Transport Layer Security in 2023

Michael W Lucas

https://mwl.io

Who Am I?

- Author
- Unix since 1986, 1987, something like that
- Sysadmin since 1995
- Founding member of SouthEast Michigan BSD User Group, semibug.org
- Author of many tech books, including *TLS Mastery*
- As Michael Warren Lucas, novels like *git commit murder*
- Martial arts, pet rats, married, blah blah blah

What's in this talk?

- What Is TLS?
- Certificates
- Revocation and Invalidation
- TLS Negotiation
- CSRs
- ACME
- Edge Cases

- Focuses on web servers, because we only have an hour

Tools in This Talk

- Current OpenSSL or LibreSSL
- CentOS, Debian, FreeBSD, OpenBSD

Transport layer security (TLS) doesn’t do any of this

- Add Security
- Block Intruders
- Keep Your Credit Card Secret
- Stop Password Theft

what tls does

- Encrypt traffic between client and server
- Identify server, client, or both

- That’s it. Nothing more.

tls history

- 1994: Netscape develops Secure Sockets Layer 1.0 in-house
- 1995: SSLv2 rushed out the door
- 1996: SSLv3 rushed out even faster
- 1999: TLS version 1 (name changed for political reasons)
- 2006: TLS 1.1
- 2008: TLS 1.2
- 2018: TLS 1.3

Postel's Robustness Principle

- "Be conservative in what you do, be liberal in what you accept from others."
- Web server offers HTTP 0.9, 1.0, 1.1, 2.0, 3.0? Great!
- Database engine supports older query versions? Dandy!
- Older SMTP features? Just don't allow spam, we're fine.
- Older DNS query features? Meh, but okay, I guess, if you don't allow reflection attacks.

Older TLS versions?

Downgrade Attacks

- Convincing a server to support cryptographic methods that an intruder can somehow decrypt
- Applicable to many encryption technologies, not just TLS
- Solution: refuse to accept weak encryption

Weak TLS Versions

- All versions of TLS except 1.2 and 1.3 are known to be breakable and officially deprecated. Supporting older versions threatens your customers' data, your business, and everyone's privacy.
- In early 2021, the United States National Security Agency strongly discouraged use of many common TLS 1.2 configurations.
- We can't turn off 1.2 on general Internet sites.
- No FIPS-compliant TLS 1.3 browser

Encryption: What You Must Know

- A highly specialized skill with many pieces
- Very few cryptographer SREs or sysadmins

- Protect the private keys!
- Revoke at any excuse
- OpenSSL's HIGH cipher list

Encryption Trust Models

- Deciding who to trust is *the* problem of cryptography
- Two competing models
  - The Web of Trust
  - Certificate Authorities

Certificate Authorities

- Verify identities of people, hosts, and organizations
- Each has a *trust anchor* certificate
- issue X.509 certificates for hosts and users declaring that they have performed the audit, and signs each with their trust anchors

- Client software has a bundle of trust anchors, ultimately trusted certificates
- Six major trust anchor bundles: Microsoft, Apple, Google, Mozilla, Adobe, and Oracle
- Different software uses different trust anchor bundles
- Not all CAs are in all trust bundles

OpenSSL Trust Bundle

- Per-install, per-Unix basis
- Most use Mozilla bundle
- Every OS has its own way to manage trust bundles—ALWAYS read the documentation!
  - certctl?
  - add-trusted-cert?
  - update-ca-certificates?
  - Raw OpenSSL commands?
  - Dump in directory? Overwritten!
- Use `openssl -CAfile` option to trust a specific cert in a specific command

Trust Bundles

- Look at the contents of any trust anchor bundle.

WHO ARE THESE PEOPLE???

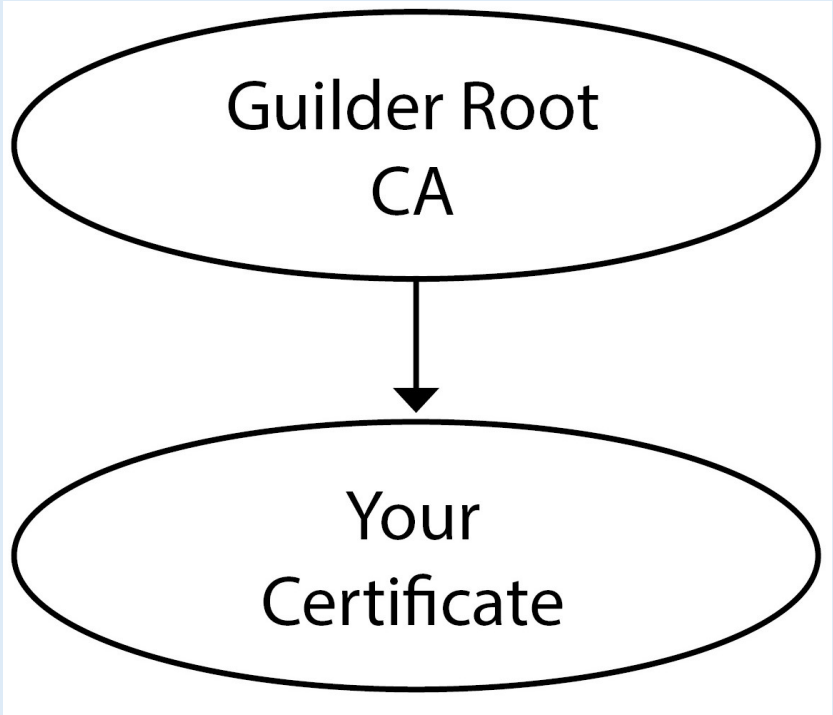
- Should I trust the Shanghai Electronic Certification Authority?
- Should I trust the French government? Or the US government? China, Netherlands, Indian, Nigerian?

Private Trust Bundle

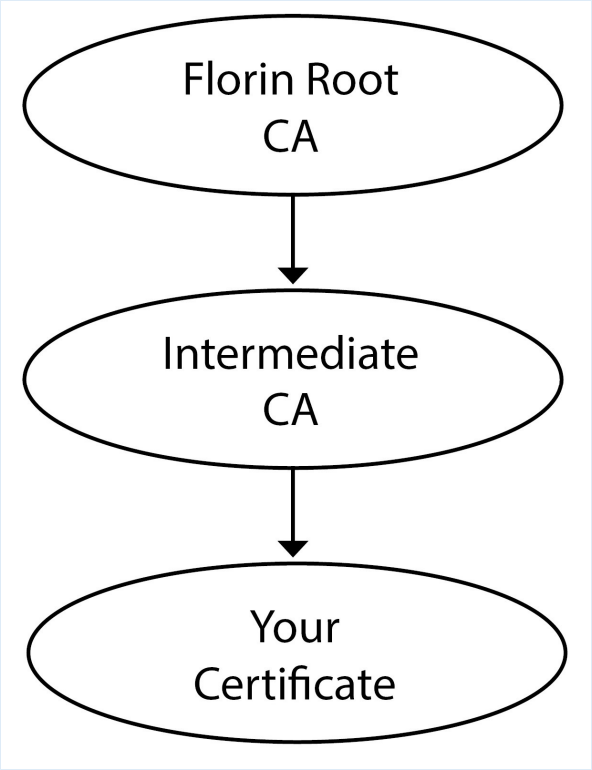
- Curating your own trust bundle for web browsing?
  - Pointless, but educational

- Curating a trust bundle for applications? Useful.
- Select a single supplier for application certificates. Install that supplier’s trust anchor on all clients, remove all others.

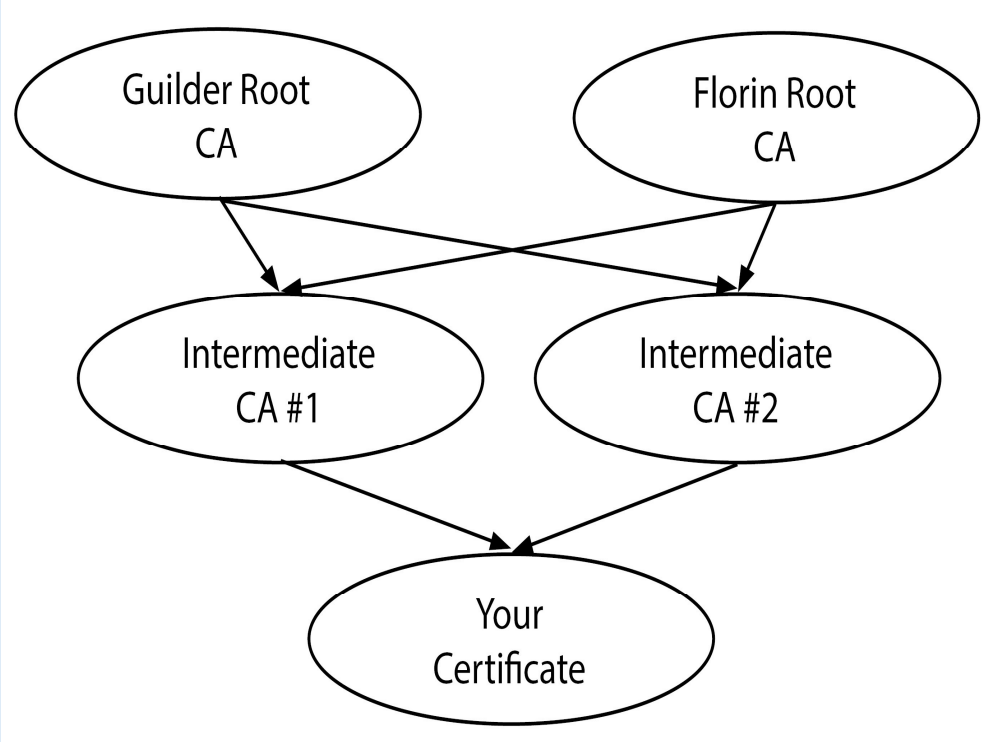
Simple Chain of Trust



Chain of Trust with Intermediate Certificates



Modern Chain of Trust



Certificates

- X.509 – the International Telecommunication Union's digital certificate standard.
- Built on ASN.1, the same standard used by SNMP and telecom firms.
- Do not need to know ASN.1, except when you see something like .1.3.6.1.4.1.44947.1.1.1
- Also pillages X.500 directory standard, used by LDAP
- Gives information labels like OU=, O=, ST=, and so on
- Each label has an ASN.1 identifier
- Expires, usually every year or 90 days

Certificate Components

- Certificate Signing Request – the “please sign me” form you send to the CA. Irrelevant once you have the certificate. Includes a public key, comes with a private key file
- Private Key – the secret for your certificate
- Certificate File – the signed extract of CSR information
- Server needs both Certificate and Private Key

- If an intruder captures your private key file, they can use the certificate and pretend to be you.
- PROTECT THE PRIVATE KEY

Certificate Validation Level

- Domain Validated — the CA verifies that the entity requesting the cert controls the host the cert is for
- Organization Validated – the CA verifies that the entity requesting the cert belongs to the organization that controls the host the cert is for
- Extended Validation – the CA verifies that the organization requesting the cert is a specific legal entity

- All of them provide the browser lock icon

Certificate Algorithms

- RSA – the “old trusty” standard. Key length dictates strength, 2048 bits is current standard.
- ECDSA – newer standard. Does not use key length. Uses named elliptic curves instead. Roughly as strong as RSA, but requires less computing to validate.

- Use ECDSA if you’re targeting constrained computing environments, like mobile devices

Common Names

- Most widely recognized part of a certificate
- Could be an email address, UID, first and last name, a device serial number

- CN was used to store hostnames until 2000, when this was deprecated.
- People still use this. It is wrong.
- Some software expects to find a hostname there.
- Being actively exterminated.

Why is hostname in CN Deprecated?

- CN can have a maximum of 63 characters
- Domain names can be longer than that.
- I have the domain YouKeepUsingThatWordIDoNotThinkItMeansWhatYouThinkItMeans.com
- The use of hostnames in CN is holding back TLS improvements
- CN is not checked against constraints
- Replacement is called Server Alternative Names, or SAN

Viewing Certificates

```
$ openssl x509 -in file.crt -text -noout
```

- spills certificate contents
- see certificate chain, identified entities, extensions, and more

Modern Certificate Hostnames

- Subject Alternative Names (SAN)
- Stuff beyond CN

```
$ openssl x509 -in mwlio.cer -noout -ext subjectAltName
X509v3 Subject Alternative Name:
    DNS:cdn.mwl.io, DNS:mwl.io, DNS:www.mwl.io
```

Wildcard Certificates

- A certificate for any host in a domain or subdomain, like *.mwl.io
- Sounds great, right?
  - Must put private key on all hosts that use the certificate
  - One compromise, everything is compromised
  - if the intruder is sneaky, you'll never know
- Safest use, for devops and such: *.api.mwl.io

Revoking Certificates

- If your private key is compromised, tell your CA: “revoke” the cert
- Distributed by two different methods, Certificate Revocation Lists and Online Certificate Status Protocol.
- If you’re using ACME with Let’s Encrypt, or your CA offers revocation insurance, revoke on any doubt

- Firefox and Safari distribute revoked cert lists to clients
- Chrome filters the revocation list Chrome users see

#include <chrometantrum.h>

Certificate Revocation Lists

- CA distributes a file containing serial numbers of all unexpired revoked certificates
- Might be 75MB, might be 1MB
- Can be cached for time set by CA, usually 24 hours
- Usually in DER format, to reduce size
- Doesn’t scale well

Online Certificate Status Protocol

- Check individual certificate, no giant list to download
- Digitally signed, can operate over plain HTTP
- Valid for 7 days

- Server can make OCSP queries, attach to certificate, and return with client request – called “stapling”
- Client skips OCSP call on stapled cert
- Server can declare “must-staple” – no staple, cert is invalid
- Must enable OCSP on Chrome & Safari
- Chrome ignores must-staple

If Revocations are Ignored, What Now?

- Short-lived certificates (name constraint certificates)
- DANE
- Smack vendors
- Deploy browsers or forks that enable secure features
- store private keys in Hardware Security Modules (HSM)

Getting Certificates

- Certificate Signing Requests (CSR)
- Contain all the information that the CA must validate
- That friendly OpenSSL dialog that gives you a CSR?
  - It puts the hostname in CN.
  - It supports only one hostname.
  - CN in hostname has been obsolete since 2000. We're really trying to kill it.
  - Don't use it.
- Use the modern, config-file based method. Or generate wholly on the command line.

Reusing CSRs

- It is technically possible to reuse previous year's CSR to renew your certificate
- It is HORRIBLE practice
- The CSR is tied to a specific private key
- Are you certain that key did not leak?
- Have key standards changed since last year?
- CSRs are free. Keys are free. Creation can be scripted.

Make new keys and new CSRs.

Information in your CSR

- Only include the information you must validate in your CSR.
- DV cert – only need hostnames
- OV, EV – also need organization information

CSR X.500 Information

- C= two-letter country code from ISO 3166
- ST = state or province, spelled out
- L = locality, city. Spell out official city name
- O= organization, company name

- Gather official names before starting your CSR

RSA Config File

```
[ req ]
prompt                  = no
default_bits            = 2048
default_keyfile         = mwl.io-private.key
distinguished_name      = req_distinguished_name
req_extensions          = v3_req

[ req_distinguished_name ]
CN = mwl.io

[ v3_req ]
subjectAltName                  = @alt_names

[alt_names]
DNS.1                           = mwl.io
DNS.2                           = www.mwl.io
```

Using the CSR Config File

```
$ openssl req -newkey rsa -config mwl.io.conf \
      -out mwl.io.csr
Generating a RSA private key
.............+++++
.....................................................
...........+++++
writing new private key to 'mwl.io-private.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
```

- Generates a CSR in mwl.io.csr and a private key named after config file entry

Viewing CSRs

```
$ openssl req -in mwl.io.csr -noout -text
Certificate Request:
Data:
    Version: 1 (0x0)
    Subject: CN = www.mwl.io
    Subject Public Key Info:
    Public Key Algorithm: id-ecPublicKey
…
```

CSRs Without Config Files

```
$ openssl req -newkey rsa:2048 -keyout www-private.pem \
        -out www-request.csr -subj /CN=mwl.io –addext \
        "subjectAltName=DNS:mwl.io,DNS:www.mwl.io,DNS:*.cdn.mwl.io"
```

Reconnecting Files

- Scrambled previous years' certificates and private key files, and don't know which go with which?
- A keypair's modulus is a shared numerical value unique to that key
- Monitoring!

```
$ openssl x509 -noout -modulus -in server.crt | openssl md5
d261c8136ff4154881814df84cc0829a
$ openssl rsa -noout -modulus -in server.key | openssl md5
d261c8136ff4154881814df84cc0829a
$ openssl req -noout -modulus -in server.csr | openssl md5
2606bbdc0f7b099117342eefd049d5b0
```

Automated Certificate Management Protocol (ACME)

- Created by Internet Security Research Group (ISRG)
- ISRG founded free CA “Let’s Encrypt.”
- LE provides free 90-day DV X.509 certificates to all users

How ACME Works

- Domain Validation: verify that the entity requesting the certificate controls the host.
- This can be automated.
  - Client: “I’d like a certificate for this host”
  - CA: “Oh yeah? Make this change on the host to prove you control the host.”
  - Client: “The host is changed.”
  - CA: “I guess you do control the host. Send me your CSR.”
  - Client: “Here you go.”
  - CA: “And here’s your 90-day certificate. See me again in 60 days.”

ACME Challenges

- HTTP-01 – “Put this file in
  http://server/.well-known/acme/challenge/”
- DNS-01 – “Make this change in _acme-challenge.domain.name.”
- ALPN-01 – “Make this change in TLS protocol ID such-and-such”

Which should I use?

- Cert used on single server? HTTP-01
- Wildcard certs? DNS-01
- Load balancers, server pools, clustering, general mayhem? ALPN-01

ACME Renewals

- Try to renew weekly, starting 2/3 of the way through certificate lifetime
- Gives you four weeks to resolve transient issues
- Schedule in cron
- Most clients handle this

ACME Clients

- certbot – official Let's Encrypt client, in python
- Apache – mod_md for ACME inside web server
- OpenBSD acme_client(1)
- Docker container that handles ACME for you

- My favorite: dehydrated.io. Extensible, configurable, lightweight, can run on rPi without strain

TLS Potpourri

- Get all your servers up to TLS 1.2/1.3
- Then investigate these

HTTP Strict Transport Security (HSTS)

- HTTP header that says “This web site only serves pages over TLS. Please cache this information.”
- Client rewrites all HTTP requests into HTTPS, refuses HTTP from this hostname
- Web-only

TLS Testing

- https://www.ssllabs.com
- Tests server configuration, looks for bad TLS versions, broken ciphers, and general misconfiguration
- Only works on public web sites. Test might set off intrusion detection

- Private testing software? https://testssl.sh

Running Your Own CA

- Corporate CA? Many companies do it.
- Small CA from openssl-ca(1)? A great learning experience, but not intended for long-term use, multiple users, or exposure to the Internet.
- Use software like easy-rsa, XCA, Dogtag.
- Big organization? FreeIPA or EJBCA.

Name Constraint CAs

- Big Company can buy a signing cert that allows them to sign certs in their domain.
- I could buy a cert that lets me sign *.mwl.io. Each cert is only good for ten days, or 24 hours, or 12 hours
- Requires extensive reliable automation.
- Currently expensive. Like DV certificates, will get cheaper.

That’s it!

- TLS is huge, but: any sysadmin-level questions?

- Want more? My book *TLS Mastery* is out everywhere

- My bookstore: https://tiltedwindmillpress.com
- My web site: https://mwl.io