An Opinionated Proposal to Improve Internet TLS Certificate Management:

Burn It All Down

Joe Thompson, Clarity Business Solutions

Intro; Let's Talk

Who's that at the podium?

Almost 30 years in IT (Kubernetes-enabled since 2015)

Past employers: HashiCorp, Mesosphere, CoreOS, Red Hat, among others

- Currently a Consulting Engineer for Clarity Business Solutions

Pronouns: he/him
Blood type: Caffeine-positive
Pop-culture references center of gravity: around 1989

How to get in touch:

- Kubernetes Slack: @kensey
- LinkedIn

Now about the most important people here -- you!

- Who's here at their first SCaLE?
- Who here has run an internal certificate authority?
- Who here has ever had to deal regularly with a public certificate authority?
- Who here enjoyed doing that?

What we're aiming to cover

- Where things started
- The mess we're currently in
- Where we go from here

Warning: this is a rant

- I intentionally take a very hard line here: the current situation is untenable
- I am not deliberately attempting to be unfair to any party but I'm also not especially concerning myself with nuance and subtlety here
- I touch only the surface level of most of this because the situation is fractally bad: every horrible piece of it look at has is own deeper levels of horror
- I do not authoritatively propose a *solution* to the problem but I do share some ideas about the general shape one might have

I was there, Gandalf, 30 years ago...

Let's get in the time machine: early 1995

- X-Files was in its second season
- I was finishing my first (and as it happened, final) full year at UVa
- And a bunch of Netscape engineers got together and said "Let's make the whole infrastructure of Internet security a tottering pile of Jenga bricks"

Not really, but that's basically what happened

Let's back up: what is Netscape trying to do in 1995?

Secure communications across untrusted channels

- Fortunately, public-key cryptography has an answer for this: each party has a keypair, one public, one private
- Private keys can sign data and decrypt data encrypted with the public key
- Public keys can encrypt data and verify signatures made with the private key
- This requires that parties exchange public keys to secure a communication channel in both directions, but public keys can safely be sent over unsecured channels

Key exchange does nothing to verify *identity*

- If a connection is subverted before security can be established, an attacker can send you its own key instead of the real second party's

We're really getting ready for e-commerce

- Identity verification is *critical* when money is moving around at the speed of light

Enter the "certificate authority" (CA)

- A third party that creates a certificate of identity that also includes the public key tied to that identity, and signs that certificate with its own key, published in its own self-signed certificate of identity
- The certificate is based on the pre-existing X.509 standard (part of the hundreds of pages of the suite of standards for the X.500 directory system)

But how do I trust the CA's identity?

- Browsers (and operating systems) have preloaded certificates from certificate authorities deemed trustworthy

Deemed by who? Based on what criteria?

So how does this work in practice?

- Your browser requests a secure connection to a site
- The site sends back a certificate chain: its own identity certificate, plus all the intermediate certificates that signed it
- Your browser verifies the validity of that chain, then verifies that it anchors to a previously-trusted root
- The key in the certificate is then known to be the correct key for the known party and is used to establish the secured connection

So that works, right?

- Of course it does! Because CAs will never ever have any kind of issue that makes them or the certificates they issue untrustworthy
- Look, I don't know what to tell you... it was the '90s, we didn't even have SSH yet

Where that ended up

So what's the problem?

- There is no inherent limitation on the scope of certificates a CA can issue
- CAs make mistakes
- CAs take shortcuts to get around issues with data availability to verify identity
- CAs sometimes just aren't very good at being CAs
- CAs are fooled by bad actors and issue fraudulent certificates
- CAs are fooled by bad actors (or just decide to take the money) and issue those bad actors their own CA certificates
- CA system vulnerabilities can be exploited to get fraudulent certificates, or certificates that exploit client implementation vulnerabilities, issued

Certificates must occasionally be revoked

- All the ways of doing that are terrible
- CAs can publish a list of revoked certificates called a CRL
  - CRLs can get really big
- OCSP allows a browser to query for the validity of one single certificate
  - Potential privacy compromise, requires the responder to be extremely highly available
- OCSP Stapling sends a pre-retrieved OCSP response along with the certificate so a query is not necessary
  - Some clients query anyway

It's not always an innocent mistake

- CAs are also vulnerable to insider threats or might be (or become) outright rogue actors, so there have to be transparency measures
- Certificate Transparency involves a CA writing every certificate it issues to a transparency log
- Domain owners then can check the log for malicious certificates
- What about malicious omission of a malicious certificate from a transparency log?
  - It's up to clients to refuse to accept certificates not documented in a log, but there are some situations where a new certificate is valid and should be accepted despite not being in any log yet
  - To prevent a malicious issuer from colluding with a malicious log to exploit these scenarios, browsers typically require verification from multiple logs and only query certain logs deemed trustworthy

Can you control which CAs issue certs for you?

- Of course not!
- At best, you can publish records in your DNS that follow a scheme called CAA to announce which certificate authorities you authorize to issue your certs
- CAA is self-enforced by CAs

There are other threats that only exist because of this ultimate-trust model

- In early 2015 it was revealed that Lenovo laptops shipped with software preinstalled that used its own CA certificate added to the trust store, and an embedded private key, to subvert the security of TLS connections by essentially locally issuing valid certificates for arbitrary domains
- Then right after that had been thoroughly discussed in the tech press, Dell did the same thing

OK, so it's a mess, but it mostly works, right? There's a small set of trusted CAs, we basically know who they are, we basically know they can be trusted, it's *pretty much* fine, right?

You wish.

Poll: How many public CAs does your system trust?

- It's a trick! The set of certs isn't consistent between the various browser and OS trust stores, and the specific certs contained are constantly changing as old CA certs expire, new CAs are added, and (far too regularly) existing CAs are detrusted for cause.
- In general though it's something in the neighborhood of 100-250
- How do they get there?

Remember "deemed by who?"

- The CA/Browser Forum, that's who
- Who the heck are they?

It's a big club... and you ain't in it.

- George Carlin (1937-2008)

OK, so if it's this big a problem, why isn't anybody else talking about it?

Well, they are:

- Moxie Marlinspike laid out the issues in a BlackHat presentation in 2011 and coauthored an IETF Draft for a protocol known as TACK in 2012
- DANE (DNS-based Authentication of Named Entities) was published as RFC 6394 in 2011
- An implementation of DANE called DANE TLSA was published as RFC 6698 in 2012
- Neither of these really went anywhere, mostly because browser makers didn't implement them

Where can we go and how can we get there?

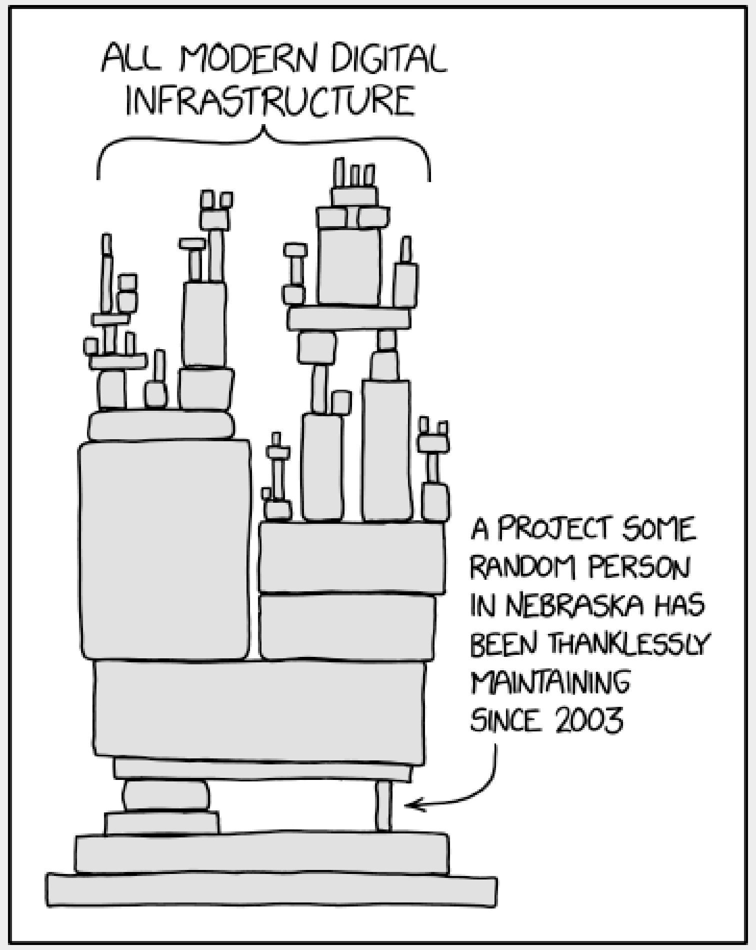
OK, so it really is that bad

- What do we do?

BURN IT DOWN

It's all hacks built on hacks built on other hacks.

I truly believe it's beyond saving in its current form.


ALL MODERN DIGITAL INFRASTRUCTURE
A PROJECT SOME RANDOM PERSON IN NEBRASKA HAS BEEN THANKLESSLY MAINTAINING SINCE 2003

This, but also on top of 100-250 other tiny skinny vertically-stacked blocks named "A certificate authority of unknown trustworthiness" (image credit: xkcd.com)

What do we replace it with, though?

Good question. I hope you can answer it ("no authoritative solutions", remember?) But in general I think this is the box solutions need to fit in:

- We have to give up the idea that any CA that a small cabal of mostly commercial interests chooses, is ultimately trusted to issue any certificate for any domain on the entire Internet with only their own good behavior and the unceasing vigilance of every single owner of every domain as enforcement
- We have to have some mechanism for detrusting incompetent/rogue CAs immediately (+/-), not with a delay of "everybody in the world finally downloads the latest OS/browser updates"
- We may have to give up some or all of the entire idea of in-band identity verification based on pre-trusted authorities

Are people really going to do this?

- Ideally, yes. In that scenario very serious conversations, like the ones that took place around the Spectre/Meltdown CPU vulnerabilities, start happening sooner rather than later. In my opinion this is at least that serious.
- Realistically, I expect everybody to do what people have been doing already: pretend everything is basically OK and we just need this one more little fix to plug the last hole (till the next last hole comes along), until something truly catastrophic happens

Wrapping up

Final thoughts

- **Don't Panic** but DO get concerned
- Get informed!
- Get loud!

Thank you!

Slides:

https://tinyurl.com/burn-tls-down