Why “Dane”?

Geoff Huston
Chief Scientist, APNIC
Which Bank?
Which Bank? My Bank!
Which Bank? My Bank!

I hope!
Security on the Internet

How do you know that you are really going to where you thought you were going to?
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, 5:00 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.
Security on the Internet

How do you know that you are going to where you thought you were going to?
Security on the Internet

Also, how can you keep your session a secret from wire(less) snoopers?
Opening the Connection: First Steps

Client:

*DNS Query:*
www.commbank.com.au?

*DNS Response:*
23.77.138.30

*TCP Session:*
TCP Connect 23.77.138.30, port 443
Hang on...

$ dig -x 23.77.138.30 +short

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
$ dig -x 23.77.138.30 +short

That’s an Akamai address block

And I am NOT a customer of the Internet Bank of Akamai!

So why should my browser trust that 23.77.138.30 is really the “proper” web site for the Commonwealth Bank of Australia, and not some dastardly evil scam designed to steal my passwords and my money?
The major question...

How does my browser tell the difference between an intended truth and a 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
The Power of Primes

\[(m^e)^d \equiv m \pmod{n}\]

As long as \(d\) and \(n\) are relatively large, and \(n\) is the product of two large prime numbers, then finding the value of \(d\) when you already know the values of \(e\) and \(n\) is computationally expensive.
Why is this important?

Because much of the foundation of Internet Security rests upon this prime number relationship.
Secure Connections using TLS

## Secure Connections using TLS

![TLS Connection Diagram]

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

- **ServerHello**

- **ServerHelloDone**

- **ClientKeyExchange**
  - Secret PreMasterKey encrypted using Server's public key.

- **ChangeCipherSpec**

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

- **ChangeCipherSpec**

- **Finished**
  - Client decrypts message using previously exchanged keys.

[https://rhsecurity.wordpress.com/tag/tls/](https://rhsecurity.wordpress.com/tag/tls/)
Secure Connections using TLS

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

ServerHello
Server chooses TLS version, cipher, compression method. Server sends its certificate

ServerHelloDone

ClientKeyExchange
Secret PreMasterKey encrypted using Server's public key

ChangeCipherSpec

Finished
Client decrypts message using previously exchanged keys

ChangeCipherSpec

Finished
Server decrypts message using previously exchanged keys
Secure Connections using TLS

- **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**
- **Finished**: Server decrypts message using previously exchanged keys.
- **ChangeCipherSpec**
- **Finished**: Client decrypts message using previously exchanged keys.

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

How does the client “recognise” this certificate as the “right” certificate?
### Certificate Details

**Subject Name**
- **Inc. Country**: AU
- **Business Category**: Private Organization
- **Serial Number**: 123 123 124
- **Country**: AU
- **Postal Code**: 2000
- **State/Province**: New South Wales
- **Locality**: SYDNEY
- **Street Address**: 201 SUSSEX ST
- **Organization**: Commonwealth Bank of Australia
- **Organizational Unit**: CBA Business System Hosting
- **Common Name**: www.commbank.com.au

**Issuer Name**
- **Country**: US
- **Organization**: Symantec Corporation
- **Organizational Unit**: Symantec Trust Network
- **Common Name**: Symantec Class 3 EV SSL CA - G3
- **Serial Number**: 1A 9F E9 49 03 92 E2 9A 96 10 56 69 60 3E 9B AE
- **Version**: 3
- **Signature Algorithm**: SHA-256 with RSA Encryption (1.2.840.113549.1.1.1)
- **Parameters**: none
- **Not Valid Before**: Monday, 4 May 2015 at 10:00:00 AM Australian Eastern Standard Time
- **Not Valid After**: Saturday, 27 February 2016 at 10:59:59 AM Australian Eastern Daylight Time

**Public Key Info**
- **Algorithm**: RSA Encryption (1.2.840.113549.1.1.1)
- **Parameters**: none
- **Public Key**: 256 bytes: CA 84 74 93 EB 00 22 10 ...
- **Exponent**: 65537
- **Key Size**: 2048 bits
- **Key Usage**: Encrypt, Verify, Wrap, Derive
- **Signature**: 256 bytes: 95 32 C3 F0 62 F1 F8 F1 ...
How did my browser know that this is a valid cert?
Domain Name Certification

- The Commonwealth Bank of Australia has generated a key pair
- And they passed a certificate signing request to a company called “Symantec”
- Who was willing to vouch (in a certificate) that the entity who goes by the domain name of www.commbank.com.au also has a certain public key value
- 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 Symantec and the certificates that they issue
- Symantec NEVER lie!
Domain Name Certification

• The Commonwealth Bank of Australia has generated a key pair
• And they passed a certificate signing request to a company called “Symantec”
• Who was willing to vouch (in a certificate) that the entity who goes by the domain name of www.commbank.com.au also has a certain public key value
• 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 Symantec and the certificates that they issue
• Symantec NEVER lie!

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 or Local Credulity*

That’s a big list of people to Trust

Are they all trustable?
Local Credulity

That’s a big list of people to Trust

Are they all trustable?

Evidently Not!

Maintaining digital certificate security

Posted: Monday, March 23, 2015

Posted by Adam Langley, Security Engineer

On Friday, March 20th, we became aware of unauthorized digital certificates for several Google domains. The certificates were issued by an intermediate certificate authority apparently held by a company called MCS Holdings. This intermediate certificate was issued by CNNIC.

CNNIC is included in all major root stores and so the misissued certificates would be trusted by almost all browsers and operating systems. Chrome on Windows, OS X, and Linux, ChromeOS, and Firefox 33 and greater would have rejected these certificates because of public-key pinning, although misissued certificates for other sites likely exist.

We promptly alerted CNNIC and other major browsers about the incident, and we blocked the MCS Holdings certificate in Chrome with a CRLSet push. CNNIC responded on the 22nd to explain that they had contracted with MCS Holdings on the basis that MCS would only issue certificates for domains that they had registered. However, rather than keep the private key in a suitable HSM, MCS installed it in a man-in-the-middle proxy. These devices intercept secure connections by masquerading as the intended destination and are sometimes used by companies to intercept their employees’ secure traffic for monitoring or legal reasons. The employees’ computers normally have to be configured to trust a proxy for it to be able to do this. However, in this case, the presumed proxy was given the full authority of a public CA, which is a serious breach of the CA system. This situation is similar to a failure by ANSSI in 2013.
Local Credulity

That’s a big list of people to Trust

Are they all trustable?

Evidently Not!
But my bank used Symantec

as their Certificate Authority

And Symantec NEVER lie in the certificates they issue
Never?
Well, hardly ever

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.
With unpleasant consequences when it all goes wrong
With unpleasant consequences when it all goes wrong

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 have led to as many as 300,000 people being spied on this summer.

Comodohacker, as he calls himself, insists that he acted on his own and is unperturbed by the notion that his work might have been used to spy on antigovernment compatriots.

“I’m totally independent,” he said in an e-mail exchange with The New York Times. “I just share my findings with some people in Iran. They are free to do anything they want with my findings and things I share with them, but I’m not responsible.”

In this moment of reckoning, this is most likely to happen if the hacking community is divided.

International Herald Tribune
Sep 13, 2011 Front Page
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 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!

Here’s a lock – it might be the lock on your front door for all I know.

The lock might LOOK secure, but don’t worry – literally ANY key can open it!
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 you 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
Than what ‘wins’ in the market?

Sustainable
Trusted
Resilient
Privacy
Secure

?
In a commercial environment

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

Sustainable
Resilient
Secure
Privacy
Trusted
Cheap!
Where now?

Option A: Take all the money out of the system!
Where now?

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

Will the automation of the Cert issuance coupled with a totally free service make the overall environment more or less secure? We’re probably going to find out real soon!
Where now?

Option B: White Listing and Pinning with HSTS

https://code.google.com/p/chromium/codesearch#chromium/src/net/http/transport_security_state_static.json
Where now?

Option B: White Listing and Pinning with HSTS

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

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

Its just Google protecting itself and no one else
Where now?

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.

INFOWORLD TECH WATCH
By Fahmida Y. Rashid, Senior Writer, InfoWorld | JAN 30, 2017

Google moves into the Certificate Authority business

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

```c
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.
23 //
```
Where now?

Option C: Use the DNS!
Where now?

Option C: Use the DNS!

We believe in rough consensus and running code.

Just put it in the DNS.

www.cafepress.com/nxdomain
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?
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 (pinning record)?
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?
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?
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?
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?
DANE

• Using the DNS to associated domain name public key certificates with domain name

RFC 6698 -- You should read this!
DANE

• Using the DNS to associate domain name public key certificates with domain name

You probably should read RFC 7671 as well!
2.3. TLSA RR Examples

An example of a hashed (SHA-256) association of a PKIX CA certificate:

```
_443._tcp.www.example.com. IN TLSA (0 0 1 d2abde240d7cd3ee6b4b28c54df034b9
  7983a1d16e8a410e4561cb106618e971 )
```

An example of a hashed (SHA-512) subject public key association of a PKIX end entity certificate:

```
_443._tcp.www.example.com. IN TLSA
  1 1 2 92003ba34942dc74152e2f2c408d29ec
  a5a520e7f2e06bb944f4dca346baf63c
  1b177615d466f6c4b71c216a50292bd5
  8c9ebd2f74e38fe51ffd48c43326cbc )
```

An example of a full certificate association of a PKIX trust anchor:

```
_443._tcp.www.example.com. IN TLSA
  2 0 0 30820307308201efa003020102020... )
```
EECert TLSA record generation

; Convert the public key certificate to DER format
; Generate the SHA256 hash
; Add DNS gunk!

$ /usr/bin/openssl x509 -in /usr/local/etc/letsencrypt/live/www.dotnxdomain.net/cert.pem -outform DER | /usr/bin/openssl sha256 |
cut -d ' ' -f 2 |
awk '{print "_443._tcp.www.dotnxdomain.net IN TLSA 3 0 1 "$1}" |

_443._tcp.www.dotnxdomain.net. 899 IN TLSA 3 0 1 D42101BCCE941D22E8E467C5D75E77EC4A7B8B7C9366C6A878CB4E15 7E602F17

$ dig +dnssec TLSA _443._tcp.www.dotnxdomain.net.

_443._tcp.www.dotnxdomain.net. 899 IN TLSA 3 0 1 D42101BCCE941D22E8E467C5D75E77EC4A7B8B7C9366C6A878CB4E15 7E602F17

_443._tcp.www.dotnxdomain.net. 899 IN RRSIG TLSA 13 5 900 20200724235900 20170122043100 56797 www.dotnxdomain.net.
dUYD1sMIpBc6RsUhturFzz5G8qX6oaDGRzaD/q6n+YJi2kqzDfWZls6F 3X1mXdpeQQYz52yOU0cdWvFR09TQZQ==
SPKI TLSA record generation

; Generate the public key
; Convert it to DER format
; Generate the SHA256 hash
; Add DNS gunk!

```
$ /usr/bin/openssl x509 -in /usr/local/etc/letsencrypt/live/www.dotnxdomain.net/cert.pem -pubkey -noout | openssl rsa -pubin -outform der | /usr/bin/openssl sha256 | cut -d ' ' -f 2 | awk '{ print "_443._tcp.www.ndotnxdomain.net IN TLSA 3 1 1 " $1}''

_443._tcp.www.ndotnxdomain.net IN TLSA 3 1 1 df3a810d998cfddf8fa935ed33065ee27a67747366e2da40ddefef2b3a2032eb
```
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

TLS Client
- ClientHello
- ServerHello
- ServerHelloDone
- ClientKeyExchange
- ChangeCipherSpec
- Finished

DNS Name

TLSA query

Public Key Cert

TLS Server
- Client decrypts message using previously exchanged keys
- ChangeCipherSpec
- 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 its called **DNSSEC**!
DNSSEC Interlocking Signatures

. (root)
  . Key-Signing Key – signs over
  . Zone-Signing Key – signs over
    DS for .com (Key-Signing Key)

.com
  .com Key-Signing Key – signs over
  .com Zone-Signing Key – signs over
    DS for example.com (Key-Signing Key)

.example.com
  example.com Key-Signing Key – signs over
    example.com Zone-Signing Key – signs over
      www.example.com

www.example.com
DNSSEC Interlocking Signatures

. (root)
  . Key-Signing Key – signs over
    . Zone-Signing Key – signs over
      DS for .com (Key-Signing Key)

.com
  .com Key-Signing Key – signs over
    .com Zone-Signing Key – signs over
      DS for example .com (Key-Signing Key)

.example.com
  example.com Key-Signing Key – signs over
    example.com Zone-Signing Key – signs over
      www.example.com

www.example.com IN A 192.0.1
DNSSEC Interlocking Signatures

- (root)
  - Key-Signing Key – signs over
    - Zone-Signing Key – signs over
      DS for .com (Key-Signing Key)

- .com
  - Key-Signing Key – signs over
    - Zone-Signing Key – signs over
      DS for example .com (Key-Signing Key)

- example.com
  - Key-Signing Key – signs over
    - Zone-Signing Key – signs over
      www.example.com

- www.example.com
  - IN A 192.0.1
DNSSEC Interlocking Signatures

As long as you have a valid local trust anchor for the root zone then you can validate a signed DNS response by constructing this backward path to the local root trust anchor.
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 Does DNS via a Browser Extension

DNSSEC/TLSA Validator is a web browser add-on which allows you to check the existence and validity of DNS Security Extensions (DNSSEC) records and Transport Layer Security Association (TLSA) records related to domain names. Results of these checks are displayed by using icons and information texts in the page's address-bar or browser tool-bar. Currently, Internet Explorer (IE), Mozilla Firefox (MF), Google Chrome/Chromium (GC), Opera (OP), Apple Safari (AS) are supported.

Version: 2.2.0
Thu 04 September 2014

New Features:
- New js-ctypes-based implementation for Firefox.
- New validator implementation for Chromium/Chrome/Opera based on Native Messaging.
So we need DNSSEC as well as DANE...

How much DNSSEC Validation is out there?
Do we do DNSSEC Validation?

Use of DNSSEC Validation for World (XA)

(stats.labs.apnic.net/dnssec/XA)
Where do we do DNSSEC Validation?

stats.labs.apnic.net/dnssec/XA
Where now?

Browser vendors appear to be dragging the chain on DANE support

DANE exists today as plug-ins rather than a core functionality

Cynically, one could observe that fast but insecure is the browser vendors’ current preference!
Where now?

Browser vendors appear to be dragging the chain on DANE support

DANE exists today as plug-ins rather than a core functionality

Cynically, one could observe that fast but insecure is the browser vendors’ current preference!
Or...

• We could change the DNS to allow TLS to make efficient use of DANE
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
Where now?

We could do a far better job at Internet Security:
- Publishing DNSSEC-signed zones
- Publishing DANE TLSA records
- Using DNSSEC-validating resolution
- Using TLSA records to guide TLS Key Exchange

What this can offer is robust, affordable, accessible security without the current overheads of high priced vanity CA offerings
Let’s Do it!

What Let’s Encrypt and DNSSEC offers is robust, affordable, accessible security without the current overheads of high priced vanity CA offerings.
That's it!

Questions?