encrypt to transparently provision HTTPS. In this section, we survey some of the noteworthy clients and gauge their popularity.

6.1 Certbot

In conjunction with Let's Encrypt's launch, the Electronic Frontier Foundation (EFF) created an ACME client, now named Certbot [40]. Unlike most other clients, Certbot aims to fully automate secure HTTPS deployment, rather than simply procuring a certificate. Certbot currently is the most popular client in terms of unique IP addresses. During the time period we studied, 50% of clients (by unique IP address) used Certbot, and about 16% of certificates were requested using it, as shown in Table 2. This suggests that Certbot is particularly popular with operators of individual servers.

Certbot automatically parses and modifies configuration files for popular Web servers, including Apache and Nginx. This lets Certbot automate complicated HTTPS configuration steps that have historically confused administrators [36, 107] and contributed to TLS vulnerabilities (e.g., [11, 17, 23]). Furthermore, Certbot updates can automatically patch server configurations to apply new security configuration recommendations or mitigate newfound attacks.

Certbot's extensible Python-based framework supports a variety of Web servers and validation types. *Authenticator* plugins implement means of satisfying ACME challenges. Certbot provides authenticators for solving HTTP challenges with or without an existing HTTP service, as well as for solving DNS challenges with common DNS providers such as Amazon Route 53. *Installer* plugins configure a service to use a newly obtained certificate and can modify configurations for improved security. Current installers support Apache, Nginx, HAProxy, and other services.

Working with server configuration files has been one of the most difficult challenges for Certbot. Neither Apache nor Nginx offers an API to access its configuration, and their configuration languages are not well specified. To reduce the chance of breaking the server's configuration, Certbot runs automated checks and will revert any modifications if a test fails. Unfortunately, this cannot guarantee that all changes have the intended effects. We encourage server developers to provide APIs that allow external services to modify their configurations, publish a formal specification for their configuration languages, and provide secure defaults that do not require correction by third-party tools.

ACME Agent	Certificates	FQDNs	Client IPs
cPanel	10.0M (17.5%)	63.8M (48.7%)	87K (2.95%)
Certbot	9.4M (16.4%)	14.8M (11.3%)	1.5M (49.5%)
Squarespace	5.1M (8.97%)	5.2M (3.95%)	<50 (0.00%)
acme4j	4.6M (8.09%)	7.8M (5.92%)	2.4K (0.08%)
Net-ACME2	3.3M (5.79%)	3.7M (2.79%)	13K (0.43%)
curl	3.1M (5.53%)	4.0M (3.09%)	113K (3.83%)
Acme::Client	3.0M (5.21%)	1.1M (0.84%)	15K (0.51%)
Plesk	2.0M (3.49%)	3.7M (2.84%)	164K (5.54%)
xenolf-acme	1.8M (3.16%)	1.5M (1.15%)	54K (1.82%)
Go-http-client	1.7M (2.91%)	0.2M (0.16%)	192K (6.48%)
acme.sh	1.7M (2.91%)	2.3M (1.73%)	211K (7.11%)
eggsampler	1.2M (2.05%)	3.6M (2.76%)	11K (0.36%)
OVH ACME	0.9M (1.65%)	2.4M (1.85%)	<50 (0.00%)
Google	0.9M (1.63%)	1.2M (0.89%)	<500 (0.02%)
Other (130k unique)	8.4M (14.7%)	15.8M (12.1%)	0.6M (21.4%)
Total	57.2M	147M	3.0M

Table 2: Most popular ACME clients. There is a rich ecosystem of ACME clients. Here we show the clients that requested the most certificates in December 2018 and January 2019, based on data from Let's Encrypt logs. Some clients that are integrated with large hosting providers (e.g., Squarespace) tend to issue many certificates from a small number of IP addresses.

6.2 Servers with Automatic Provisioning

Several Web servers can automatically provision Let's Encrypt certificates. One of the first was Caddy [67], a Go-based Web server with an integrated ACME client that provisions and renews certificates without user interaction. Servers running Caddy exhibit nearly ubiquitous HTTPS deployment and use of modern TLS configurations, although they account for only a small fraction of Let's Encrypt certificates. We hope to see other popular server software follow Caddy's lead. The Apache project's `mod_md` provides similar functionality [15], although it has yet to see significant adoption.

Web control panels, which help administrators configure servers, have also added ACME support, including cPanel [101], Plesk [76], and Webmin [102]. These integrations are frequently deployed by hosting providers and help drive HTTPS adoption for smaller sites. They account for a huge portion of Let's Encrypt usage. About half of domains with a Let's Encrypt certificate and 18% of Let's Encrypt certificates are issued using the cPanel plugin alone.

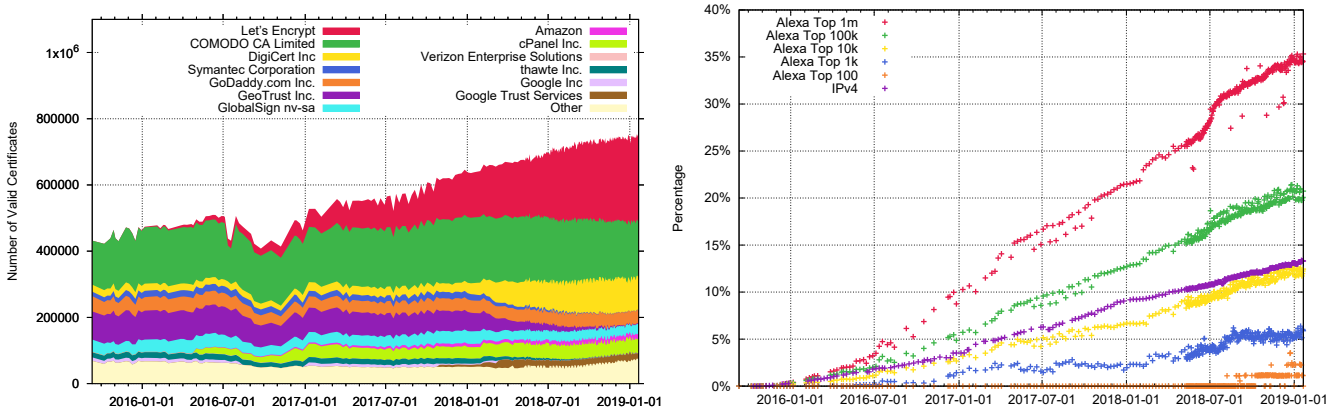
6.3 Hosting Providers

Prior to Let's Encrypt, a disproportionate number of sites without HTTPS were small sites, a significant number of which were served by shared hosting providers that controlled all aspects of HTTPS configuration (see Section 7). Often, these services provided no means for enabling HTTPS or required customers to upload their own certificates—an approach that was problematic for the technically unsophisticated users that such providers tend to attract.

Let's Encrypt has enabled a number of very large providers to automatically provision HTTPS for all of their sites without any user interaction. Around 200 hosting providers, including Squarespace [90], WordPress [8], OVH [74], and Google, now have built-in Let's Encrypt provisioning. As we discuss in Section 7, more than half of publicly accessible Let's Encrypt certificates are hosted by these providers. While most individual sites hosted by these providers see only relatively little traffic, an avenue for migrating them to HTTPS with little or no user interaction is necessary to transition the entire Web to HTTPS.

6.4 Embedded Devices

A growing number of embedded devices include ACME clients, including products from Asus, D-Link, Synology, and Zyxel. AVM GmbH's Fritz!Box residential gateway [18] illustrates a typical implementation. Devices are assigned a subdomain of `myfritz.net`, which acts as a dynamic DNS provider, requests a certificate from Let's Encrypt, and allows users to administer the device without needing to bypass any browser warnings. Such devices are only responsible for a small fraction of Let's Encrypt certificates.



(a) CA market share **(b) Let's Encrypt growth**

Figure 4: CA market share and Let's Encrypt's growth for Alexa Top Million domains. Since Let's Encrypt launched, it has seen more growth among top million sites than any other CA. It was used by about 35% of top million sites with HTTPS as of January 2019.

7 LET'S ENCRYPT'S USAGE AND IMPACT

Since its launch in December 2015, Let's Encrypt has steadily grown to become the largest CA in the Web PKI by certificates issued and the fourth largest known CA by Firefox Beta TLS full handshakes. As of January 21, 2019, the CA had issued a total of 538M certificates for 223M unique FQDNs, and there were 91M unexpired Let's Encrypt certificates valid for 155M unique FQDNs. This represents more unique certificates than all other CAs combined. (Except where otherwise noted, analysis in this section is based on data from that date. For Certificate Transparency data, we count certificates by the SHA-256 hash of the `tbsCertificate` structure after removing CT poison and SCTs to prevent double counting pre-certificates and certificates.) In this section, we analyze Let's Encrypt's role in the HTTPS ecosystem from a variety of perspectives including longitudinal Censys HTTPS scans [35], Certificate Transparency (CT) logs [62], Firefox client telemetry, and our own HTTPS scans of names in CT logs.

7.1 Adoption By Popular Websites

Let's Encrypt is more commonly used by the long tail of sites on the Web than by the most popular sites. Indeed, while the majority of HTTPS sites use Let's Encrypt, until recently no sites in the Alexa Top 100 used a Let's Encrypt certificate. The CA's market share increases as site popularity decreases: 5% of the top 1K, 20% of the top 100K, and 35% of the top 1M sites with HTTPS use Let's Encrypt. Only 3.6% of full TLS handshakes by Firefox Beta users are protected by Let's Encrypt certificates. Instead, trust anchors belonging to DigiCert and GlobalSign—the two CAs that have transitively issued certificates for the ten most popular sites by Alexa rank—authenticate the majority of connections (Figure 5).

One likely reason that popular sites prefer other CAs is demand for Organization Validation (OV) and Extended Validation (EV) certificates, which Let's Encrypt does not issue. Over 83% of sites in the Top 100 use OV or EV certificates, while only 18% of all trusted certificates are EV or OV validated. Given Firefox, Chrome, and Safari's moves to remove unique indicators for each validation level [53, 73, 94, 96], more sites may move to DV certificates. Stack

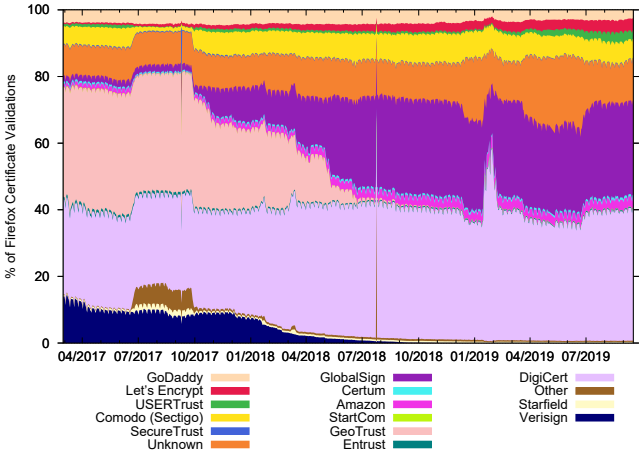


Figure 5: Firefox HTTPS connections by trust anchor. We show the trust anchors responsible for authenticating full TLS handshakes by Firefox Beta users. Let's Encrypt has become the fourth largest known CA.

Overflow (Alexa Rank 38) began using a Let's Encrypt certificate in July 2019, after our primary analysis was complete.

Let's Encrypt has seen rapidly growing adoption among top million sites since its launch, while most other CAs have not (Figure 4b). It was the fastest growing CA for those sites, increasing in market share from just 2% in July 2016 to over 25% in January 2019 (Figure 4a). By contrast, the market share of Sectigo—the second most popular CA within the top million sites—remained relatively steady, at 17%. The only other major CA that showed significant growth during this period is DigiCert, which grew from 2% to 10%. Firefox Beta channel telemetry shows Let's Encrypt increasing from 0.44% of full handshakes in the first 28 days of March 2017 to 3.7% in the 28 days ending September 17, 2019 (Figure 5).

Most sites that have adopted Let's Encrypt are new to HTTPS, but many of the most popular sites that use the Let's Encrypt previously deployed HTTPS with a different CA. Of the 94K sites consistently in the Alexa Top Million between 2015 and 2019 that now use a

Authority	Active FQDNs		Certificates	
Let's Encrypt	123.6M	(58%)	91.3M	(57%)
cPanel	45.4M	(21%)	15.8M	(10%)
Sectigo (previously Comodo)	14.9M	(7%)	10.0M	(7%)
DigiCert	8.7M	(4%)	7.1M	(4%)
Cloudflare	7.4M	(3%)	16.1M	(10%)
GoDaddy	4.2M	(2%)	4.8M	(3%)
GlobalSign	1.9M	(0.9%)	1.0M	(0.6%)
Nazwa.pl	1.0M	(0.5%)	0.9M	(0.5%)
Amazon	0.9M	(0.4%)	1.5M	(1%)
Starfield	0.9M	(0.4%)	0.3M	(0.2%)
TrustAsia	0.5M	(0.2%)	0.9M	(0.6%)
Other	3.6M	(1.7%)	11.3M	(7.0%)
Total	213M		161M	

Table 3: Most popular certificate authorities based on certificates in public CT logs and based on domains responding to HTTPS requests. Let's Encrypt has issued more certificates and is served on more unique domains than all other CAs combined.

Provider	LE	% LE	% of LE	Provider	LE	% LE	% of LE
Unified Layer	33.8M	91%	27.0%	Wix	3.9M	85%	3.2%
OVH	5.1M	60%	4.0%	Hetzner	3.0M	57%	2.4%
Amazon	4.9M	60%	4.0%	Google	2.5M	80%	2.0%
Squarespace	4.9M	97%	3.9%	PDR	2.0M	59%	1.6%
Automattic	4.3M	96%	3.5%	SingleHop	2.0M	64%	1.6%

Table 4: Providers with most Let's Encrypt domains on live websites. While automatic provisioning of Let's Encrypt certificates has explained part of its explosive growth, many of the networks with the most certificates are cloud providers where users have chosen to use Let's Encrypt.

Let's Encrypt certificate, 75% previously deployed HTTPS with a different CA. (We note that sites consistently in the Alexa Top Million are likely to be the most popular and may not represent the top million more broadly.) These include Sectigo (27%), GeoTrust (15%), GlobalSign (8%), GoDaddy (6.4%), and DigiCert (6.3%).

7.2 Automatic HTTPS Configuration

One reason that Let's Encrypt has experienced rapid growth is that its service has been integrated by hosting and CDN providers in order to automatically provision certificates for their customers. For example, the drag-and-drop website building service Wix provides HTTPS for all users using Let's Encrypt certificates [105]. There are nearly 4M unique active FQDNs with Let's Encrypt certificates that point to servers in the Wix AS (Table 4). In the most extreme case, Unified Layer, a subsidiary of Endurance International Group (EIG), which provides public hosting through Bluehost, HostGator, and several other subsidiary brands, hosts nearly 34M sites with certificates from Let's Encrypt. This accounts for 27% of the publicly accessible names found in Let's Encrypt certificates.

Half of names in Let's Encrypt certificates are located in just 10 ASes, and 80% are in 100 ASes. However, this concentration does not necessarily indicate that these providers are automatically issuing Let's Encrypt certificates for every site. Rather, this represents a broader centralization of the Web. Many of the networks with the most Let's Encrypt certificates are large cloud providers like OVH,

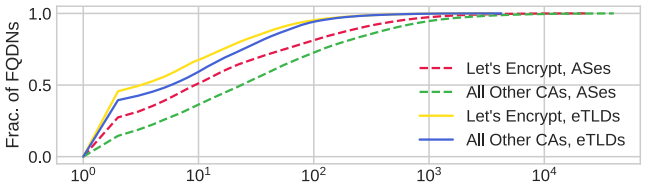


Figure 6: CDF of ASes and eTLDs for Let's Encrypt certificates. While a large number of providers automatically provision certificates using Let's Encrypt, certificates from the CA are only somewhat more concentrated (i.e., located in a small number of ASes) than those from other authorities.

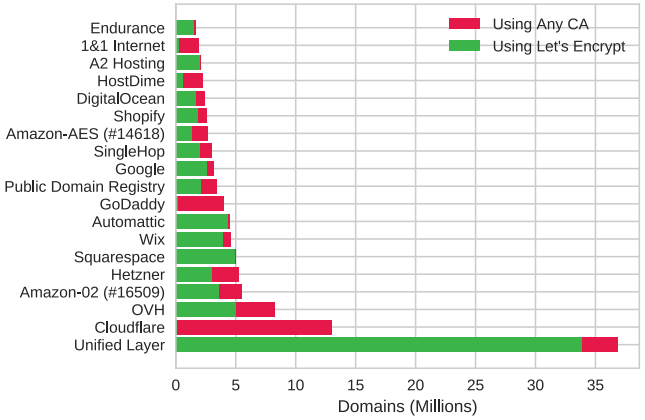


Figure 7: Share of domains using Let's Encrypt for the ASes with the most HTTPS sites. The largest, Unified Layer, represents 27% of accessible domains using Let's Encrypt certificates.

Amazon, Hetzner, and Digital Ocean, where customers are provisioning certificates from a variety of CAs (Figure 7). Of the top ten largest Let's Encrypt clients by network (i.e., ASN), only three use Let's Encrypt for over 90% of their HTTPS sites. This centralization is also present for other CAs (Figure 6). Beyond Unified Layer, the next largest provider that automatically provisions certificates for all customers is Squarespace, which accounts for under 4% of Let's Encrypt domains. In total, the largest providers that automatically provision certificates for all customers account for only 34% of Let's Encrypt names (Table 4).

7.3 Web Servers

While some Web servers like Caddy automatically provision Let's Encrypt certificates, the bulk of sites are served from traditional Web servers like Nginx and Apache. We see Let's Encrypt usage on almost all of the common Web servers, though we see significantly different levels of adoption between servers (Table 5). For example, while 85% of sites hosted on Nginx use Let's Encrypt, less than half of Apache and IIS sites do. Nginx is the most popular Web server used to host public Let's Encrypt-protected domains. Caddy is the most popular server with near 100% Let's Encrypt usage.

7.4 User Demographics

Beyond automatic provisioning, Let's Encrypt usage is broadly similar to that of other authorities by both geographic breakdown

Name	All FQDNs	LE FQDNs	% LE	% of LE
Apache	79.3M	36.7M	46%	30%
Nginx	66.0M	56.2M	85%	45%
LiteSpeed	14.6M	7.25M	49%	5.8%
Cloudflare	13.2M	31K	0.2%	0.3%
cPanel	10.8M	4.8M	43%	3.8%
Microsoft	3.9M	1.5M	38%	1.2%
OpenResty	3.5M	3.1M	87%	2.5%
DPS (GoDaddy)	1.6M	9	0%	0%
Cowboy	0.5M	0.1M	25%	0.1%
GSE	0.5M	0.4M	81%	0.3%
Other	2.4M	1.7M	70%	1.4%
Not Specified	18.5M	13.7M	74%	11%

Table 5: Most common server types. Domains that responded to HTTPS requests on January 29, 2019, grouped by type of server and categorized as using Let's Encrypt or another CA. Apache is the most popular server overall, but Nginx is the most popular among Let's Encrypt deployments.

eTLD	LE	% LE	% of LE	eTLD	LE	% LE	% of LE
.com	44.3M	52%	48.5%	.nl	1.9M	71%	2.1%
.de	3.9M	69%	4.3%	.br	1.8M	72%	1.9%
.org	3.3M	68%	3.6%	.fr	1.7M	82%	1.8%
.uk	2.5M	62%	2.7%	.com.br	1.6M	73%	1.8%
.co.uk	2.2M	63%	2.4%	.ru	1.3M	79%	1.4%

Table 6: eTLDs with most Let's Encrypt certificates based on valid certificates in the Censys dataset.

and TLD. For example, the most common TLDs are largely the same as for other CAs, though we see differing adoption rates in some countries (Table 6). For example, while only 49% of `.com` domains use Let's Encrypt, around 80% of `.ru` and `.fr` domains do.

There are several public suffixes with a disproportionate number of Let's Encrypt certificates. For example, 30 suffixes have more than 100,000 domains and greater than 95% Let's Encrypt adoption. These fall into several broad categories:

Blog and hosting providers. Several large blog providers that create a unique subdomain for each blog they host. For example, there are certificates for 1.3M subdomains of home.blog and 287K domains under automattic.com.

IoT devices. There are a handful of IoT manufacturers who create a subdomain and certificate for each deployed IoT device. For example, there are 875K domains under keenetic.io. We also see remotewd.com (384K), freeboxos.fr (182K), and myfritz.net (237K). For these cases, over 99% of subdomains use Let's Encrypt.

TLDs. There are three TLDs with near 100% LE usage: `.blog` (2.3M), `.jobs` (165K), and `.ir` (792K). Let's Encrypt is one of a small number of CAs that issue certificates for names in Iran's TLD.

These domains illustrate several new HTTPS use cases and populations of users enabled by automated certificate issuance.

7.5 Certificate Renewals

Compared to other sites, fewer sites with Let's Encrypt certificates serve expired certificates, and few Let's Encrypt renewals (2.9%) occur after the prior certificate has expired. Only 2.2% of sites in the Top Million with Let's Encrypt certificates currently serve expired certificates, while 3.9% of all HTTPS sites have expired certificates. Most renewals occur in the last 30 days before the validity period of a certificate expires (64%), but over a third of renewals occur in the early (19%) and middle (16%) periods of first 30 days and 30–60 days, respectively. This indicates that organizations are able to maintain an improved security posture through automation despite shorter certificate lifespans than other authorities offer.

8 DISCUSSION AND SECURITY LESSONS

When we started work on Let's Encrypt, the two most commonly voiced criticisms about the Web PKI were that (a) it was too difficult for server operators to use, and (b) it wasn't secure anyway. Let's Encrypt was intended to take aim directly at the first complaint, based on our belief that the usability problem was the more serious and that it was responsible for the relatively low deployment of HTTPS. The data in Section 7 (and particularly in Figure 4b) suggest that this analysis was correct: Let's Encrypt has been responsible for significant growth in HTTPS deployment.

By contrast, Let's Encrypt has had only an indirect impact on the security of the HTTPS ecosystem itself. Ultimately, the security of certificates is dictated by that of the weakest CA, and security only improves when all CAs do a better job. In parallel with our efforts, browser-makers and security advocates within the Web PKI community have been working to increase PKI security through tightened requirements for CAs, promotion of new security mechanisms such as Certificate Transparency, and enforcement of greater CA transparency and accountability. Let's Encrypt has been an eager participant in these changes, which we consider to have been quite productive, and has attempted to set an example of good PKI citizenship, including through its commitment to openness and its record of fast and complete incident disclosure.

8.1 Why Was Let's Encrypt Successful?

At some level, the answer to Let's Encrypt's success is easy: it was free and easy to use (and in fact automated). While some previous CAs such as StartCom had free tiers and others had some level of automation, no previous CA had attempted to combine these two into a single service offering. These properties turn out to be strongly interdependent: Automation is necessary to have a free CA and free certificates make automation practical.

Automation enables free certificates. The dependency of free certificates on automation is relatively obvious: if certificates are free and your intent is to issue millions of certificates, then it is critically important to keep per-certificate costs down; automation is the only plausible mechanism for doing so. Removing humans from the validation process also reduces the possibility that social engineering or simple misjudgment will lead to misissuance, both of which are sources of risk for a CA on a limited budget.

In addition to lowering the direct monetary cost of certificates, automation lowers the cost to administrators of managing them. Manual management, especially of large server farms, is inherently expensive and also introduces the risk of configuration errors—such as failure to renew certificates—that can lead to downtime. Together with free certificates, the net impact is a significant lowering of the overall cost of serving HTTPS.

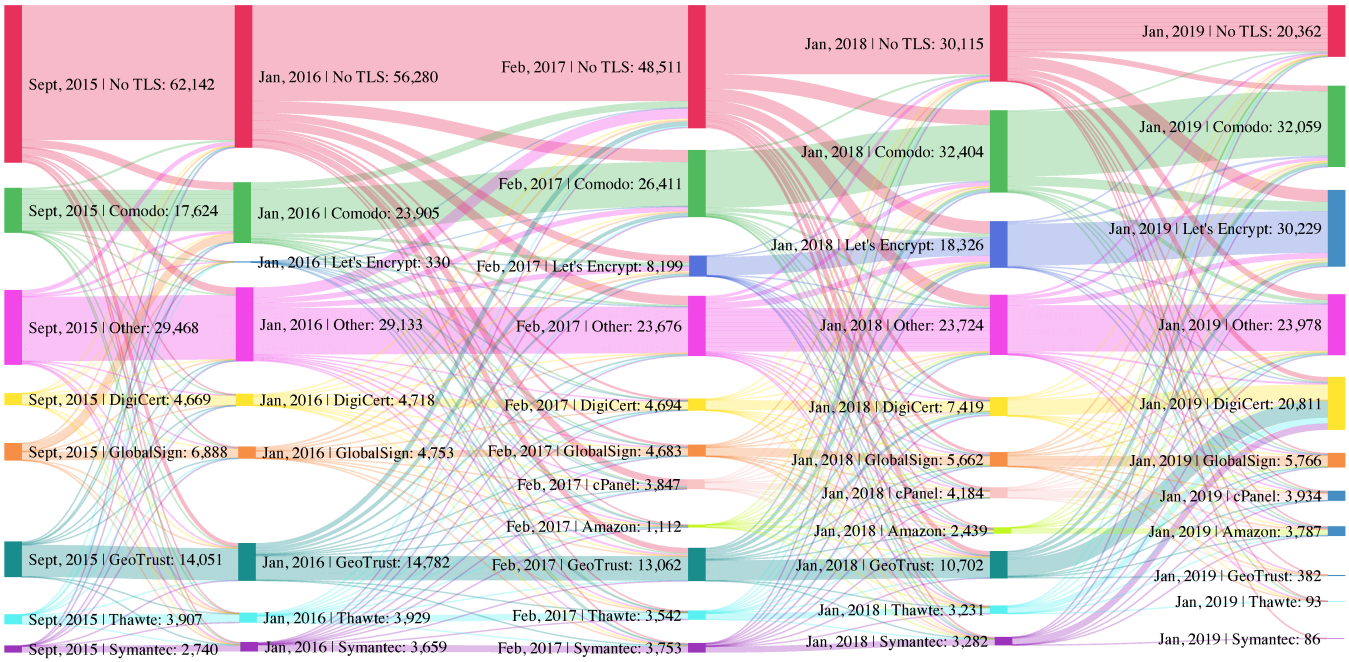


Figure 8: Certificate authority flow among stable, popular sites. We track CA choice for 141K domains over five snapshots, from 7/2015 to 1/2019. The included sites are those that were ranked in the Alexa Top Million at every snapshot, and so are likely more popular and long-lived than the top million overall.

Free certificates make automation practical. Although it is technically possible to have automation with non-free certificates, the requirement for payment makes automation significantly less valuable. This is most obvious in the case of large hosting providers such as Automattic or Squarespace, which have been responsible for a large fraction of the use of Let's Encrypt (see Section 4). Each of these providers made use of Let's Encrypt's APIs to automatically issue certificates for large fractions of their users. This would have been more difficult, if not impossible for most providers, if they had to pay for each certificate.

Even for smaller operators, free is an enabler. It allows for initial setup without arranging for payment and then "fire and forget" configuration without concerns that the credit card will become invalid and result in reissuance failures.

Gradual deployment is essential. Let's Encrypt is far from the only attempt to provide universal free server authentication, but it is the first to see widespread adoption. Other efforts include DNS Authentication of Named Entities (DANE) [19], Sovereign Keys [39], and MonkeySphere [70]. As a concrete comparison, the DNSSEC root zone was signed in 2010 and DANE was published in 2012—three years before Let's Encrypt launched—yet there is essentially no server deployment and no major Web client supports DANE. While DANE suffers from a number of deficiencies [60], the primary problem is hinted at by "no major Web client". In order for servers to use DANE, clients first have to change, but, conversely, without any servers, clients have no incentive to support it. This is also true for Sovereign Keys and other alternative proposals for Web server authentication. By contrast, because Let's Encrypt issued ordinary Web PKI certificates that, thanks to the cross-signature, were immediately valid in nearly any browser, it was able to deploy rapidly without asking both sides to change.

There is an important lesson here for the future. The application of blockchain technologies for cryptocurrency has led to a number of proposals for rooting identity in a blockchain, so that it would be possible to directly verify ownership of a given name without requiring some third party to verify and then vouch for that ownership, as with DV certificates. However, as with previous proposals, actual deployment of such a system would require every client to change, and until that happens, servers would need an ordinary Web PKI certificate in any case. If history is any guide, this chicken-and-egg problem will stall deployment. One potential solution would be to issue Web PKI certificates but to attach proofs of ownership rooted in the blockchain-based identity system. This would allow all clients to talk to those servers while clients and Certificate Transparency monitors could validate that those certificates had been properly issued and report any misissuance.

8.2 Remaining Security Concerns

As noted above, Let's Encrypt has increased HTTPS deployment but has not directly made the Web PKI itself appreciably more secure. Important security challenges remain, including that domain validation as a whole remains far from as secure as we would like it to be. Moreover, debate continues about whether Let's Encrypt has had a negative impact on some aspects of Web PKI security.

Automation has improved but not solved validation. While a fully-automated validation process has eliminated many of the human sources of error that plagued other certificate authorities, the approach is not foolproof.

One major security objective we have not yet accomplished with Let's Encrypt is strongly protecting validation against DNS- and routing-layer attacks. An attacker who can use DNS or BGP hijacking to redirect traffic—or who can compromise a network device close to the server or the CA—can intercept domain validation traffic and falsely request a certificate [26, 52, 84]. Domain owners can make some of these attacks more difficult by limiting the CAs that can issue for them using CAA [49], and vigilant operators can use CT monitors (e.g., [41]) to detect false issuances and respond.

As a further mitigation against route hijacking and man-in-the-middle attacks, Let's Encrypt is experimenting with multiple perspective validation. In this approach (similar to [104]), control of the domain is simultaneously verified from multiple vantage points in different autonomous systems, a strong majority of which must succeed for the certificate to be issued.

Unfortunately, none of these measures can protect against the full spectrum of validation attacks. The fundamental problem is that domain validation is itself not cryptographically protected, since it is the bootstrapping mechanism by which sites join the PKI. After a domain has been validated once, it might seem sensible to give it some way to disable future non-cryptographic validation methods, but, as with HPKP [50], this risks creating downtime for sites that lose their validation keys—or, worse, when attackers temporarily take over a domain and change the key to one they control. There may be no easy solution.

Phishing remains a challenge for the Web in general. Probably the most frequent complaint about Let's Encrypt is that it is used in the perpetration of phishing attacks. By some measures, more than half of phishing sites now use HTTPS [56], as do many sites that distribute malware, and a large number of those sites use certificates issued by Let's Encrypt [34]. Some observers have called for CAs to take a more active role in combating such sites.

In our view, CAs are not well positioned to detect phishing and malware campaigns, or to police content more generally. They simply do not have sufficient ongoing visibility into sites' content, which can change much faster than certificate issuance and revocation cycles. As a result, certificates cannot offer assurances related to the safety of Web content.

Attempts to limit certificates (and thus HTTPS) to domains with entirely safe content are likely to be highly problematic. Some sites will be denied service because of false positives or questionable definitions of what constitutes safe content. Another problem is that the only enforcement mechanism CAs have is to deny service to entire domains. Should a major global news site have its certificates revoked because a single ad on a single page had malware embedded? Finer-grained mechanisms for protecting users are needed. Let's Encrypt once checked Google's Safe Browsing API before certificate issuance, but it has stopped doing so for these reasons.

Browsers and search engines have much greater content awareness, and they can protect users at the page level (or better). For instance, Google Safe Browsing [89] uses machine learning to continuously detect malicious content. The results are used to warn Chrome users when they try to load pages with unsafe content. Users are much better informed and protected when browsers include such anti-phishing and anti-malware features.

Fortunately, widespread HTTPS deployment has made it possible for browsers to change their security indicator UIs in ways that reduce the risk of user confusion. Rather than showing a positive security indicator for HTTPS (which users might mistake for a "seal of approval" on the site's content), Chrome, Firefox, and other browsers have begun to show negative security indicators for HTTP sites [83]. This also further encourages sites to adopt HTTPS.

9 CONCLUSION

In this paper, we described how we created Let's Encrypt, a free, open, and automated HTTPS certificate authority (CA) designed to accelerate universal adoption of HTTPS. We presented the architecture of the CA software system (Boulder) and the structure of the organization that operates it (ISRG). We also described the design of ACME, the IETF-standard protocol we created to automate CA–server interactions and certificate issuance. Finally, we measured the CA's impact on the Web and the CA ecosystem.

Prior to our work, a major barrier to wider HTTPS adoption was that deploying it was complicated, expensive, and error-prone for server operators. Let's Encrypt overcomes these through a strategy of automation: identity validation, certificate issuance, and server configuration are fully robotic, which also results in low marginal costs and enables the CA to provide certificates at no charge.

We designed Let's Encrypt to scale to the size of the entire Web. In just over three years of operation, it is well on its way: it has issued over 538 million certificates and accounts for more valid browser-trusted certificates than all other CAs combined. We hope that in the near future, clients will start using HTTPS as the default Web transport. Eventually, we may marvel that there was ever a time when Web traffic traveled over the Internet as plaintext.

ACKNOWLEDGMENTS

We thank the entire staff at Let's Encrypt as well as the hundreds of community members who have helped make this project successful—without them, the Web would be far less secure. We also thank J.C. Jones and the members of the IETF ACME working group. Mark Reid and Daniel Thorn provided assistance with Firefox data. This material is based upon work supported by the U.S. National Science Foundation under Awards CNS-1345254, CNS-1409505, CNS-1518888, and CNS-1823192, and by the Alfred P. Sloan Research Fellowship. Let's Encrypt gratefully acknowledges its sponsors and donors, who are listed at https://letsencrypt.org/sponsors/.

REFERENCES

[1] Josh Aas. 2014. Let's Encrypt: Delivering SSL/TLS Everywhere. Let's Encrypt Blog. https://letsencrypt.org/2014/11/18/announcing-lets-encrypt.html
[2] Josh Aas. 2015. Entering Public Beta. Let's Encrypt Blog. https://letsencrypt.org/2015/12/03/entering-public-beta.html
[3] Josh Aas. 2015. Let's Encrypt is Trusted. Let's Encrypt Blog. https://letsencrypt.org/2015/10/19/lets-encrypt-is-trusted.html
[4] Josh Aas. 2015. Our First Certificate Is Now Live. Let's Encrypt Blog. https://letsencrypt.org/2015/09/14/our-first-cert.html
[5] Josh Aas. 2018. Issue with TLS-SNI-01 and Shared Hosting Infrastructure. Let's Encrypt Community Forum. https://community.letsencrypt.org/t/2018-01-09-issue-with-tls-sni-01-and-shared-hosting-infrastructure/49996
[6] Josh Aas. 2018. Let's Encrypt Root Trusted By All Major Root Programs. Let's Encrypt Blog. https://letsencrypt.org/2018/08/06/trusted-by-all-major-root-programs.html
[7] Josh Aas. 2019. Transitioning to ISRG's Root. Let's Encrypt Blog. https://letsencrypt.org/2019/04/15/transitioning-to-isrg-root.html
[8] Barry Abrahamson. 2016. HTTPS Everywhere: Encryption for All WordPress.com Sites. The WordPress.com Blog. https://en.blog.wordpress.com/2016/04/08/https-everywhere-encryption-for-all-wordpress-com-sites/

[9] Mustafa Emre Acer, Emily Stark, Adrienne Porter Felt, Sascha Fahl, Radhika Bhargava, Bhanu Dev, Matt Braithwaite, Ryan Sleevi, and Parisa Tabriz. 2017. Where the wild warnings are: Root causes of Chrome HTTPS certificate errors. In *24th ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*. ACM, New York, NY, USA, 1407–1420. https://doi.org/10.1145/3133956.3134007

[10] C. Adams, S. Farrell, T. Kause, and T. Mononen. 2005. *Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)*. RFC 4210. IETF. https://doi.org/10.17487/RFC4210

[11] David Adrian, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J. Alex Halderman, Nadia Heninger, Drew Springall, Emmanuel Thomé, Luke Valenta, Benjamin VanderSloot, Eric Wustrow, Santiago Zanella-Béguelin, and Paul Zimmermann. 2015. Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice. In *22nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15)*. ACM, New York, NY, USA, 5–17. https://doi.org/10.1145/2810103.2813707

[12] Akamai. 2018. Certificate Provisioning System API v2. Akamai Developer. https://developer.akamai.com/api/core_features/certificate_provisioning_system/v2.html

[13] Nadhem J. Al Fardan and Kenneth G. Paterson. 2013. Lucky Thirteen: Breaking the TLS and DTLS Record Protocols. In *34th IEEE Symposium on Security and Privacy (S&P '13)*. IEEE, New York, NY, USA, 526–540. https://doi.org/10.1109/SP.2013.42

[14] Johanna Amann, Oliver Gasser, Quirin Scheitle, Lexi Brent, Georg Carle, and Ralph Holz. 2017. Mission Accomplished? HTTPS Security After DigiNotar. In *17th ACM Internet Measurement Conference (IMC '17)*. ACM, New York, NY, USA, 325–340. https://doi.org/10.1145/3131365.3131401

[15] Apache HTTP Server Project. 2019. Apache Module mod_md. Apache HTTP Server Version 2.5. https://httpd.apache.org/docs/trunk/mod/mod_md.html

[16] ASUS. 2019. How to Enable HTTPS and Create a Certificate by Using ASUS Let's Encrypt? Support FAQ. https://www.asus.com/us/support/FAQ/1034294/

[17] Nimrod Aviram, Sebastian Schinzel, Juraj Somorovsky, Nadia Heninger, Maik Dankel, Jens Steube, Luke Valenta, David Adrian, J. Alex Halderman, Viktor Dukhovni, Emilia Käsper, Shaanan Cohney, Susanne Engels, Christof Paar, and Yuval Shavitt. 2016. DROWN: Breaking TLS Using SSLv2. In *25th USENIX Security Symposium (USENIX Security '16)*. USENIX Association, Berkeley, CA, USA, 689–706. https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_aviram.pdf

[18] AVM GmbH. 2019. MyFRITZ! Internetzugriff auf Ihre FRITZ!Box nun ohne Sicherheitshinweise im Browser. FRITZ! Labor. https://avm.de/fritz-labor/weitere-produkte/neues-verbesserungen/lets-encrypt/

[19] Richard Barnes. 2011. *Use Cases and Requirements for DNS-Based Authentication of Named Entities (DANE)*. RFC 6394. IETF. https://doi.org/10.17487/RFC6394

[20] Richard Barnes, Jacob Hoffman-Andrews, Daniel McCarney, and James Kasten. 2019. *Automated Certificate Management Environment (ACME)*. RFC 8555. IETF. https://doi.org/10.17487/RFC8555

[21] Adam Barth. 2011. *The Web Origin Concept*. RFC 6454. IETF. https://doi.org/10.17487/RFC6454

[22] Matthew Bernhard, Jonathan Sharman, Claudia Ziegler Acemyan, Philip Kortum, Dan Wallach, and J. Alex Halderman. 2019. On the Usability of HTTPS Deployment. In *2019 ACM Conference on Human Factors in Computing Systems (CHI '19)*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3290605.3300540

[23] Benjamin Beurdouche, Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Markulf Kohlweiss, Alfredo Pironti, Pierre-Yves Strub, and Jean Karim Zinzindohoue. 2015. A Messy State of the Union: Taming the Composite State Machines of TLS. In *36th IEEE Symposium on Security and Privacy (S&P '15)*. IEEE, New York, NY, USA, 535–552. https://doi.org/10.1109/SP.2015.39

[24] Karthikeyan Bhargavan, Antoine Delignat-Lavaud, and Nadim Kobeissi. 2017. Formal Modeling and Verification for Domain Validation and ACME. In *21st International Conference on Financial Cryptography and Data Security (FC '17)*. Springer, New York, NY, USA, 561–578. https://doi.org/10.1007/978-3-319-70972-7_32

[25] Henry Birge-Lee, Yixin Sun, Annie Edmundson, Jennifer Rexford, and Prateek Mittal. 2017. Using BGP to Acquire Bogus TLS Certificates. In *HotPETS 2017*. 2 pages. https://petsymposium.org/2017/papers/hotpets/bgp-bogus-tls.pdf

[26] Henry Birge-Lee, Yixin Sun, Anne Edmundson, Jennifer Rexford, and Prateek Mittal. 2018. Bamboozling Certificate Authorities with BGP. In *27th USENIX Security Symposium (USENIX Security '18)*. USENIX Association, Berkeley, CA, USA, 833–849. https://www.usenix.org/system/files/conference/usenixsecurity18/sec18-birge-lee.pdf

[27] Simon Blake-Wilson, Nelson Bolyard, Vipul Gupta, Chris Hawk, and Bodo Moeller. 2006. *Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)*. RFC 4492. IETF. https://doi.org/10.17487/RFC4492

[28] CA/Browser Forum. 2019. Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates. Version 1.6.5. https://cabforum.org/wp-content/uploads/CA-Browser-Forum-BR-1.6.5.pdf

[29] CA/Browser Forum. 2019. Guidelines For The Issuance And Management Of Extended Validation Certificates. Version 1.6.9. https://cabforum.org/wp-content/uploads/CA-Browser-Forum-EV-Guidelines-v1.6.9.pdf

[30] Jeff Chester. 2018. AT&T, Comcast and Verizon Expand "Big Data" Tracking and Targeting of Consumers. Center for Digital Democracy Blog. https://www.democraticmedia.org/blog/att-comcast-verizon-expand-big-data-tracking-targeting-consumers

[31] Comodo. 2015. SSL Certificate with Free Trust Logo. Internet Archive version. https://web.archive.org/web/20150421032800/https://ssl.comodo.com/

[32] DigiCert. 2015. SSL Certificate Comparison. Internet Archive version. https://web.archive.org/web/20150905120255/https://www.digicert.com/ssl-certificate-comparison.htm

[33] DigiCert. 2019. Standard SSL Certificates. Retrieved September 25, 2019 from https://www.digicert.com/standard-ssl-certificates/

[34] Vincent Drury and Ulrike Meyer. 2019. Certified Phishing: Taking a Look at Public Key Certificates of Phishing Websites. In *15th Symposium on Usable Privacy and Security (SOUPS '19)*. USENIX Association, Berkeley, CA, USA, 211–223. https://www.usenix.org/system/files/soups2019-drury.pdf

[35] Zakir Durumeric, David Adrian, Ariana Mirian, Michael Bailey, and J. Alex Halderman. 2015. Censys: A Search Engine Backed by Internet-Wide Scanning. In *22nd ACM Conference on Computer and Communications Security (CCS '15)*. ACM, New York, NY, USA, 542–553. https://doi.org/10.1145/2810103.2813703

[36] Zakir Durumeric, James Kasten, Michael Bailey, and J Alex Halderman. 2013. Analysis of the HTTPS Certificate Ecosystem. In *13th ACM Internet Measurement Conference (IMC '13)*. ACM, New York, NY, USA, 291–304. https://doi.org/10.1145/2504730.2504755

[37] Zakir Durumeric, Frank Li, James Kasten, Johanna Amann, Jethro Beekman, Mathias Payer, Nicolas Weaver, David Adrian, Vern Paxson, Michael Bailey, and J. Alex Halderman. 2014. The Matter of Heartbleed. In *14th ACM Internet Measurement Conference (IMC '14)*. ACM, New York, NY, USA, 475–488. https://doi.org/10.1145/2663716.2663755

[38] Donald Eastlake. 2011. *Transport Layer Security (TLS) Extensions: Extension Definitions*. RFC 6066. IETF. https://doi.org/10.17487/RFC6066

[39] Peter Eckersley. 2011. Sovereign Key Cryptography for Internet Domains. https://github.com/EFForg/sovereign-keys/blob/master/sovereign-key-design.txt

[40] Electronic Frontier Foundation. 2019. Certbot. https://certbot.eff.org/

[41] Facebook. 2019. Certificate Transparency Monitoring. Facebook for Developers. https://developers.facebook.com/tools/ct/search/

[42] Stephan Friedl, Andrei Popov, Adam Langley, and Emile Stephan. 2014. *Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension*. RFC 7301. IETF. https://doi.org/10.17487/RFC7301

[43] GeoTrust. 2015. Compare SSL Certificates. Internet Archive version. https://web.archive.org/web/20150319134219/https://www.geotrust.com/ssl/compare-ssl-certificates.html

[44] GeoTrust. 2019. SSL Certificates. Retrieved September 25, 2019 from https://www.geotrust.com/ssl/

[45] GoDaddy. 2015. SSL Certificates. Internet Archive version. https://web.archive.org/web/20150601032030/https://www.godaddy.com/ssl/ssl-certificates.aspx

[46] GoDaddy. 2019. SSL Certificates. Retrieved September 25, 2019 from https://www.godaddy.com/web-security/ssl-certificate

[47] Google. 2019. Transparency Report: HTTPS Encryption by Chrome Platform. Retrieved May 2019 from https://transparencyreport.google.com/https/overview

[48] Ilya Grigorik. 2019. Is TLS Fast Yet? https://istlsfastyet.com

[49] Phillip Hallam-Baker and Rob Stradling. 2013. *DNS Certification Authority Authorization (CAA) Resource Record*. RFC 6844. IETF. https://doi.org/10.17487/RFC6844

[50] Scott Helme. 2017. I'm Giving Up on HPKP. https://scotthelme.co.uk/im-giving-up-on-hpkp/

[51] Kipp E.B. Hickman. 1995. SSL 0.2 Protocol Specification. https://www-archive.mozilla.org/projects/security/pki/nss/ssl/draft02.html

[52] Muks Hirani, Sarah Jones, and Ben Read. 2019. Global DNS Hijacking Campaign: DNS Record Manipulation at Scale. FireEye Threat Research. https://www.fireeye.com/blog/threat-research/2019/01/global-dns-hijacking-campaign-dns-record-manipulation-at-scale.html

[53] Troy Hunt. 2019. Extended Validation Certificates are (Really, Really) Dead. https://www.troyhunt.com/extended-validation-certificates-are-really-really-dead/

[54] ISRG. 2019. About Internet Security Research Group: Board of Directors. https://www.abetterinternet.org/about/

[55] James D. Kasten. 2015. *Server Authentication on the Past, Present, and Future Internet*. Ph.D. Dissertation. University of Michigan, Ann Arbor, MI, USA. Advisor(s) J. Alex Halderman. https://doi.org/2027.42/116632

[56] Brian Krebs. 2018. Half of All Phishing Sites Now Have the Padlock. Krebs on Security. https://krebsonsecurity.com/2018/11/half-of-all-phishing-sites-now-have-the-padlock/

[57] Katharina Krombholz, Wilfried Mayer, Martin Schmiedecker, and Edgar Weippl. 2017. "I Have No Idea What I'm Doing": On the Usability of Deploying HTTPS. In *26th USENIX Security Symposium (USENIX Security '17)*. USENIX Association, Berkeley, CA, USA, 1339–1356. https://www.usenix.org/system/files/

conference/usenixsecurity17/sec17-krombholz.pdf
[58] Deepak Kumar, Zhengping Wang, Matthew Hyder, Joseph Dickinson, Gabrielle Beck, David Adrian, Joshua Mason, Zakir Durumeric, J. Alex Halderman, and Michael Bailey. 2018. Tracking Certificate Misissuance in the Wild. In *39th IEEE Symposium on Security and Privacy (S&P '18)*. IEEE, New York, NY, USA, 785–798. https://doi.org/10.1109/SP.2018.00015
[59] Adam Langley. 2010. Overclocking SSL. ImperialViolet blog. https://www.imperialviolet.org/2010/06/25/overclocking-ssl.html
[60] Adam Langley. 2015. Why Not DANE in Browsers. ImperialViolet blog. https://www.imperialviolet.org/2015/01/17/notdane.html
[61] James Larisch, David Choffnes, Dave Levin, Bruce M. Maggs, Alan Mislove, and Christo Wilson. 2017. CRLite: A Scalable System for Pushing All TLS Revocations to All Browsers. In *38th IEEE Symposium on Security and Privacy (S&P '17)*. IEEE, New York, NY, USA, 539–556. https://doi.org/10.1109/SP.2017.17
[62] Ben Laurie, Adam Langley, and Emilia Kasper. 2013. *Certificate Transparency*. RFC 6962. IETF. https://doi.org/10.17487/RFC6962
[63] Let's Encrypt. 2016. Certificate Compatibility. https://letsencrypt.org/docs/certificate-compatibility/
[64] Let's Encrypt. 2018. Let's Encrypt Privacy Policy. https://letsencrypt.org/privacy/
[65] Let's Encrypt. 2019. ACME Client Implementations. https://letsencrypt.org/docs/client-options/
[66] Let's Encrypt. 2019. Policy and Legal Repository. https://letsencrypt.org/repository/
[67] Light Code Labs. 2019. Caddy: The HTTP/2 Server With Automatic HTTPS. https://caddyserver.com/
[68] Bill Marczak, Nicholas Weaver, Jakub Dalek, Roya Ensafi, David Fifield, Sarah McKune, Arn Rey, John Scott-Railton, Ron Deibert, and Vern Paxson. 2015. An Analysis of China's "Great Cannon". In *5th USENIX Workshop on Free and Open Communications on the Internet (FOCI '15)*. USENIX Association, Berkeley, CA, USA, 11 pages. https://www.usenix.org/system/files/conference/foci15/foci15-paper-marczak.pdf
[69] Bodo Möller, Thai Duong, and Krzysztof Kotowicz. 2014. This POODLE Bites: Exploiting the SSL 3.0 Fallback. https://www.openssl.org/~bodo/ssl-poodle.pdf
[70] Monkeysphere. 2010. The Monkeysphere Project. https://web.monkeysphere.info/
[71] Parker Moore. 2018. Custom Domains on GitHub Pages Gain Support for HTTPS. The GitHub Blog. https://github.blog/2018-05-01-github-pages-custom-domains-https/
[72] Magnus Nystrom and Burt Kaliski. 2000. *PKCS #10: Certification Request Syntax Specification Version 1.7*. RFC 2986. IETF. https://doi.org/10.17487/RFC2986
[73] Devon O'Brien. 2019. Upcoming Change to Chrome's Identity Indicators. Chromium security-dev mailing list. https://groups.google.com/a/chromium.org/forum/m/#!msg/security-dev/h1bTcoTpfeI/jUTk1z7VAAAJ
[74] OVH. 2019. Certificats SSL pour tous avec Let's Encrypt! https://www.ovh.com/fr/hebergement-web/ssl_mutualise.xml
[75] OVH. 2019. SSL Gateway: HTTPS for All. https://www.ovh.com/world/ssl-gateway/
[76] Plesk. 2019. Let's Encrypt Extension. https://www.plesk.com/extensions/letsencrypt/
[77] Max Pritikin, Peter Yee, and Dan Harkins. 2013. *Enrollment over Secure Transport*. RFC 7030. IETF. https://doi.org/10.17487/RFC7030
[78] Eric Rescorla. 2000. *HTTP over TLS*. RFC 2818. https://doi.org/10.17487/RFC2818
[79] Eric Rescorla. 2018. *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC 8446. IETF. https://doi.org/10.17487/RFC8446
[80] Paul Roberts. 2011. Phony SSL Certificates Issued for Google, Yahoo, Skype, Others. Threatpost. https://threatpost.com/phony-ssl-certificates-issued-google-yahoo-skype-others-032311/75061/
[81] Franz Rosén. 2018. How I Exploited ACME TLS-SNI-01 Issuing Let's Encrypt SSL-Certs for Any Domain Using Shared Hosting. Detectify Labs. https://labs.detectify.com/2018/01/12/how-i-exploited-acme-tls-sni-01-issuing-lets-encrypt-ssl-certs-for-any-domain-using-shared-hosting/
[82] Jim Schaad and Michael Myers. 2008. *Certificate Management over CMS (CMC)*. RFC 5272. IETF. https://doi.org/10.17487/RFC5272
[83] Emily Schechter. 2018. Evolving Chrome's Security Indicators. Chromium Blog. https://blog.chromium.org/2018/05/evolving-chromes-security-indicators.html
[84] Lorenz Schwittmann, Matthäus Wander, and Torben Weis. 2019. Domain Impersonation is Feasible: A Study of CA Domain Validation Vulnerabilities. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P '19)*. IEEE, New York, NY, USA, 544–559. https://doi.org/10.1109/EuroSP.2019.00046
[85] Sectigo. 2019. Sectigo Domain Validation Certificates. Retrieved September 25, 2019 from https://sectigo.com/ssl-certificates-tls/dv-domain-validation
[86] Nicolas Serrano, Hilda Hadan, and L. Jean Camp. 2019. A Complete Study of P.K.I. (PKI's Known Incidents). Available at SSRN. https://doi.org/10.2139/ssrn.3425554
[87] Roland Shoemaker. 2018. *ACME TLS ALPN Challenge Extension*. Internet-Draft draft-ietf-acme-tls-alpn-05. IETF. https://www.ietf.org/internet-drafts/draft-ietf-acme-tls-alpn-05.txt
[88] Christopher Soghoian and Sid Stamm. 2011. Certified Lies: Detecting and Defeating Government Interception Attacks Against SSL. In *15th International Conference on Financial Cryptography and Data Security*. Springer, New York, NY, USA, 250–259. https://doi.org/10.1007/978-3-642-27576-0_20
[89] Stephan Somogyi and Allison Miller. 2017. Safe Browsing: Protecting More Than 3 Billion Devices Worldwide, Automatically. Google Blog. https://www.blog.google/technology/safety-security/safe-browsing-protecting-more-3-billion-devices-worldwide-automatically/
[90] Squarespace. 2019. Squarespace and SSL. Squarespace Help. https://support.squarespace.com/hc/en-us/articles/205815898
[91] SSL Shopper. 2019. SSL Certificate Comparison and Reviews. https://www.sslshopper.com
[92] Symantec. 2015. Buy and Compare SSL Certificates. Internet Archive version. https://web.archive.org/web/20150610013911/http://www.symantec.com/page.jsp?id=compare-ssl-certificates
[93] Symantec. 2019. TLS/SSL Certificates. Retrieved September 25, 2019 from https://www.websecurity.symantec.com/ssl-certificate
[94] Wayne Thayer and Johann Hofmann. 2019. Intent to Ship: Move Extended Validation Information Out of the URL Bar. firefox-dev mailing list. https://groups.google.com/forum/m/?fromgroups&hl=en#!topic/firefox-dev/6wAg_PpnlY4
[95] Kurt Thomas, Elie Bursztein, Chris Grier, Grant Ho, Nav Jagpal, Alexandros Kapravelos, Damon McCoy, Antonio Nappa, Vern Paxson, Paul Pearce, Neil Provos, and Moheeb Abu Rajab. 2015. Ad Injection at Scale: Assessing Deceptive Advertisement Modifications. In *36th IEEE Symposium on Security and Privacy (S&P '15)*. IEEE, New York, NY, USA, 151–167. https://doi.org/10.1109/SP.2015.17
[96] Christopher Thompson, Martin Shelton, Emily Stark, Maximilian Walker, Emily Schechter, and Adrienne Porter Felt. 2019. The Web's Identity Crisis: Understanding the Effectiveness of Website Identity Indicators. In *28th USENIX Security Symposium (USENIX Security '19)*. USENIX Association, Berkeley, CA, USA, 1715–1732. https://www.usenix.org/system/files/sec19-thompson.pdf
[97] Emin Topalovic, Brennan Saeta, Lin-Shung Huang, Collin Jackson, and Dan Boneh. 2012. Towards Short-Lived Certificates. In *2012 Web 2.0 Security and Privacy Workshop (W2SP '12)*. IEEE, New York, NY, USA, 9 pages. https://www.ieee-security.org/TC/W2SP/2012/papers/w2sp12-final9.pdf
[98] U.S. Department of the Treasury. 2019. Specially Designated Nationals and Blocked Persons List (SDN) Human Readable Lists. https://www.treasury.gov/resource-center/sanctions/SDN-List/Pages/default.aspx
[99] U.S. National Security Agency. 2008. XKEYSCORE. Classified presentation leaked by Edward Snowden. https://www.theguardian.com/world/interactive/2013/jul/31/nsa-xkeyscore-program-full-presentation
[100] Narseo Vallina-Rodriguez, Johanna Amann, Christian Kreibich, Nicholas Weaver, and Vern Paxson. 2014. A Tangled Mass: The Android Root Certificate Stores. In *10th ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT '14)*. ACM, New York, NY, USA, 141–148. https://doi.org/10.1145/2674005.2675015
[101] Benny Vasquez. 2016. Announcing: cPanel & WHM's Official Let's Encrypt with AutoSSL Plugin. cPanel Blog. https://blog.cpanel.com/announcing-cpanel-whms-official-lets-encrypt-with-autossl-plugin/
[102] Webmin. 2018. Let's Encrypt. Webmin Documentation. https://doxfer.webmin.com/Webmin/Let%27s_Encrypt
[103] WebTrust/PKI Assurance Task Force. 2019. WebTrust for Certification Authorities. https://www.cpacanada.ca/-/media/site/operational/ms-member-services/docs/webtrust/webtrust-for-ca-22.pdf
[104] Dan Wendlandt, David G. Andersen, and Adrian Perrig. 2008. Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing. In *2008 USENIX Annual Technical Conference (ATC '08)*. USENIX Association, Berkeley, CA, USA, 321–334. https://www.usenix.org/legacy/events/usenix08/tech/full_papers/wendlandt/wendlandt.pdf
[105] Wix. 2019. Enabling HTTPS for Your Wix Site. Wix Support. https://support.wix.com/en/article/enabling-https-for-your-wix-site
[106] Xueyang Xu, Z Morley Mao, and J Alex Halderman. 2011. Internet Censorship in China: Where Does the Filtering Occur?. In *12th Passive and Active Network Measurement Conference (PAM '11)*. Springer, New York, NY, USA, 133–142. https://doi.org/10.1007/978-3-642-19260-9_14
[107] Eric Zeng, Frank Li, Emily Stark, Adrienne Porter Felt, and Parisa Tabriz. 2019. Fixing HTTPS Misconfigurations at Scale: An Experiment with Security Notifications. In *2019 Workshop on the Economics of Information Security (WEIS '19)*. 19 pages. https://weis2019.econinfosec.org/wp-content/uploads/sites/6/2019/05/WEIS_2019_paper_16.pdf
[108] Liang Zhang, David Choffnes, Dave Levin, Tudor Dumitraş, Alan Mislove, Aaron Schulman, and Christo Wilson. 2014. Analysis of SSL Certificate Reissues and Revocations in the Wake of Heartbleed. In *14th ACM Internet Measurement Conference (IMC '14)*. ACM, New York, NY, USA, 489–502. https://doi.org/10.1145/2663716.2663758