The Foundations of Network Security

Geoff Huston AM
Chief Scientist, APNIC
The Foundations of Network Security

insecurity!

Geoff Huston AM
Chief Scientist, APN\'IC
1. The Current Security Framework for the Internet
Let's start with a simple example:
Why should you pass your account and password to this web site? It might look like your bank, but frankly it could just as easily be a fraudulent site intended to steal your banking credentials. Why should you trust what you see on the screen?
Which Bank? My Bank!

Ok – it’s not a random example. It’s the online bank I use! But the same question is still there. Why should I trust this web page?
Security on the Internet

How do you know that you are really going to where you thought you were going to?

It's trivial to create a web page to look exactly like another.
Security on the Internet

How do you know that you are really going to where you thought you were going to?

So why should I enter my username and password into this particular screen?

It's trivial to create a web page to look exactly like another.

And what does this padlock icon really mean?
Opening the Connection: First Steps

Client:

**DNS Query:**

www.commbank.com.au?

**DNS Response:**

104.116.164.218

**TCP Session:**

TCP Connect 104.116.164.218, port 443
Hang on...

Who “owns” that IP address? The Commonwealth Bank? Someone else?

Let’s look at little more:

$ dig -x 104.116.164.218 +short
a104-116-164-218.deploy.static.akamaitechnologies.com
$ dig -x 104.116.164.218 +short
a104-116-164-218.deploy.static.akamaitechnologies.com

That’s not an IP addresses that was allocated to the Commonwealth Bank!

The Commonwealth Bank of Australia has the address blocks
140.168.0.0 - 140.168.255.255 and
203.17.185.0 - 203.17.185.255
Hang on...

$ dig -x 104.116.164.218 +short @::1
a104-116-164-218.deploy.static.akamaitechonologies.com

That’s an Akamai IP address

And I’m NOT a customer of the Internet Bank of Akamai!

Why should my browser trust that 104.116.164.218 is really the authentic web site for the Commonwealth Bank of Australia, and not some dastardly evil scam designed to steal my passwords and my money?

And why should I trust my browser?
The major question...

How does my browser tell the difference between an intended truth and a dastardly lie?
It's all about cryptography
Public Key Cryptography

Pick a pair of keys such that:

– Messages encoded with one key can only be decoded with the other key
– Knowledge of the value of one key does not infer the value of the other key
– Make one key public, and keep the other a closely guarded private secret
This is important

So I will repeat it:

– Using public/private key cryptography requires a pair of keys (A,B) such that:
  • Anything encrypted using key A can ONLY be decrypted using key B, and no other key
  • Anything encrypted using key B can ONLY be decrypted using key A, and no other key
  • Knowing the value of one key WILL NOT let you work out the value of the other key anytime soon!

This form of asymmetric cryptography lies at the heart of the Internet’s security framework
If I have a copy of your PUBLIC key, and you encrypt a message with your PRIVATE key, and I can decrypt the message using your PUBLIC key, then

– I know no one has tampered with your original message
– And I know it was you that sent it.
– And you can’t deny it.

If we negotiate a session key using the combination of your public key and a local private session key and encrypt all session messages using this session key, then

– I am confident no one else can eavesdrop on our conversation in this session
Public Key Certificates

But how do I know this is YOUR public key?
   – And not the public key of some dastardly evil agent pretending to be you?

• I don’t know you
• I’ve never met you
• So, I have absolutely no clue if this public key value is yours or not!
Public Key Certificates

What if I ‘trust’ an intermediary*?

– Who has contacted you and validated your identity and conducted a ‘proof of possession’ test that you have control of a private key that matches your public key

• If this trusted intermediary signs an attestation that this is your public key (with their private key) then I would be able to trust this public key

• This ‘attestation’ takes the form of a “public key certificate”

* If you have ever used “public notaries” to validate a document, then this is a digital equivalent
I trust that this is the web site of the Commonwealth Bank because I used the Commonwealth Bank’s public key to set up the encrypted connection to the server.

And I can trust that this is the Commonwealth Bank’s public key because I trust that Entrust has performed a number of checks before issuing a public key certificate for this public key.
And another example

- Let’s take www.apnic.net and look at that certificate
This certificate was issued to Cloudflare, not APNIC, and it is associated with the name "www.apnic.net" through the use of a Subject Alternative Name in the certificate.
And another

• Let’s look at my own web site, with its certificate issued by Let’s Encrypt
This certificate binds a public key to a domain name without any attestation to the identity of the name “holder.”
Spot the Difference
Spot the Difference

This web site’s certificate was issued to an organisation called the “Commonwealth Bank of Australia” located in Sydney, Australia.

This web site’s certificate was issued to “Cloudflare Inc” located in San Francisco, USA!!

This web site’s certificate says *nothing* about the entity that holds the public key associated with this domain.
Spot the Difference

• The certification processes used to issue the certificate were different in each of these cases.
  – One confirmed the identity of the public key holder as well as their association with the domain name
  – The second used a proxy agent and there is no association between the entity domain name that is certified here and the proxy agent
  – The third simply associates a public key with a domain name without any form of identification of the holder of the domain name

• They have very different levels of trustworthiness, yet they all display to the user in exactly the same way
  – Because when we tried to differentiate these different levels of trust (such as painting the padlock icon in green) nobody understood what was going on and nobody cared anyway!
Spot the Difference

• While there are important differences in the trustworthiness in these three certificates, they all display on the user’s screen in precisely the same way

• As an attacker, if I can use the lowest threshold of proof to have a counterfeit certificate issued, then perhaps there is a viable attack vector, as the user would not notice the switch to a less strict form of subject identity validation
Moving on...

• Ok, so the certificate system is a mess, but the subsequent secure transport session (TLS) still works, right?
• Let’s look at the way TLS starts a secure session
Secure Connections using TLS

TLS Client

- **ClientHello**
  Offers TLS version, list of ciphers, compression methods etc

- **ServerHello**
  Server chooses TLS version, ciphers, compression method. Server sends its certificate

- **ServerHelloDone**

- **ClientKeyExchange**
  Secret PreMasterKey encrypted using Server’s public key

TLS Server

- **Server decrypts message using previously exchanged keys**

- **ChangeCipherSpec**

- **Finished**

- **ChangeCipherSpec**

- **Finished**

---

https://rhsecurity.wordpress.com/tag/tls/
Secure Connections using TLS

Diagram:
- **ClientHello**: Offers TLS version, list of ciphers, compression methods, etc.
- **ServerHello**: Server chooses TLS version, cipher, compression method, sends certificate.
- **ServerHelloDone**: Server sends its certificate.
- **ClientKeyExchange**: Secret from master key encrypted using server's public key.
- **ChangeCipherSpec**: Client decrypts message using previously exchanged keys.
- **Finished**: ChangeCipherSpec
- **Server**: Server decrypts message using previously exchanged keys.
- **Finished**: ChangeCipherSpec
Secure Connections using TLS

[Diagram of TLS handshake]

- **ClientHello**: Offers TLS version, list of ciphers, compression methods, etc.
- **ServerHello**: Server chooses TLS version, cipher, compression method, and Server's certificate.
  - **ServerHelloDone**: Server sends certificate and cipher to client.
- **ClientKeyExchange**: Client sends Private Key encrypted using Server's public key.
- **ChangeCipherSpec**: Client decrypts message using previously exchanged keys.
- **Finished**: Server decrypts message using previously exchanged keys.

https://rhsecurity.wordpress.com/tag/tls/
<table>
<thead>
<tr>
<th>Subject Name</th>
<th>Country or Region</th>
<th>Country</th>
<th>New South Wales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Localy</td>
<td></td>
<td></td>
<td>Sydney</td>
</tr>
<tr>
<td>Inc. Country/Region</td>
<td></td>
<td></td>
<td>AU</td>
</tr>
<tr>
<td>Organisation</td>
<td></td>
<td></td>
<td>Commonwealth Bank of Australia</td>
</tr>
<tr>
<td>Business Category</td>
<td></td>
<td></td>
<td>Private Organization</td>
</tr>
<tr>
<td>Serial Number</td>
<td></td>
<td></td>
<td>48 123 123 124</td>
</tr>
<tr>
<td>Common Name</td>
<td></td>
<td></td>
<td><a href="http://www.commbank.com.au">www.commbank.com.au</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Issuer Name</th>
<th>Country or Region</th>
<th>Country</th>
<th>US</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organisation</td>
<td></td>
<td></td>
<td>Entrust, Inc.</td>
</tr>
<tr>
<td>Organisational Unit</td>
<td></td>
<td></td>
<td>See <a href="http://www.entrust.net/legal-terms">www.entrust.net/legal-terms</a></td>
</tr>
<tr>
<td>Common Name</td>
<td></td>
<td></td>
<td>Entrust Certification Authority - LIM</td>
</tr>
<tr>
<td>Serial Number</td>
<td></td>
<td></td>
<td>24 F5 4B 83 F7 9F 29 57 72 A5 F1 CF 8F 3D E7 A8</td>
</tr>
<tr>
<td>Version</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Signature Algorithm</td>
<td></td>
<td></td>
<td>SHA-256 with RSA Encryption ( 1.2.840.113549.1.1.11 )</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
<td></td>
<td>None</td>
</tr>
<tr>
<td>Not Valid Before</td>
<td></td>
<td></td>
<td>Wednesday, 30 March 2022 at 10:09:12 am Australian Eastern Daylight Time</td>
</tr>
<tr>
<td>Not Valid After</td>
<td></td>
<td></td>
<td>Saturday, 29 April 2023 at 9:59:12 am Australian Eastern Standard Time</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Public Key Info</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm</td>
<td>RSA Encryption ( 1.2.840.113549.1.1.1 )</td>
</tr>
<tr>
<td>Parameters</td>
<td>None</td>
</tr>
<tr>
<td>Public Key</td>
<td>256 bytes: BF 7E 21 BA 6C E0 A4 90 ...</td>
</tr>
<tr>
<td>Exponent</td>
<td>00537</td>
</tr>
<tr>
<td>Key Size</td>
<td>2048 bits</td>
</tr>
<tr>
<td>Key Usage</td>
<td>Encrypt, Verify, Wrap, Derive</td>
</tr>
<tr>
<td>Signature</td>
<td>256 bytes: C5 28 89 A4 13 91 80 8A ...</td>
</tr>
</tbody>
</table>
Secure Connections using TLS

How does the client recognise this certificate as the “right” certificate?
How did my browser know that this is a "valid" cert?
Domain Name Certification

- The Commonwealth Bank of Australia has generated a Public/Private key pair
- And they passed a certificate signing request to a company called “Entrust” in the US
- Who was willing to vouch (in a certificate) that the entity is called the Commonwealth Bank of Australia and they have control of the domain name www.commbank.com.au and they have a certain public key
- So, if I can associate this public key with a connection then I have a high degree of confidence that I’ve connected to an entity that is able to demonstrate knowledge of the private key for www.commbank.com.au, as long as I am prepared to trust Entrust and the certificates that they issue
- And I’m prepared to trust them because Entrust NEVER lie!
Domain Name Certification

• The Commonwealth Bank of Australia has generated a Public/Private key pair
• And they passed a certificate signing request to a company called “Entrust” in the US
• Who was willing to vouch (in a certificate) that the entity is called the Commonwealth Bank of Australia and they have control of the domain name www.commbank.com.au and they have a certain public key
• So, if I can associate this public key with a connection then I have a high degree of confidence that I’ve connected to an entity that is able to demonstrate knowledge of the private key for www.commbank.com.au, as long as I am prepared to trust Entrust and the certificates that they issue
• And I’m prepared to trust them because Entrust NEVER lie!

How do I know that? Why should I trust them?
Local Trust

The cert I'm being asked to trust was issued by a certification authority that my browser already trusts — so I trust that cert!
Local Trust

These Certificate Authorities are listed in my computer’s trust set because they claim to operate according to the practices defined by the CAB industry forum (of which they are a member) and they never lie!
Local Trust

These Certificate Authorities are listed in my computer’s trust set because they claim to operate according to the practices defined by the CAB industry forum (of which they are a member) and they **never lie**!

So somebody (I have never met) paid someone else (whom I have also never met) some money and then my browser trusts everything they have ever done and everything they will ever do in the future – ok?
Local Trust or Local Credulity*

Wow!

Are they all trustable?

* credulity

(a tendency to be too ready to believe that something is real or true.)
Local Credulity

Wow!
Are they all trustable?

Evidently Not!
Local Credulity

Wow!
Are they all trustable?

Evidently Not!

The real security issue behind the Comodo hack

The Comodo hack has grabbed headlines, but more troubling is the public’s ignorance over PKI and digital certificates

MORE LIKE THIS
Weaknesses in SSL certification exposed by Comodo security breach
Hackers target Google, Skype with rogue SSL certificates
Revise certificates when you need to - the right way

News of an Iranian hacker duping certification authority Comodo into issuing digital certificates to one or more unauthorized parties has caused an uproar in the IT community, moving some critics to call for Microsoft and Mozilla to remove Comodo as a trusted root certification authority from the systems under their control. Though the incident highlights the potential for compromising a site containing a hard-coded login name and password, then generating certificates for several well-known sites, including Google, Live.com, Skype, and Yahoo, I’m not bothered by the
Never?
Well, hardly ever

Distrust of the Symantec PKI: Immediate action needed by site operators
March 7, 2018

Posted by Devon O’Brien, Ryan Sleevi, Emily Stark, Chrome security team

We previously announced plans to deprecate Chrome’s trust in the Symantec certificate authority (including Symantec-owned brands like Thawte, VeriSign, Equifax, GeoTrust, and RapidSSL). This post outlines how site operators can determine if they’re affected by this deprecation, and if so, what needs to be done and by when. Failure to replace these certificates will result in site breakage in upcoming versions of major browsers, including Chrome.

Chrome 66

If your site is using a SSL/TLS certificate from Symantec that was issued before June 1, 2016, it will stop functioning in Chrome 66, which could already be impacting your users.

If you are uncertain about whether your site is using such a certificate, you can preview these changes in Chrome Canary to see if your site is affected. If connecting to your site displays a certificate error or a warning in DevTools as shown below, you’ll need to replace your certificate. You can get a new certificate from any trusted CA, including DigiCert, which recently acquired Symantec’s CA business.
These are isolated events

No, they’re not:

https://www.feistyduck.com/ssl-tls-and-pki-history/
With unpleasant consequences when it all goes wrong
With unpleasant consequences when it all goes wrong

Cuba aimed at U.S. per husband not to stay right here with anything happens, told him in October to be with you, and I you, and the children without you.”

Interview conducted of only three that after Mr. Kennedy’s established as a published as a "His-

Iranian activists feel the chill as hacker taps into e-mails

BY SOMINI SENGUPTA

He claims to be 21 years old, a student of software engineering in Tehran who reveres Ayatollah Ali Khamenei and despises dissidents in his country.

He sneaked into the computer systems of a security firm on the outskirts of Amsterdam. He created fake credentials that could allow someone to spy on Internet connections that appeared to be secure. He then shared that bounty with people he declines to identify. The fruits of his labor are believed to be worth $30,000.

Volatility is the new market norm

Large swings in share prices are more common now than at any other time in recent stock market history. PAGE 16
Suspicious event hijacks Amazon traffic for 2 hours, steals cryptocurrency

Almost 1,300 addresses for Amazon Route 53 rerouted for two hours.

DAN GOODIN  -  4/25/2018, 6:00 AM

Amazon lost control of a small number of its cloud services IP addresses for two hours on Tuesday morning when hackers exploited a known Internet-protocol weakness that let them to redirect traffic to rogue destinations. By subverting Amazon's domain-resolution service, the attackers masqueraded as cryptocurrency website MyEtherWallet.com and stole about $150,000 in digital coins from unwitting end users. They may have targeted other Amazon customers as well.

The incident, which started around 6 AM California time, hijacked roughly 1,300 IP addresses, Oracle-owned Internet Intelligence said on Twitter. The malicious redirection was caused by fraudulent routes that were announced by Columbus, Ohio-based eNet, a large Internet service provider that is referred to as autonomous system 10297. Once in place, the eNet announcement caused Hurricane Electric and possibly Hurricane Electric customers and other eNet peers to send traffic over the same unauthorized routes. The 1,300 addresses belonged to Route 53, Amazon's domain name system service.

The attackers managed to steal about $150,000 of currency from MyEtherWallet users.
What's going wrong here?
What's going wrong here?

• The TLS handshake cannot specify WHICH CA should be used by the client to validate the digital certificate that describes the server’s public key
• The result is that your browser will allow ANY CA to be used to validate a certificate!
What's going wrong here?

• The TLS handshake cannot specify WHICH CA should be used by the client to validate the digital certificate that describes the server’s public key

• The result is that your browser will allow ANY CA to be used to validate a certificate!

WOW! That's awesomely bad!
What's going wrong here?

• The TLS handshake cannot specify WHICH CA should be used by the client to validate the digital certificate that describes the server's public key.

• The result is that your browser will allow ANY CA to be used to validate a certificate!

WOW! That's awesomely bad!
What's going wrong here?

• There is no incentive for quality in the CA marketplace
• Why pay more for any certificate when the entire CA structure is only as strong as the weakest CA
• And your browser trusts a LOT of CAs!
  – About 60 – 100 CA’s
  – About 1,500 Subordinate RA’s
  – Operated by 650 different organisations

See the EFF SSL observatory
http://www.eff.org/files/DefconSSLiverse.pdf
In a Commercial Environment

Where CA’s compete with each other for market share
And quality offers no protection
Then what ‘wins’ in the market?

Sustainable
Resilient
Secure
Privacy
Trusted
In a Commercial Environment

Where CA’s compete with each other for market share
And quality offers no protection
Then what ‘wins’ in the market?

Sustainable
Resilient
Secure
Privacy
Trusted
Cheap!
But it's all OK

Really.

• Because ‘bad’ certificates can be revoked
• And browsers always check revocation status of certificates before they trust them
Always?
Ok - Not Always.
Some do.
Sometimes.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Chrome</th>
<th>Firefox</th>
<th>Opera</th>
<th>Safari</th>
<th>Edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mac OS X</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>10.15.3</td>
<td>80.0.3987.132</td>
<td>73.0.1</td>
<td>67.0.3575.53</td>
<td>13.0.5</td>
<td></td>
</tr>
<tr>
<td>iOS</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>13.3.1</td>
<td>80.0.3987.95</td>
<td>23.0</td>
<td>16.0.15</td>
<td>13.3.1</td>
<td></td>
</tr>
<tr>
<td>Android</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>80.0.3987.132</td>
<td>68.6.0</td>
<td>56.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Windows</td>
<td>NO</td>
<td>YES</td>
<td>NO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>80.0.3987.132</td>
<td>74.0</td>
<td>67</td>
<td>44.18362</td>
<td></td>
</tr>
</tbody>
</table>

*Table 1 - Browser Revocation Status*
So, we can't count on revocation

• If we can’t revoke certificates, then we need to reduce certificate lifetimes
So, we can't count on revocation

- If we can’t revoke certificates then we need to reduce certificate lifetimes
- But we are not doing that!

Yes, 2026!!!
So, we can't count on revocation

- If we can’t revoke certificates then we need to reduce certificate lifetimes
- What’s a “safe” certificate lifetime?
So, we can't count on revocation

• If we can’t revoke certificates then we need to reduce certificate lifetimes
• What’s a “safe” certificate lifetime?
• If we want 2 hours or less, then we need to think hard about how to achieve this
How can we make certificates better?

Option A: Take all the money out of the system!
How can we make certificates better?

Option A: Take all the money out of the system!

Will the automation of the Certificate Issuance coupled with a totally free service make the overall environment more or less secure?

I think we already know the answer!
How can we make certificates better?

Option B: White Listing and Pinning with HSTS

https://code.google.com/p/chromium/codesearch#chromium/src/net/http/transport_security_state_static.json
How can we make certificates better?

Option B: White Listing and Pinning with HSTS

https://code.google.com/p/chromium/codesearch#chromium/src/net/transport_security_state_static.json

It's not a totally insane idea -- until you realise that it appears to be completely unscaleable!

It's just Google protecting itself and no one else.
How can we make certificates better?

Option B: White Listing and Pinning with HSTS

It's not a totally insane idea -- until you realise that it appears to be completely unscaleable!


It's just Google protecting itself and no one else

Google moves into the Certificate Authority business

Google doesn't seem to trust the current system, as it has launched its own security certificates

```cpp
17 // reports will be in the format defined in RFC 7469
18 //
19 // For a given pinset, a certificate is accepted if at least one of the
20 // "static_spki_hashes" SPKIs is found in the chain and none of the
21 // "bad_static_spki_hashes" SPKIs are. SPKIs are specified as names, which must
22 // match up with the file of certificates.
```
How can we make certificates better?

Option C: Certificate Transparency
How can we make certificates better?

Option C: Certificate Transparency

In order to provide encrypted traffic to users, a site must first apply for a certificate from a trusted Certificate Authority (CA). This certificate is then presented to the browser to authenticate the site the user is trying to access. In recent years, due to structural flaws in the HTTPS certificate system, certificates and issuing CAs have proven vulnerable to compromise and manipulation. Google's Certificate Transparency project aims to safeguard the certificate issuance process by providing an open framework for monitoring and auditing HTTPS certificates.

This is true

This is a fail
How can we make certificates better?

Option C: Certificate Transparency

It's just so broken. These transparency logs are a case of same week service in a millisecond world -- assuming anyone looks in the first place!

Cert Transparency is probably worse than a placebo!
How can we make certificates better?

Option D: Use the DNS!
Seriously? The DNS?

Where better to find out the public key associated with a DNS-named service than to look it up in the DNS?

– Why not query the DNS for the HSTS record?
– Why not query the DNS for the issuer CA?
– Why not query the DNS for the hash of the domain name cert?
– Why not query the DNS for the hash of the domain name public key?
Seriously? The DNS?

Where better to find out the public key associated with a DNS-named service than to look it up in the DNS?

– Why not query the DNS for the HSTS record?
– Why not query the DNS for the issuer CA?
– Why not query the DNS for the hash of the domain name cert?
– Why not query the DNS for the hash of the domain name public key?

Who needs CA's anyway?
• Using the DNS to associated domain name public key certificates with domain name
TLS with DANE

• Client receives server cert in Server Hello
  – Client lookups the DNS for the TLSA Resource Record of the domain name
  – Client validates the presented certificate against the TLSA RR
• Client performs Client Key exchange
TLS Connections

1. DNS Name
2. DNS TLSA query
3. Public Key
   - Cert

Flow Diagram:
- **Client**
  - ClientHello
  - ServerHello
  - ServerHelloDone
  - ClientKeyExchange
  - ChangeCipherSpec
  - Finished
- **Server**
  - ServerHello
  - ServerHelloDone
  - ChangeCipherSpec
  - Finished

Notes:
- ClientHello: Offers TLS version, list of ciphers, compression methods etc
- ServerHello: Server chooses TLS version, cipher, compression method, Server sends its certificate
- ClientKeyExchange: Secret PreMasterKey encrypted using Server’s public key
- ChangeCipherSpec: Client decrypts message using previously exchanged keys
- Finished: Server decrypts message using previously exchanged keys

https://rhsecurity.wordpress.com/tag/tls/
Just one problem...

• The DNS is full of liars and lies!
• And this can compromise the integrity of public key information embedded in the DNS
• Unless we fix the DNS we are no better off than before with these TLSA records!
Just one response...

• We need to allow users to validate DNS responses for themselves
• And for this we need a Secure DNS framework
• Which we have – and it’s called DNSSEC!
DANE + DNSSEC

• Query the DNS for the TLSA record of the domain name and ask for the DNSSEC signature to be included in the response
• Validate the signature to ensure that you have an unbroken signature chain to the root trust point
• At this point you can accept the TLSA record as the authentic record, and set up a TLS session based on this data
DANE + DNSSEC

- Query the DNS for the TLSA record of the domain name and ask for the DNSSEC signature to be included in the response.
- Validate the signature to ensure that you have an unbroken signature chain to the root.
- At this point you can accept the TLSA record as the authentic record, and set up a TLS session based on this data.

*Yes, but No!*
DNSSEC authenticated HTTPS in Chrome (16 Jun 2011)

Update: this has been removed from Chrome due to lack of use.

DNSSEC validation of HTTPS sites has been hanging around in Chrome for nearly a year now. But it’s now enabled by default in the current canary and dev channels of Chrome and is on schedule to go stable with Chrome 14. If you’re running a canary or dev channel (and you need today’s dev channel release: 14.0.794.0) then you can go to https://dane.imperialviolet.org and see a DNSSEC signed site in action.

DNSSEC stapled certificates (and the reason that I use that phrase will become clear in a minute) are aimed at sites that currently have, or would use, self-signed certificates and, possibly, larger organisations that are Chrome based and want certificates for internal sites without having to bother with installing a custom root CA on all the client devices. Suggesting that this heralds the end of the CA system would be utterly inaccurate. Given the deployed base of software, all non-trivial sites will continue to use CA signed certificates for decades, at least. DNSSEC signing is just a gateway drug to better transport security.
Faster validation?
Or ... Look! No DNS!

- Server packages server cert, TLSA record and the DNSSEC credential chain in a single bundle
- Client receives bundle in Server Hello
  - *Client performs validation of TLSA Resource Record using the supplied DNSEC signatures plus the local DNS Root Trust Anchor without performing any DNS queries*
  - *Client validates the presented certificate against the TLSA RR*
- Client performs Client Key exchange
Doing a better job

We could do a far better job at Internet Security by moving on from X.509 public key certificates:

- Publishing DNSSEC-signed zones
- Publishing DANE TLSA records
- Using DNSSEC-validating resolution
- Using TLSA records to guide TLS Key Exchange
- Stapling the TLSA + sig bundle into TLS
Doing a better job

We could do a far better job at Internet Security by moving on from X.509 public key certificates:

Publishing DNSSEC-signed zones
Publishing DANE TLSA records
Using DNSSEC-validating resolution
Using TLSA records to guide TLS Key Exchange
Stapling the TLSA + sig bundle into TLS

But nothing has happened for more than a decade!
Why not?
2. Looking Forward
Why is this so hard?
Why is this so hard?

We have different goals

– Some people want to provide strong hierarchical controls on the certificates and keys because it entrenches their role in providing services
– Some want to do it because it gives them a point of control to intrude into the conversations of their citizens
– Others want to exploit weaknesses in the system to leverage a competitive advantage
– Some people think users prefer faster applications, even if they have security weaknesses
– Others think users are willing to pay a time penalty for better authentication controls
Why is this so hard?

Because there are so many moving parts?

– In a system that is constructed upon the efforts of multiple systems and multiple providers we are relying on someone in charge to orchestrate the components to as working whole

Saturn V Launch Vehicle
Three stage rocket, each built by a different contractor
Each of whom used multiple subcontractors
3 million components
Each supplied by the lowest bidder!
Will it get harder?

Quantum Risk

• Public/Private key cryptography does not create “impossible” problems. It uses “hard” problems.
• But “hard depends on the speed of available CPUs and the way that CPUs calculate
• Quantum computing can potentially solve such “hard” problems, assuming that we get to the point of being able to operate quantum computers with large numbers of qbits.
• So far that challenge has proved elusive, but a lot of effort is being invested into quantum computing these days
• When it happens, we will need to turn to more complex crypto algorithms that have larger keys and require more compute power to use
Will it get more expensive?

- So far Moore’s Law has absorbed the incremental cost of crypto
- As we get to 3nm tracks on chips further reductions in size and unit cost are proving to be a major challenge
- Which implies that robust crypto may become more expensive to use
- Who is going to pay the incremental cost of highly robust crypto?
It’s a tough problem...

A rather bleak prognosis from the Economist – don’t look for technology to improve this rather disturbing situation!

They suggest looking at economics and markets to try and address this problem

The problem with this suggestion is that there is no natural market that provides incentive for highly robust and secure technologies. The major market incentives are based on driving down unit costs of service delivery, and security is an obvious point of avoidable cost
The Economics of Security

• Effective security for services and infrastructure is a market failure in the IT industry
• Consumers are unwilling to pay a major price premium for a highly robust service
• Service providers do not have any market-based incentive to add robust security to their products and offerings
• The reason why the public sector is undertaking investment in cyber defence measures is that the private sector is not naturally motivated to do so!
The Economics of Security

• Domain Name certificates have only taken off when the cost of obtaining them has dropped to zero, and the demonstration of proof of control is cursory

• And in a demonstration that Gresham’s Law applies equally well in security, the low-quality cheap certificate product has driven out other forms of extended validation certification
"The market can't fix this because neither the buyer nor the seller cares.

The owners of the low-cost devices used in distributed denial-of-service attacks don't care. Their devices were cheap to buy, they still work, and they don't know any of the victims of the attacks.

The sellers of those devices don't care: They're now selling newer and better models, and the original buyers only cared about price and features.

There is no market solution, because the insecurity is what economists call an externality: It's an effect of the purchasing decision that affects other people. Think of it as a kind of invisible pollution."

https://www.schneier.com/blog/archives/2017/02/security_and_th.html
Why is this so hard?

Because we are relying on the market to provide coherence and consistency of orchestration across providers?

– And perhaps that’s the key point here
– Loosely coupled systems will always present windows of vulnerability
  • Routing integrity
  • Name registration
  • Name certification
  • Service control
– Effective defence involves not only component defence but also in defending the points of interaction between components
– And we find this very hard to achieve when the market itself is the orchestration agent
Is this another of those massive challenges of our time?

We just don’t have the tools to figure out how to stop this environment being fatally overrun by these devices:

– We can’t improve their quality
– We can’t keep building ever larger DOS barriers
– We can’t regulate behaviours of the equipment, their makers or distributors
What a dysfunctional mess we’ve created!
3. Vague Glimmers of Hope
This is not a new problem

• We had a similar problem in the aviation industry
  – Safety was a real issue for the industry
  – The response was to shift the emphasis in investigation of incidents from blame attribution to primary cause identification

• Do we need open disclosure requirements for IT goods and services?
This is not a new problem

- Industry-based safety standards have been used in other industries
- Do consumer products and services need to comply to a set of base standards relating to operational robustness?
- In today’s world of digital goods and services how would such standards be applied?
Users and Trust

• Users just want to be able to trust that the websites and services that they connect to and share their credentials, passwords and content with are truly the ones they expected to be using without first studying for a PhD in Network Operational Security

• Somehow, we’re missing that simple objective and we’ve interposed complexity and adornment that have taken on a life of their own and are in fact eroding trust

• And that’s bad!

• If we can’t trust our communications infrastructure, then we don’t have a useful communications infrastructure.
Questions?