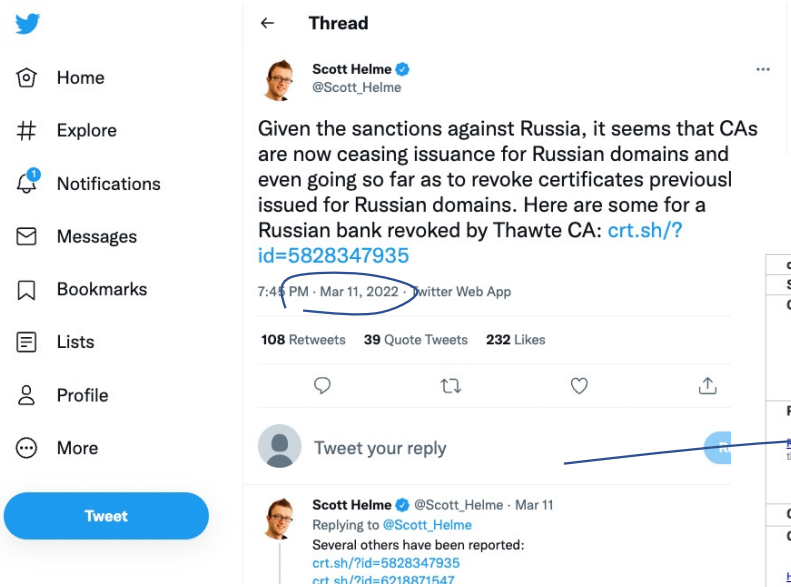
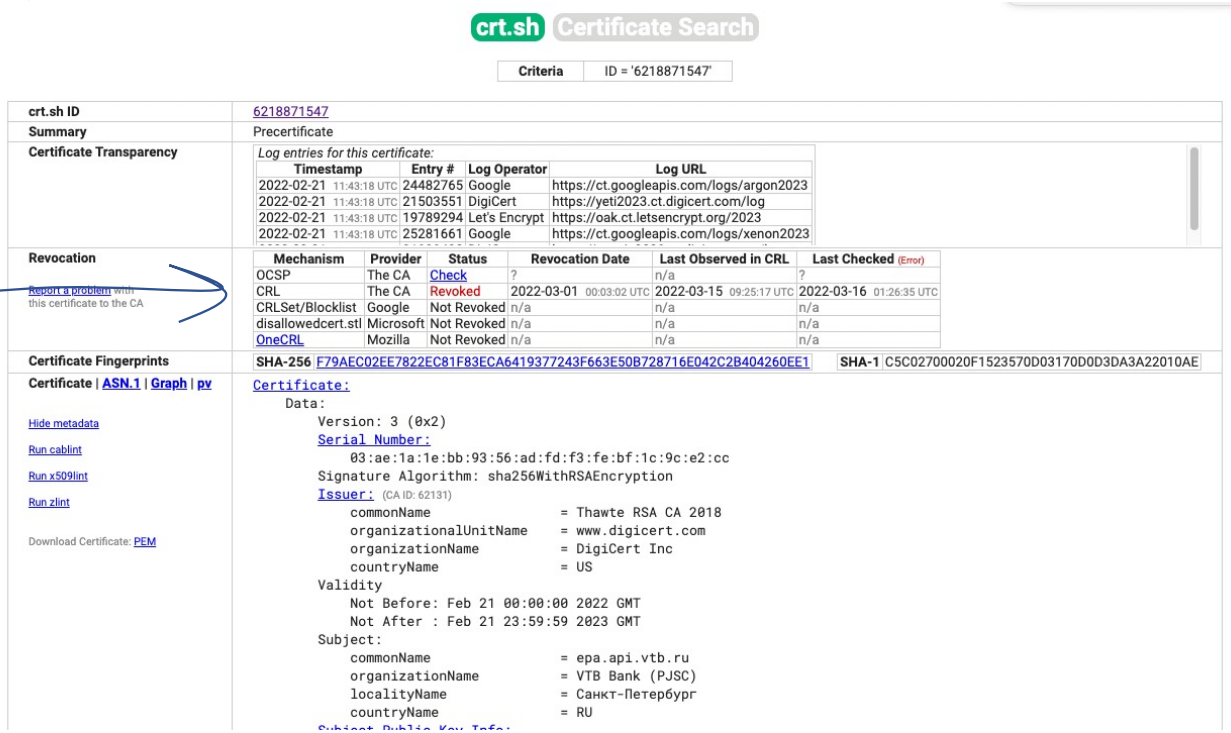
Certificate Revocation

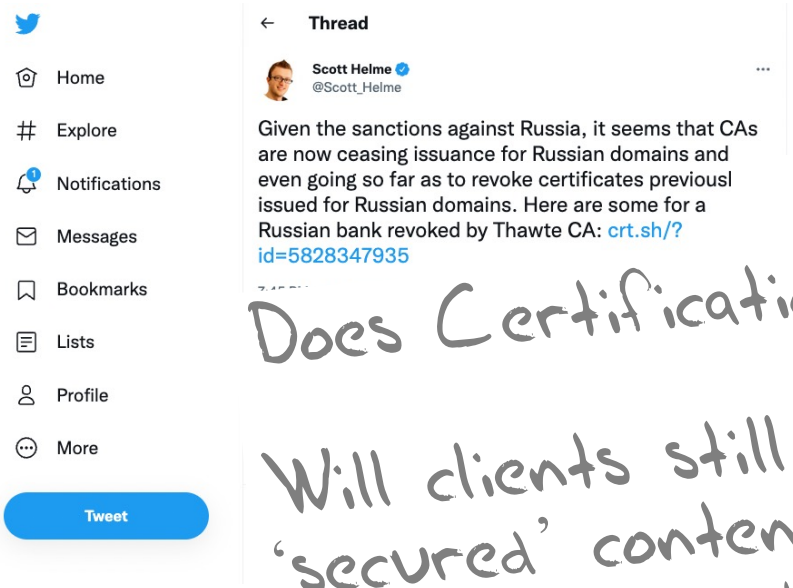
Geoff Huston AM
APNIC Labs

Certificate Revocation as a "sanction"



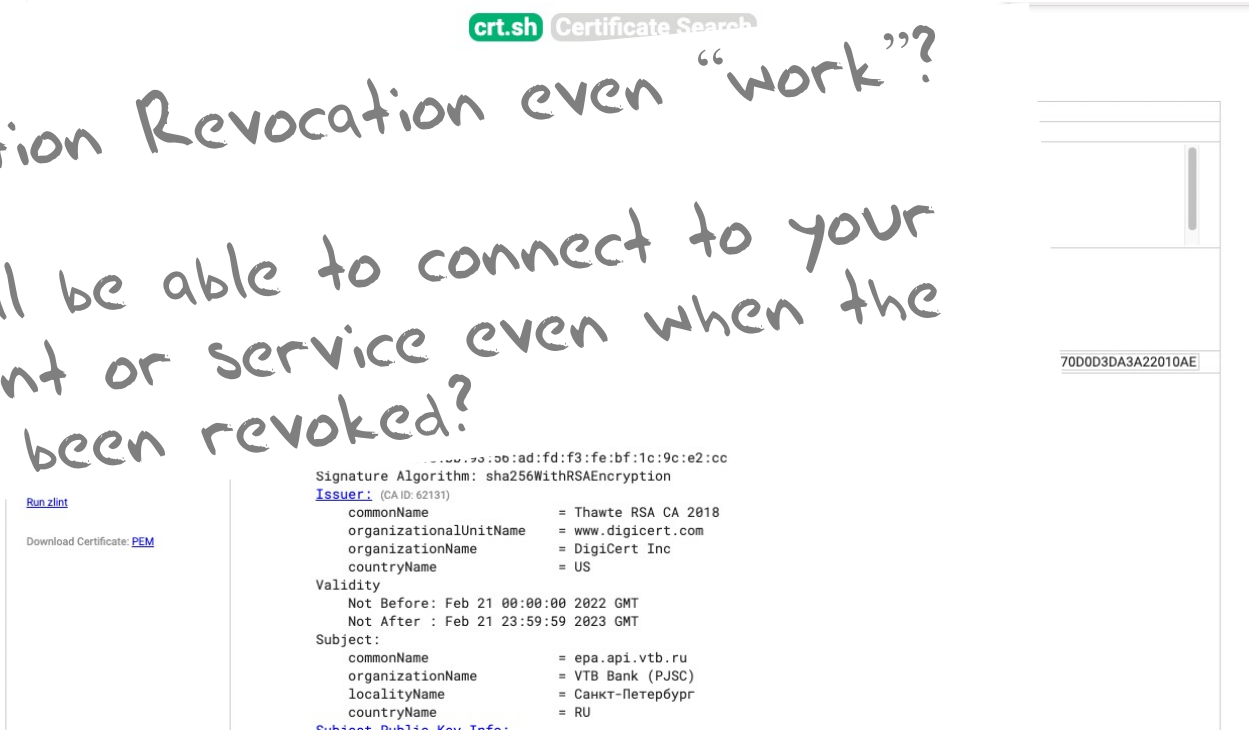


Certificate Revocation as a "sanction"

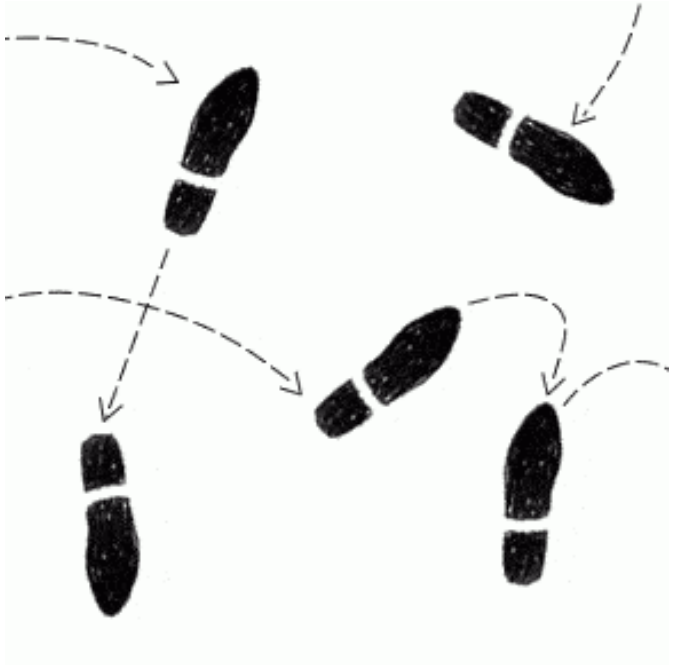


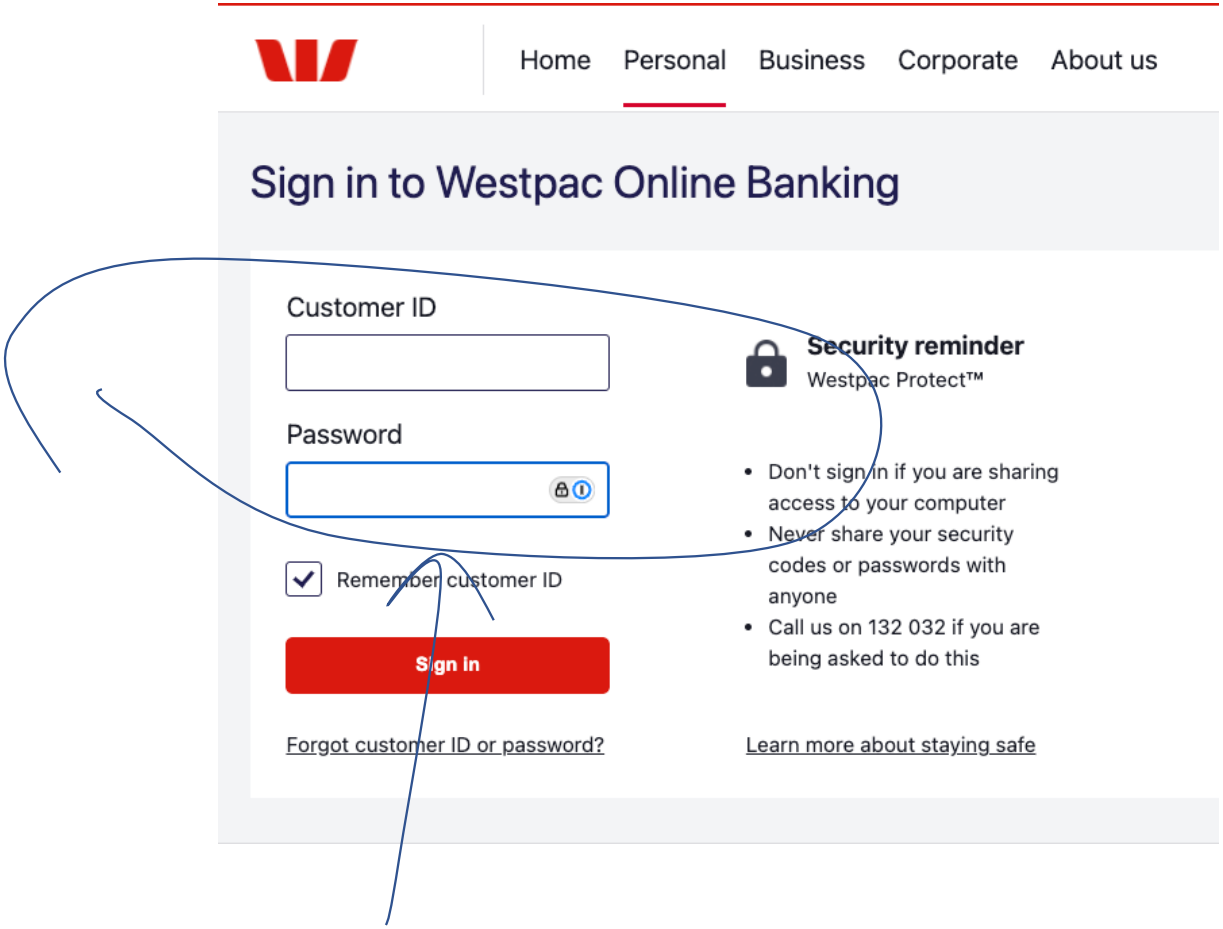
Does Certification Revocation even "work"?

Will clients still be able to connect to your 'secured' content or service even when the certificate has been revoked?



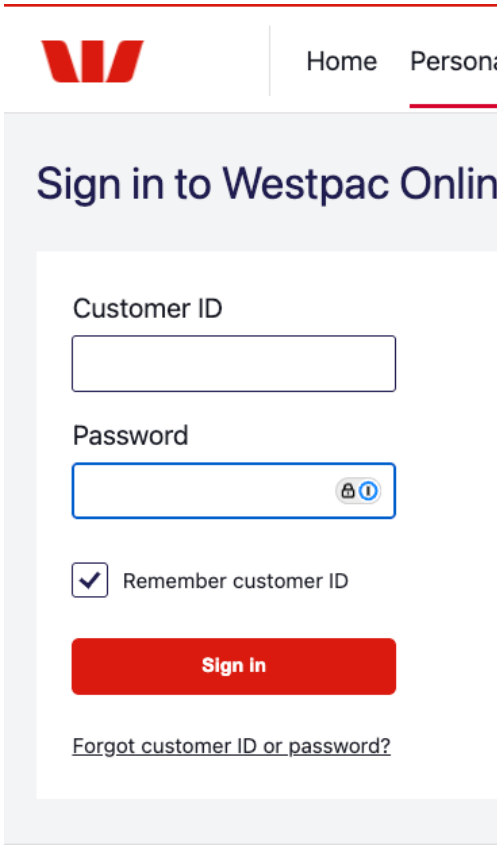
Let's take a step back…





Why should i trust this?

How can i be assured that this is my bank and not a clever scam?



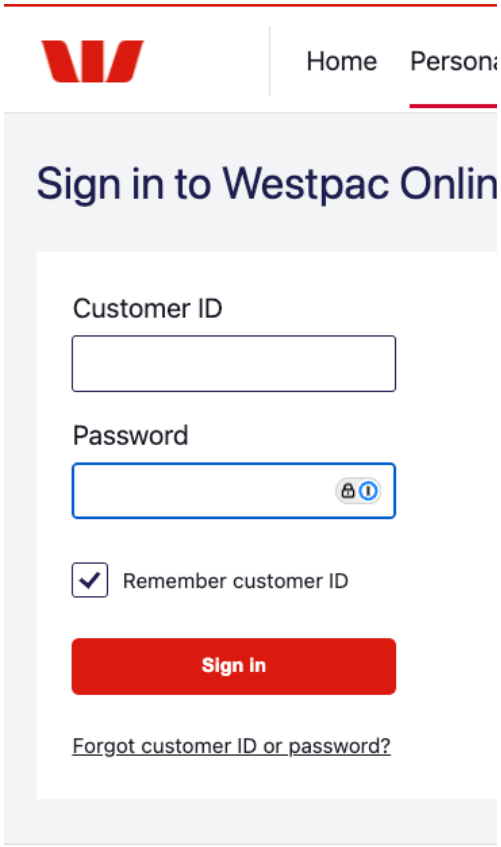
$ **dig www.westpac.com.au**

```
; <<>> DiG 9.20.24 <<>> A www.westpac.com.au @localhost
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22032
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 5faac10685f2d0b1010000006a738c4fe127d05c87c511e7 (good)
;; QUESTION SECTION:
;www.westpac.com.au.            IN      A

;; ANSWER SECTION:
www.westpac.com.au.     10784   IN      CNAME   dplenhq18279.cloudfront.net.
dplenhq18279.cloudfront.net. 54 IN      A       18.67.110.111
dplenhq18279.cloudfront.net. 54 IN      A       18.67.110.13
dplenhq18279.cloudfront.net. 54 IN      A       18.67.110.84
dplenhq18279.cloudfront.net. 54 IN      A       18.67.110.121

;; Query time: 0 msec
;; SERVER: ::1#53(localhost) (UDP)
;; WHEN: Thu Aug 06 05:17:35 AEST 2026
;; MSG SIZE  rcvd: 180
```



$ **dig +short TXT 18.67.110.111.ipasn.net**
"18.67.110.111|IPv4|ADVERTISED|18.67.104.0/21|16509|Amazon_Technologies_Inc.|US|
United_States_of_America|arin|18.64.0.0/10|assigned|2019-10-07|VLD|18.64.0.0/14|24|16509|ARIN"

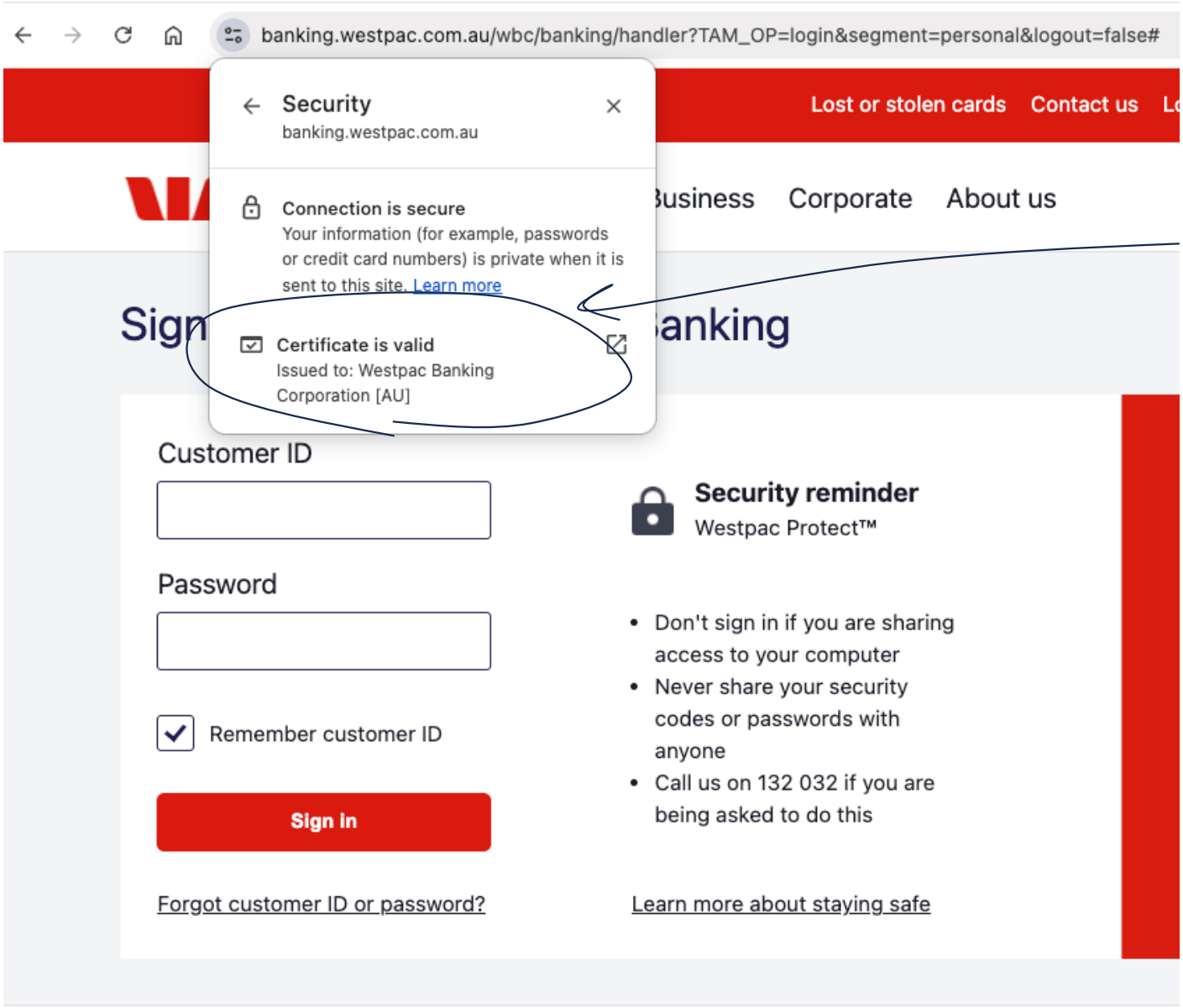
So where and who is my bank?

The server that is displaying the login page comes from an IP address operated by Amazon!

Cloudfront..net is also a domain name operated by Amazon

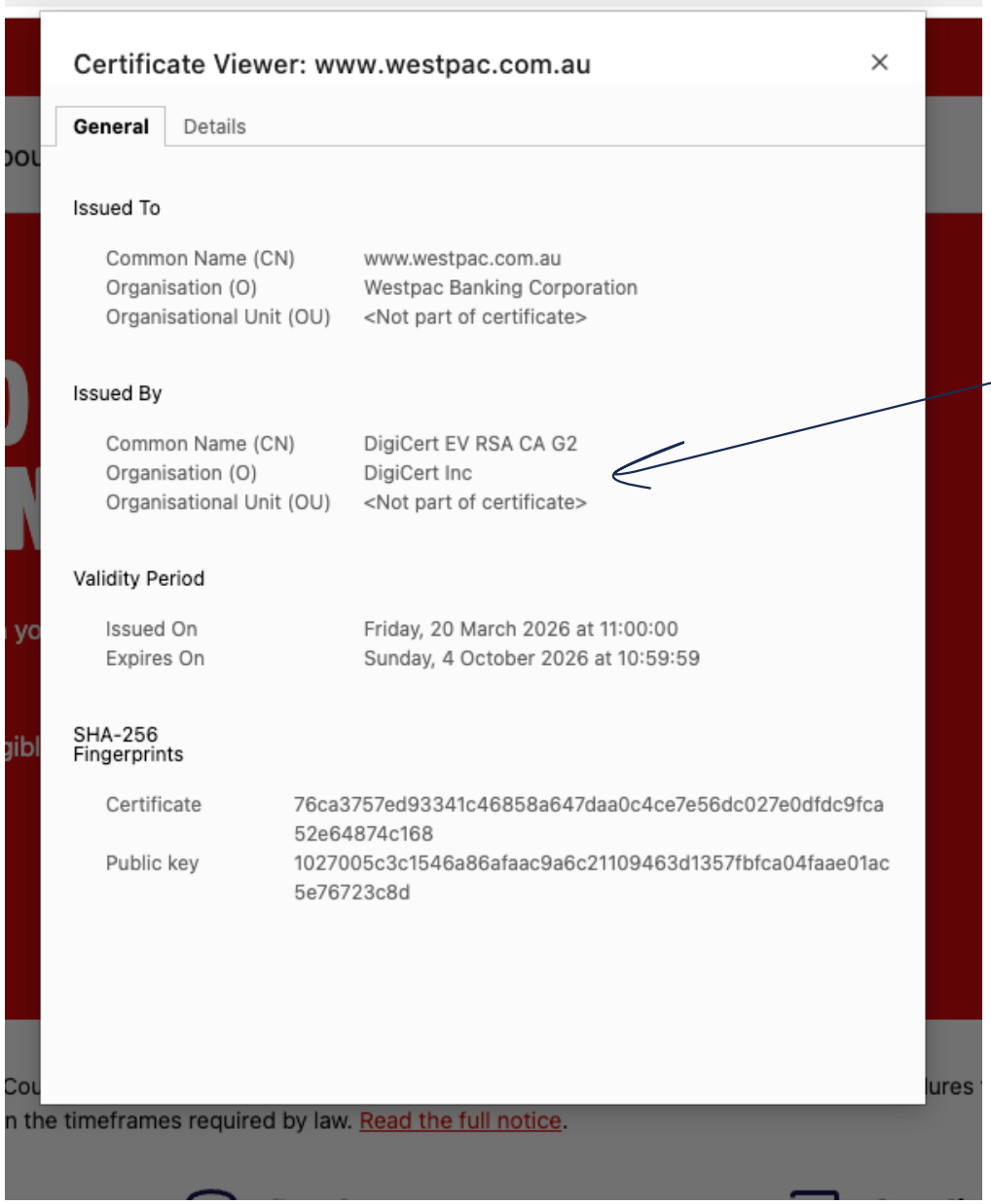
Am i entering the Bank of Amazon?

Let's ask the browser what the TLS connection has found



This sounds reassuring, but why the hell should I trust this?

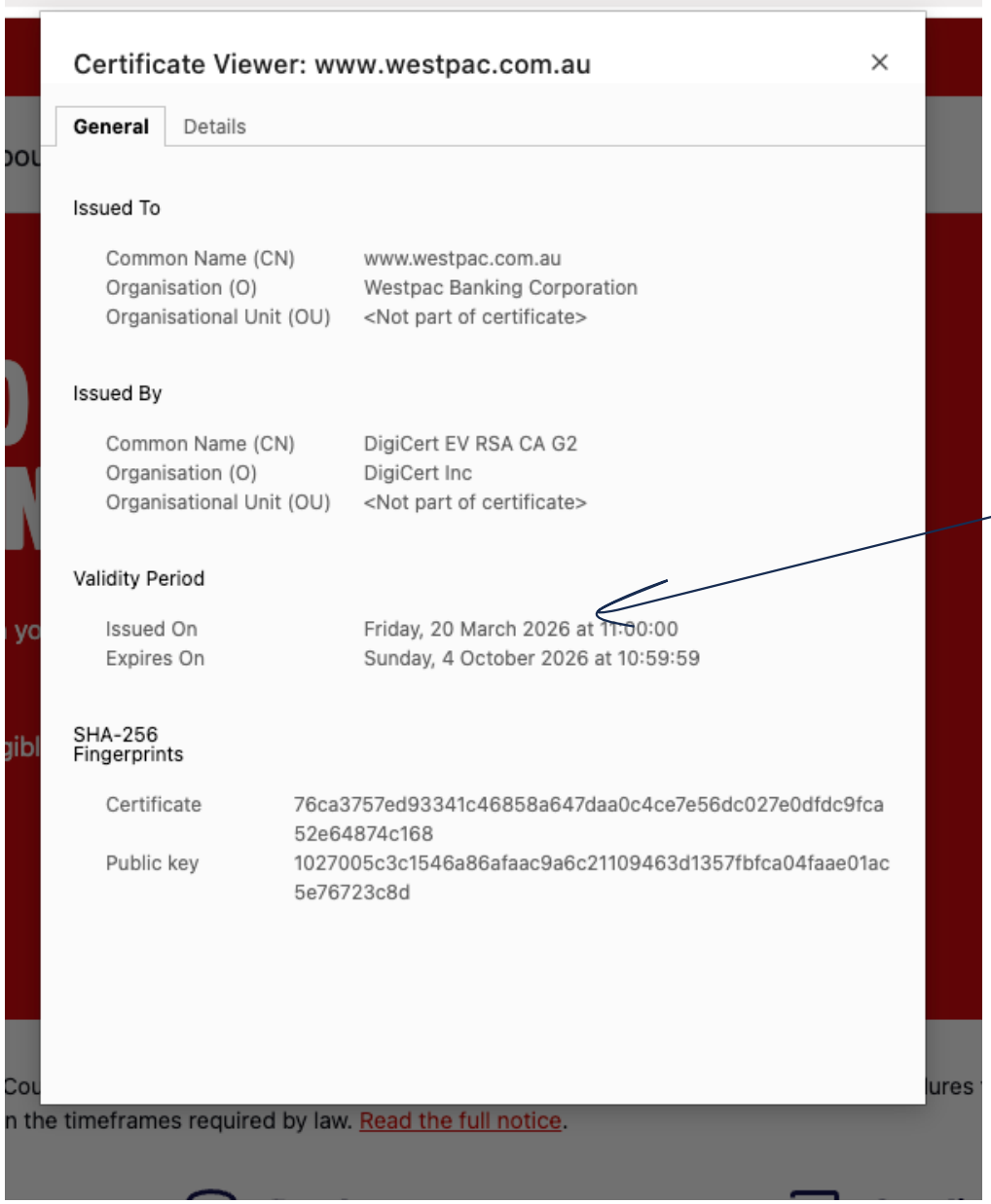
What does this certificate say?



Why the hell should i trust "DigiCert inc"?

Why does my browser trust Digicert?

i have never met these folk and i have no idea what this means!



This certificate is 6 months old, and I am being asked to trust it for the next month!

What if their private key is leaked in the next month? What if the CA is breached? What if ...

What if something happens in the next month that says that I really should not trust this certificate any more?

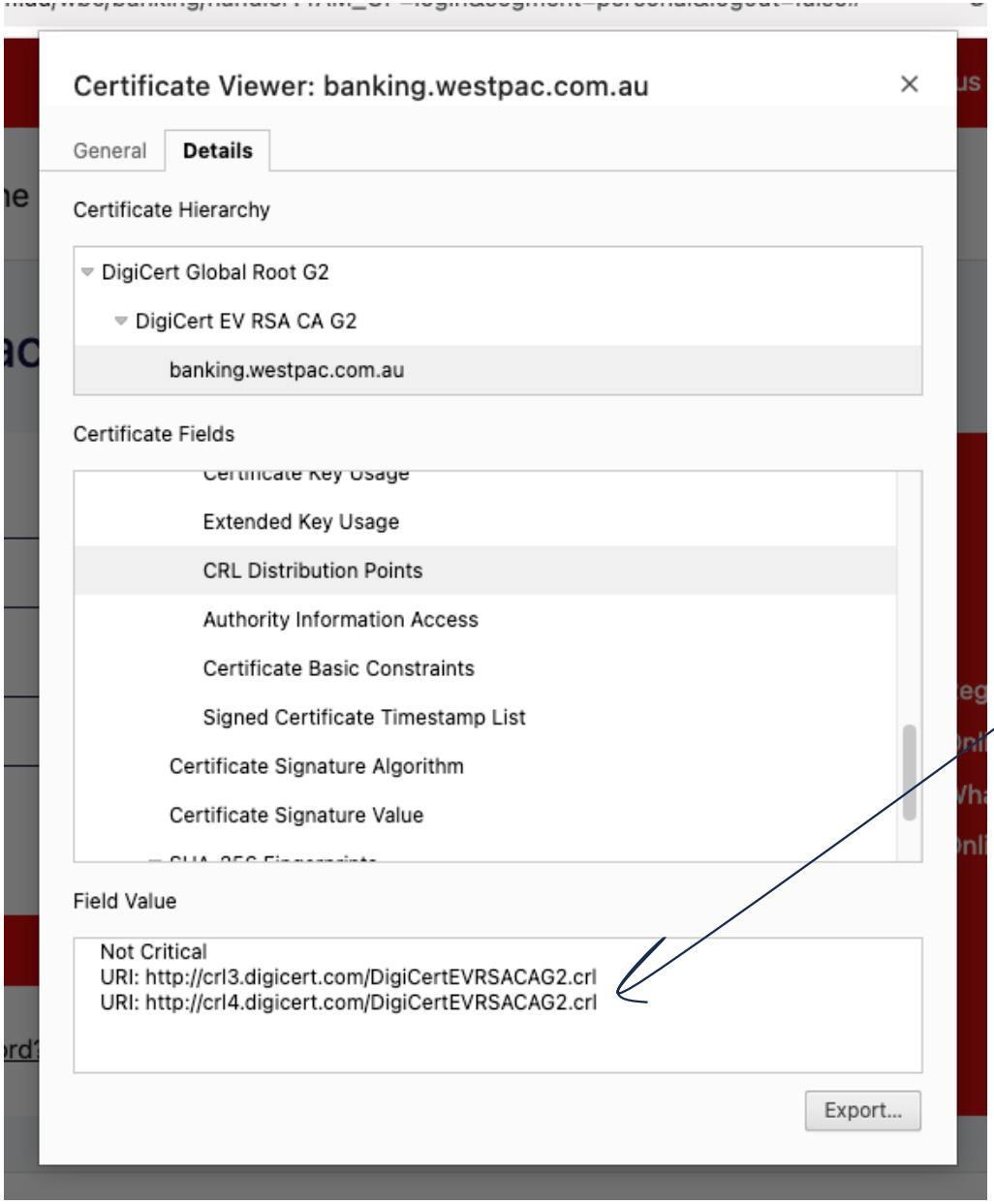
The Answer is Revocation!

- Each CA maintains a “Certificate Revocation List”
- This is a list of the serial numbers of all current certificates issued by this CA that should no longer be trusted
- If a Bad Thing happens, or for any other reason, and the CA believes that the certificate should not be trusted, then the certificate’s serial number is added to this CA’s Certificate Revocation List
- Anyone who is worried about the “currency” of a certificate should check to see if its serial number is listed in the CA’s current CRL

The Answer is Revocation!

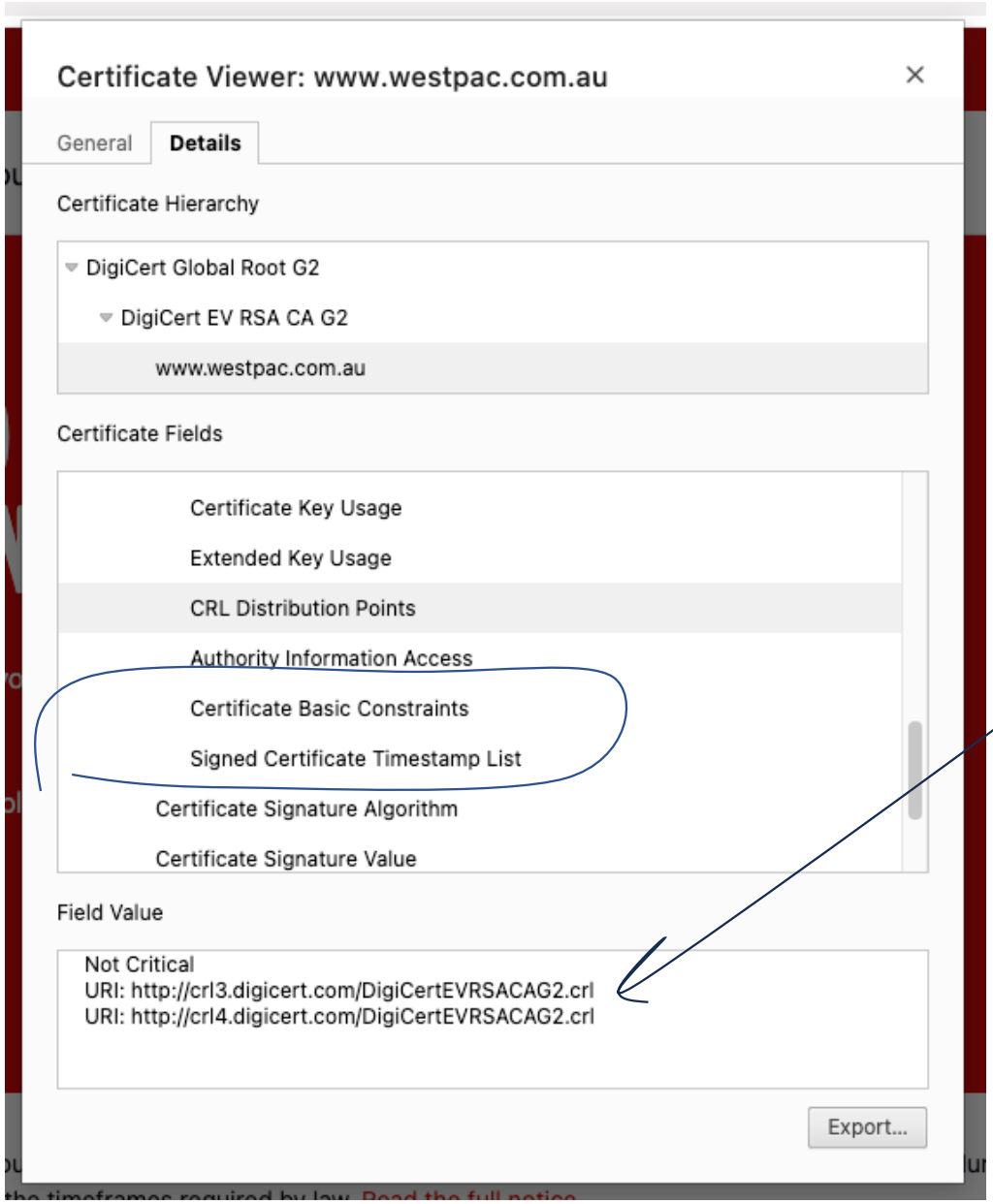
- Each CA maintains a "Certificate Revo[illegible]"
- This is a list of the serial numb[illegible]rtificates issued by this CA that should no l[illegible]
- If a Bad Thing h[illegible]ther reason, and the CA believes that the c[illegible]e trusted, then the certificate's serial number [illegible]s CA's Certificate Revocation List
- Anyone wh[illegible]worried about the "currency" of a certificate should check to see if its serial number is listed in the CA's current CRL

But how can a client perform this check?



The URL for the CA's CRL is listed in the certificate, if the CA publishes a CRL

Here's the one for DigiCert inc



The URL for the CA's CRL is listed in the certificate, if the CA publishes a CRL

Here's the one for DigiCert inc

it's not an HTTPS URL, but it's a signed object requiring authentication, so tampering is challenging whether or not the retrieval transport is authenticated and encrypted

What's in that CRL?

```
$ wget http://crl3.digicert.com/DigiCertEVRSACAG2.crl
$ openssl crl -inform DER -text -noout -in DigiCertEVRSACAG2.crl
Certificate Revocation List (CRL):
    Version 2 (0x1)
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: C=US, O=DigiCert Inc, CN=DigiCert EV RSA CA G2
    Last Update: Aug  5 03:01:51 2026 GMT
    Next Update: Aug 12 03:01:51 2026 GMT
    CRL extensions:
        X509v3 Authority Key Identifier:
            6A:4E:50:BF:98:68:9D:5B:7B:20:75:D4:59:01:79:48:66:92:32:06
        X509v3 CRL Number:
            2223
        X509v3 Issuing Distribution Point: critical
            Full Name:
              URI:http://crl3.digicert.com/DigiCertEVRSACAG2.crl
              URI:http://crl4.digicert.com/DigiCertEVRSACAG2.crl

Revoked Certificates:
    Serial Number: 0CB808AFD03A255E4066524E7659E1FB
        Revocation Date: Jul  1 15:55:02 2025 GMT
    Serial Number: 0EB379F68F503BFEB540D0CACBDA43BB
        Revocation Date: Jul  3 09:53:30 2025 GMT
    Serial Number: 0779AD2C3EB8DFC83FAC4EA6A82C9703
        Revocation Date: Jul  4 06:37:02 2025 GMT
    Serial Number: 035B803B60CE8C1389FE408B1A78635C
        Revocation Date: Jul  5 18:33:03 2025 GMT
        CRL entry extensions:
            X509v3 CRL Reason Code:
                Superseded
    [repeated 17,523 more times]
```

This CRL is only updated every 7 days?

17,527 certificates are revoked in this CRL

How do you check a certificate using a CRL?

1. Retrieve the CRL
2. Validate the digital signature of the CRL against the CRL contents
3. Validate that the digital signature was generated by the CA's private key, and create a validation chain to a Trust Anchor
4. Validate the currency of the CRL with Update Date of the CRL
5. Look for the Certificate's Serial Number in the CRL

If you find the Serial Number listed in the CRL then the certificate is a dud!

Do NOT trust it!

Does anyone actually do this?

Does anyone actually do this?

No!

Does anyone actually do this?

Well, no, not this way!

it takes too long to load all the CRLs and perform the CRL checking actions and nobody is willing to pay this time penalty

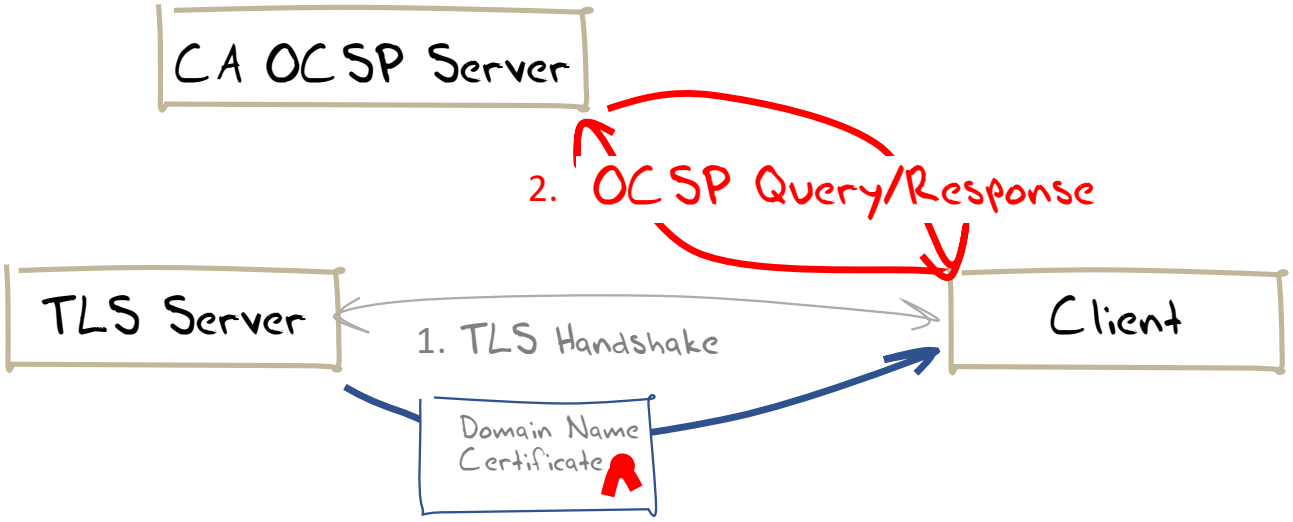
it's also a totally inefficient design – why retrieve the entire list of revoked certificates in a CRL when all you want to know is the status of a single certificate?

This may have worked for circulating printed lists of revoked credit card numbers in a bygone age, but its completely useless today (though even then nobody checked!)

Plan B - OCSP

Online Certificate Status Protocol – RFC 2560

- Allows a client to query the CA to query the revocation status of an individual certificate, without downloading the entire collection of revoked certificates from the CA



OCSP Scorecard

✗ It's still a round time time penalty of additional delay

✗ It tells the OCSP server what each client is doing => **a significant privacy leak**

✗ It imposes critical load on the OCSP server

- Because of an arrangement with Let's Encrypt, Akamai was responding to 2/3 of all OCSP responses, creating a choke point and a single point of failure

✗ What should the client do if the OCSP server is uncontactable?

- Fail? – DDOS vector
- Allow? – Vulnerability vector!

Some platforms and browsers support OCSP checking

Some don't

OCSP Scorecard

✗ It's still a round time time penalty of additional delay

✗ It tells the OCSP server what each client is doing => **privacy leak**

✗ It imposes critical load on the OCSP

- Because of [illegible] was responding to 2/3 of all OCSP
- [illegible] single point of failure

✗ What should the client do if the OCSP server is uncontactable?

- Fail? – DDOS vector
- Allow? – Vulnerability vector!

This seems like a stupidly inconsistent and haphazard basis for the web's only security framework!

Some platforms and browsers support OCSP checking

Some don't

What about OCSP Stapling?

Rather than pushing the responsibility for revocation checking entirely onto the client with CRLs, or turning it into a query/response interrogation of the CA's server with OCSP, can the content server furnish the 'current' OCSP response as a stapled attribute of the TLS credentials? After all a signed response is valid no matter how you learned of it, if the crypto chain can be validated

This can be made mandatory through the use of a certificate extension "OCSP Must Staple" - this tells browsers that, if they see that extension in a certificate, they should never contact the CA about it and should instead expect to see a stapled copy of the signed OCSP response in the TLS handshake. Failing that, browsers should refuse to connect.

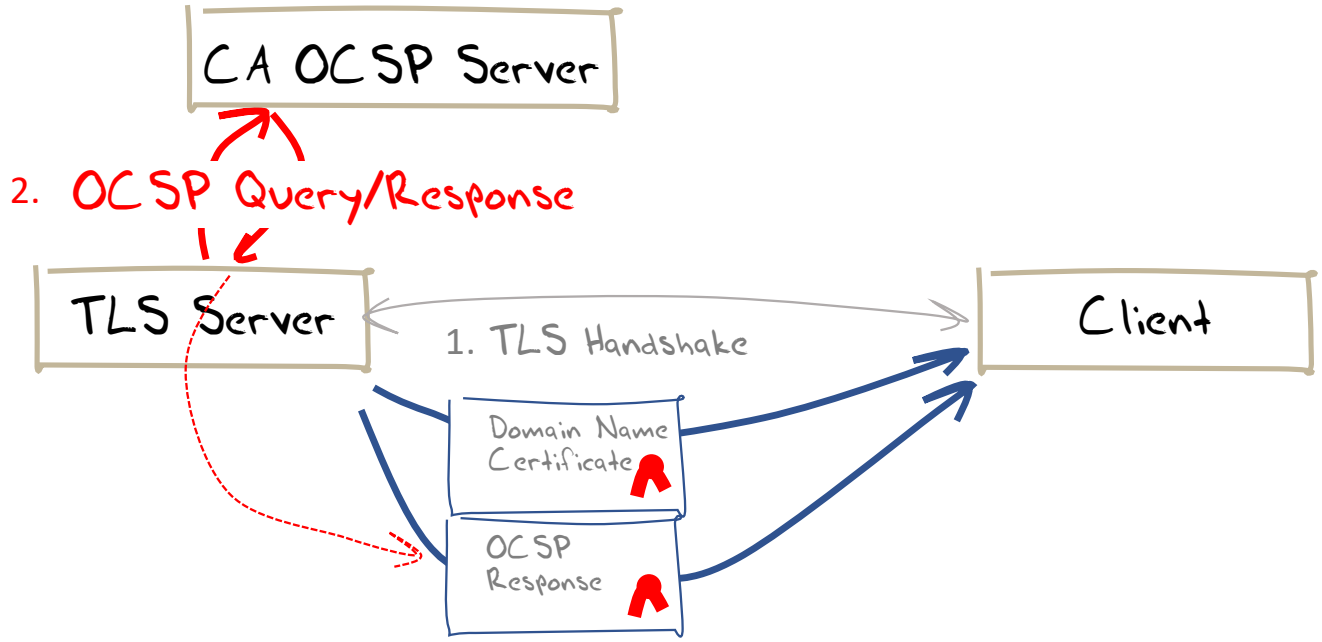
Plan C - OCSP Stapling

“If we want a scalable solution to the revocation problem then it's probably going to come in the form of short-lived certificates or something like OCSP Must Staple. Recall that the original problem stems from the fact that certificates are valid for years. If they were only valid for days, then revocation would take care of itself. ”

Adam Langley,

https://www.imperialviolet.org/2014/04/19/revchecking.html

Stapled OCSP



OCSP Stapling Scorecard?

Rather than pushing the responsibility for revocation checking entirely onto the client with CRLs, or turning it into a query/response interrogation of the CA's server with OCSP, can the content server furnish the 'current' OCSP response as a stapled attribute of the TLS credentials?

Yes – OCSP Stapling (RFC 6066) and TLS Must Staple (RFC 7633)

- ✓There is no additional network latency in TLS start as the OCSP data is in-band
- ✓The CA does not attain knowledge of client-initiated sessions
- ✓The CA does not have to operate a high-capacity server infrastructure to respond to client OCSP queries
- ✓The client can fail 'hard' on missing or unvalidatable OCSP data

What's the point of OCSP Stapling?

- The default outcome of an evaluation of a current certificate is to accept it, as long as the client can construct a validation chain to a local trust anchor
- CRLs and OCSP is meant to alert you to a changed circumstance that indicates that the certificate should not be trusted
- So, the only OCSP item that is useful is one that indicates that the certificate has been revoked
- But why should a server send me the certificate and a stapled OCSP response if the signed OCSP response says that the certificate has been revoked?
  - ***Shouldn't the server use this OCSP information itself and fail the TLS handshake if the certificate that it was going to proffer is revoked?***

OCSP Stapling

- What value is OCSP and OCSP Stapling adding to certificates and their use?
  - “nothing” appears to be the correct answer here!

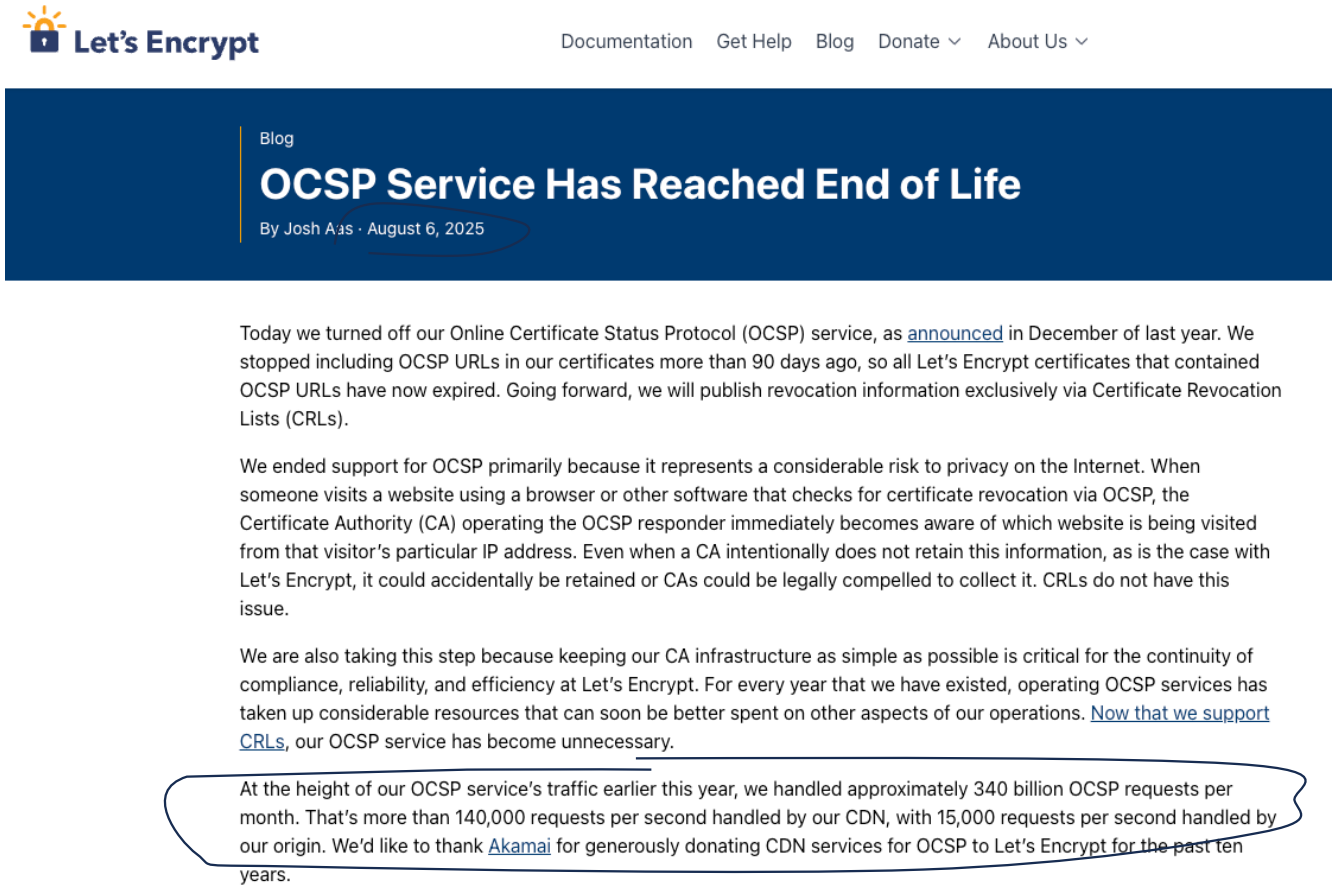
OCSP = Fail?

“That's why I claim that online revocation checking is useless - because it doesn't stop attacks. Turning it on does nothing but slow things down. You can tell when something is security theatre because you need some absurdly specific situation in order for it to be useful.”

Adam Langley,

https://www.imperialviolet.org/2014/04/19/revchecking.html

Ask not for whom the bell tolls; it tolls for OCSP!



Let's Encrypt

Documentation Get Help Blog Donate About Us

Blog

OCSP Service Has Reached End of Life

By Josh Aas · August 6, 2025

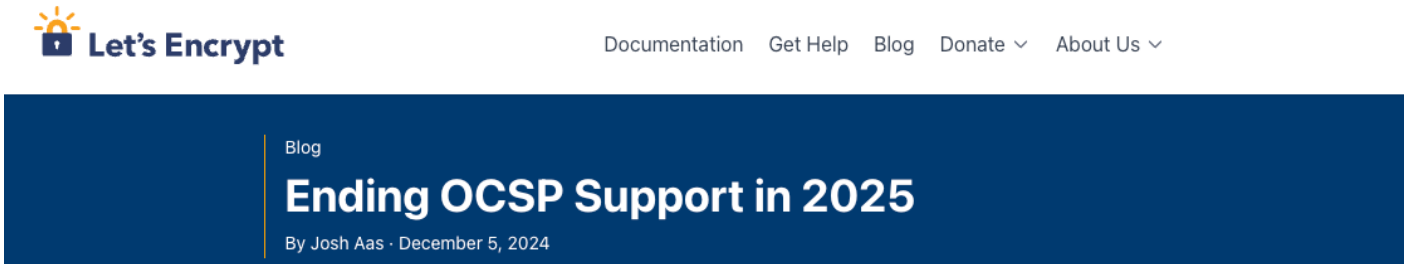
Today we turned off our Online Certificate Status Protocol (OCSP) service, as announced in December of last year. We stopped including OCSP URLs in our certificates more than 90 days ago, so all Let's Encrypt certificates that contained OCSP URLs have now expired. Going forward, we will publish revocation information exclusively via Certificate Revocation Lists (CRLs).

We ended support for OCSP primarily because it represents a considerable risk to privacy on the Internet. When someone visits a website using a browser or other software that checks for certificate revocation via OCSP, the Certificate Authority (CA) operating the OCSP responder immediately becomes aware of which website is being visited from that visitor's particular IP address. Even when a CA intentionally does not retain this information, as is the case with Let's Encrypt, it could accidentally be retained or CAs could be legally compelled to collect it. CRLs do not have this issue.

We are also taking this step because keeping our CA infrastructure as simple as possible is critical for the continuity of compliance, reliability, and efficiency at Let's Encrypt. For every year that we have existed, operating OCSP services has taken up considerable resources that can soon be better spent on other aspects of our operations. Now that we support CRLs, our OCSP service has become unnecessary.

At the height of our OCSP service's traffic earlier this year, we handled approximately 340 billion OCSP requests per month. That's more than 140,000 requests per second handled by our CDN, with 15,000 requests per second handled by our origin. We'd like to thank Akamai for generously donating CDN services for OCSP to Let's Encrypt for the past ten years.

Ask not for whom the bell tolls; it tolls for OCSP



Must Staple

Because of the privacy issues with OCSP, browsers and servers implement a feature called "OCSP Stapling", where the web server sends a copy of the appropriate OCSP response during the TLS handshake, and the browser skips making a request to the CA, thus better preserving privacy.

In addition to OCSP Stapling (a TLS feature negotiated at handshake time), there's an extension that can be added to certificates at issuance time, colloquially called "OCSP Must Staple." This tells browsers that, if they see that extension in a certificate, they should never contact the CA about it and should instead expect to see a stapled copy in the handshake. Failing that, browsers should refuse to connect. This was designed to solve some security problems with revocation.

Let's Encrypt has supported OCSP Must Staple for a long time, because of the potential to improve both privacy and security. However, Must Staple has failed to get wide browser support after many years. And popular web servers still implement OCSP Stapling in ways that create serious risks of downtime.

As part of removing OCSP, we'll also be removing support for OCSP Must Staple. CRLs have wide browser support and can provide privacy benefits to all sites, without requiring special web server configuration. Thanks to all our subscribers who have helped with the OCSP Must Staple experiment.

Its back to OCSP and CRLs (again!)

Chrome:

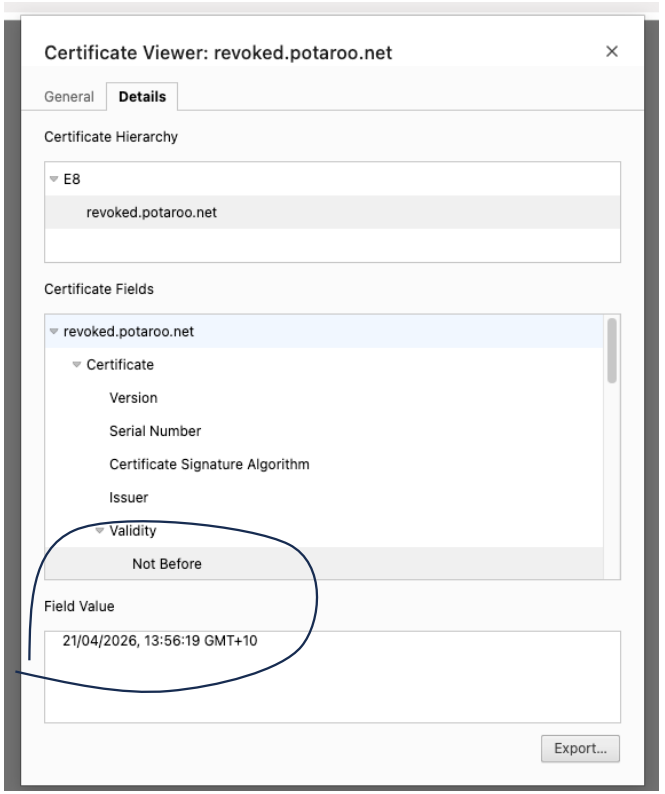
- Disabled OCSP in 2012 (Version 19)
- Did NOT support Must-Staple OCSP
- Relies on a feed of “CRLsets”

Safari:

- Uses downloaded form of “CRLsets”
- Will use Server-Provided stapled OCSP

How is this working?

1. Let’s generate a LetsEncrypt certificate:



How is this working?

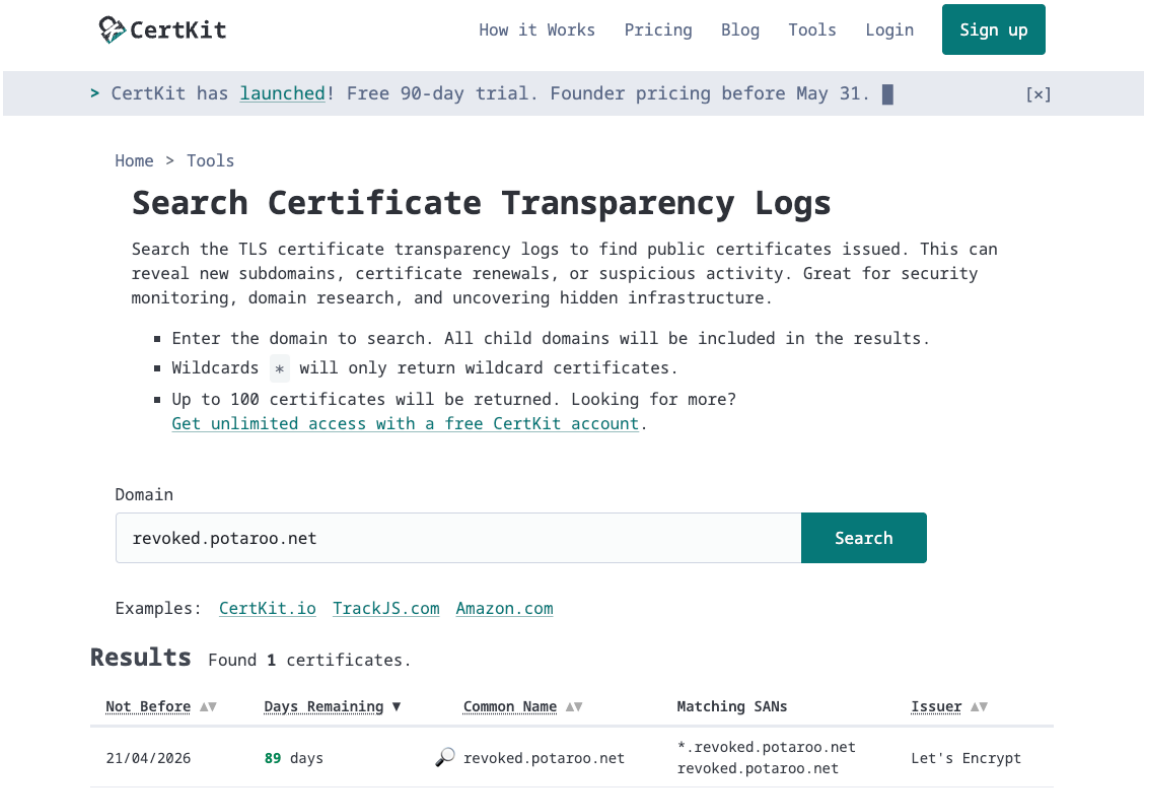
1. Let's generate a LetsEncrypt certificate
2. And then revoke it 4 hours later

```
$ curl http://e8.c.lencr.org/111.crl -o 111.crl
$ openssl crl -inform DER -text -noout -in 111.crl | grep -A 1 "Serial Number.* 0505C314D691"
    Serial Number: 0505C314D691F6819D355B9D8FF9885042EE
        Revocation Date: Apr 21 17:28:20 2026 GMT+10
```

And yes, this certificate is now listed on the Lets Encrypt CRL as revokes

Is Revocation part of Certificate Transparency?

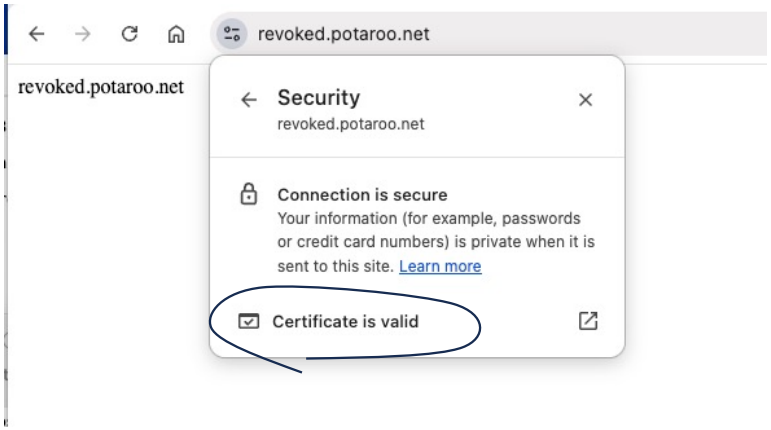
Evidently not!



How is this working?

1. Let’s generate a LetsEncrypt certificate
2. And then revoke it 4 hours later
3. Wait for a couple of weeks and see if my browser has learned of this revocation

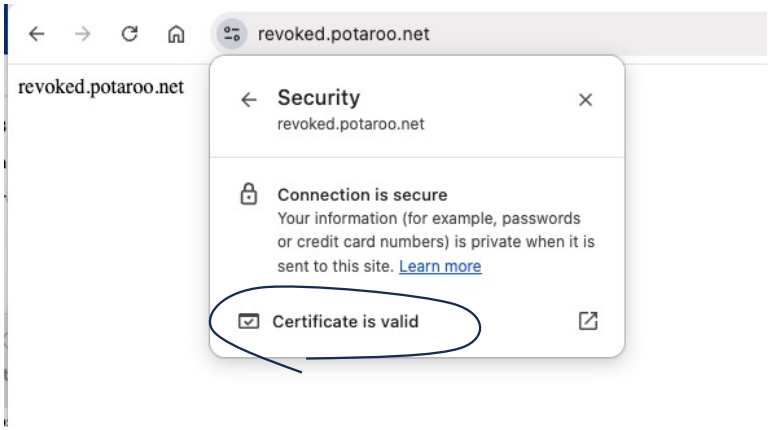
Chrome



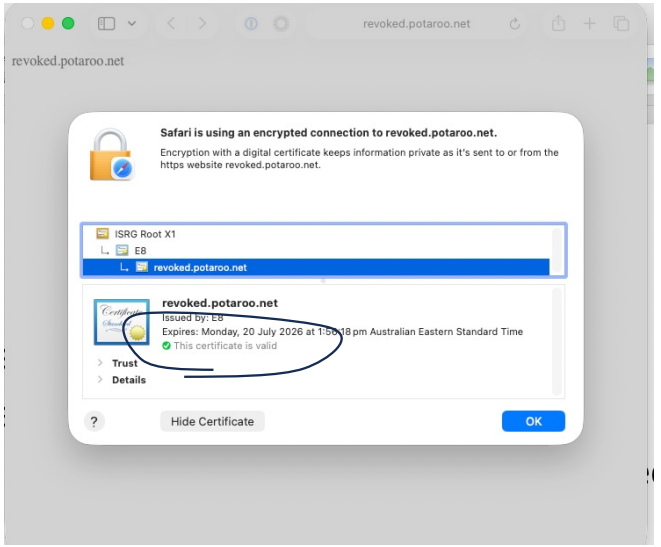
How is this working?

1. Let's generate a LetsEncrypt certificate
2. And then revoke it 4 hours later
3. Wait for a couple of weeks and see if my browser has learned of this revocation

Chrome



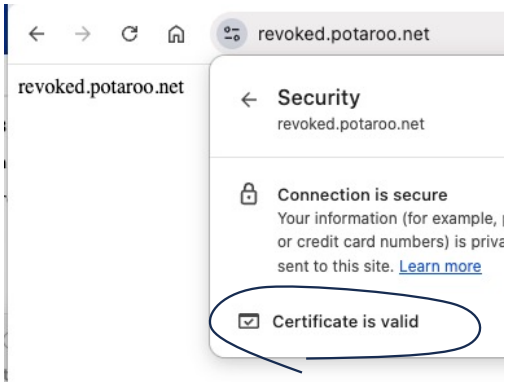
Safari



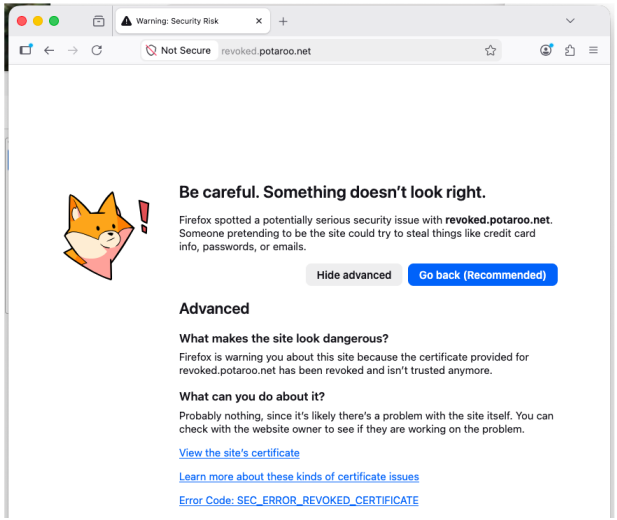
How is this working?

1. Let’s generate a LetsEncrypt certificate
2. And then revoke it 4 hours later
3. Wait for a couple of weeks and see if my browser has learned of this revocation

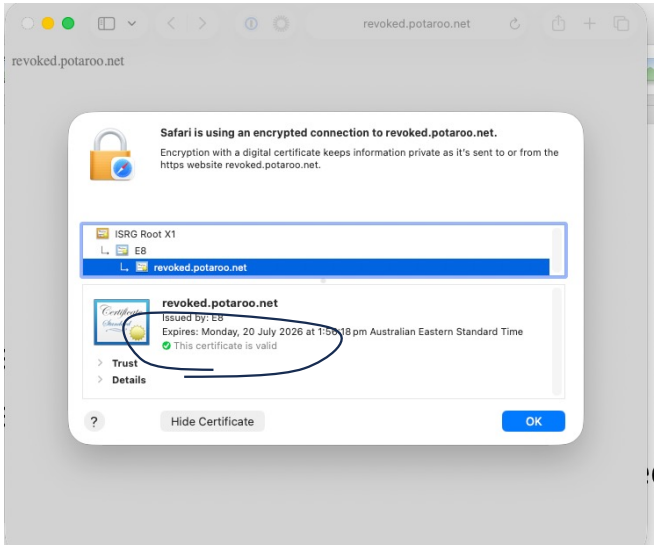
Chrome



Only Firefox!!!!!



Safari



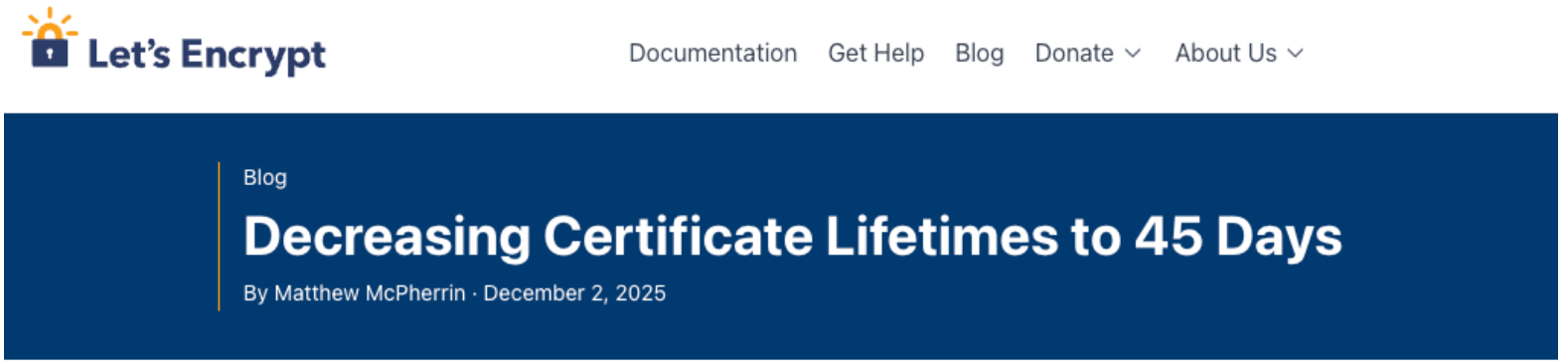
Revocation is a Failure!

- If the point of this entire certificate architecture is to inform the user that the location that they have reached is (or is not) the location that they intended to reach, then why is it useful at all if it can't inform the user that the certificate that is being used is not to be trusted **NOW?**
- If the best that CRLs, OCSP, Stapled OCSP and CRLsets can inform you is the trust status of a certificate ***at some time in the past*** then why is this information any different from the certificate itself?

Revocation is a Failure!

- If the entire purpose of these revocation mechanisms is only to reduce the “trust window” of a certificate, then why not just use certificates with a more constrained trust window?

"Shorter" Certificates



How Short can we go?

- If 90 days is ”too long” does 45 days improve the situation?
  - 45 days will cut down the CRL size
  - BUT it will not remove the need for CRLs as the window of vulnerability is still too long without some form of early revocation
- If 45 days is “too long” then would 6 days improve the situation?
  - 6 days will cut down the CRL size
  - BUT it will not remove the need for CRLs as the window of vulnerability is still too long without some form of early revocation
- Is there a “right” period for certificate lifetimes?

Certificates are a Failure!

- The problem with certificates that provide a trust window of a few hours, is that the existing CA infrastructure and the use models of locally stashed certificates just can't cope with such an increased intensity of certificate re-issuance
- Instead, we just persist with long-lived certificates and non-functional revocation mechanisms, because it's the path of least resistance
- If certificates are incapable of informing a client that they are about to be drawn into misplaced trust, then what exactly are they good for anyway?

Certificates are a Failure!

- The problem with certificates that provide a trust window of a few hours, is that the existing CA infrastructure and the use models of locally stashed certificates just can't cope with such an increased intensity of certificate re-issuance
- Instead, we just persist with long-lived certificates and non-functional revocation mechanisms, because it's the path of least resistance
- If certificates are incapable of informing a client that they are about to be drawn into misplaced trust, then what exactly are they good for anyway?
- The entire objective here was to answer the simple question: "**Is the service that I am about to connect to the service that I intended to connect to?**" And the problem is that this entire certificate structure can only answer a question that relates to the past, not the present!

Certificates are a Failure?

- The problem with certificates that provide a trust window of a few hours, is that the existing CA infrastructure and the use models of [illegible] ashed certificates just can't cope with such an [illegible] tificate re-issuance
- Instead [illegible] onal revocat[illegible]
- If certif[illegible] of informing a client that they are about to be drawn into misplaced trust, then what exactly are they good for anyway?
- The entire objective here was to answer the simple question: "**Is the service that I am about to connect to the service that I intended to connect to?**" And the problem is that this entire certificate structure can only answer a question that relates to the past, not the present!

if we can't fix these issues with X.509 certificates, then why do we bother using X.509 certificates at all?

Certificates are a Failure!

- The desire to leverage existing X.509 certificate infrastructure into the online world is an abject failure
  (some folk were telling us this for years, but we just weren't listening!)
- Is there an alternative?

There's the DNS…

The problem here is not TLS, and not the use of digital signatures to assure the veracity of information, but the properties of the certificate infrastructure to allow this veracity to be established as a **current** piece of information

The DNS has a similar issue of accuracy and timeliness

- It manages to contain this time lag though the use of Cache Timers (TTLs), managed by zone admins, to set a maximal time between referral back to the authoritative information sources
- There is a constant refresh of DNS information passed to the client from the authoritative servers

DANE!

- DNS-based Authentication of Named Entities does away with this 3rd party commentary through X.509 certificates used by the Web PKI
- The public key used to demonstrate authenticity is stored in the DNS itself as a DNSSEC-signed attribute of the name
- Key information can accessed on demand through the DNS (or provided to you in other ways!)
- Key information is validated by DNSSEC chaining which is inherent in the internal structure of the name itself
- Validity times can be managed with intervals of seconds and minutes, not months!

And then there's Stapled DANE …

If the entire purpose of this security measure is to associate a key pair with an intended service name, then why not just use the DNS?

- Place the public key of a service into the DNS alongside its other attributes
- Use conventional TTL mechanisms to control the maximal caching lag of this information
- Sign these records with DNSSEC
- And staple the DNSSEC validation chain of the public key record into the TLS handshake as a stapled chained response as an alternative to using X.509 certificates

i.e. use DANE, Chained Responses and TLS Stapling

Where have we got to?

- X.509 certificate revocation is broken **and we can't fix it**
- About the only fix is to pull certificate lifetimes down to a small number of hours – i.e. limit the scope of potential damage of a compromised certificate
  - But the certificate system as we know it won't scale with such massively shortened certificate lifetimes
- In the DNS, the common TTL is measured in hours
  - So why don't we just ditch all this X.509 brokenness and just turn all of this over to DANE?

The Road Not Followed

- It Netscape had thought of embedding public key cryptography into the DNS in 1994 with SSL 1.0, the entire X.509 web PKI would never have existed
  - Your name, and the way to authenticate the assertion that it's your name, would co-exist in the DNS

- But it's too late to change – so now we are on an inevitable path of figuring out how to cope with an infrastructure of X.509 certificates with validity intervals measured in minutes, and all that this entails!

Good Luck with That!

Questions?