ROMA
MMXVI

APPSEC
EUROPE

Everything You Need to Know About Certificate Pinning

But Are Too Afraid To Ask

$ whoami

- John Kozyrakis

  Technical strategist @ Cigital @ London, UK

  - Mobile security, secure architecture
  - Android static analysis, automation

@ikoz

https://koz.io

john.kozyrakis@owasp.org

Agenda

- Trust
- Pinning fundamentals
- Decision points
- Common mistakes
- Advanced topics

ROMA
MMXVI

APPSEC
EUROPE

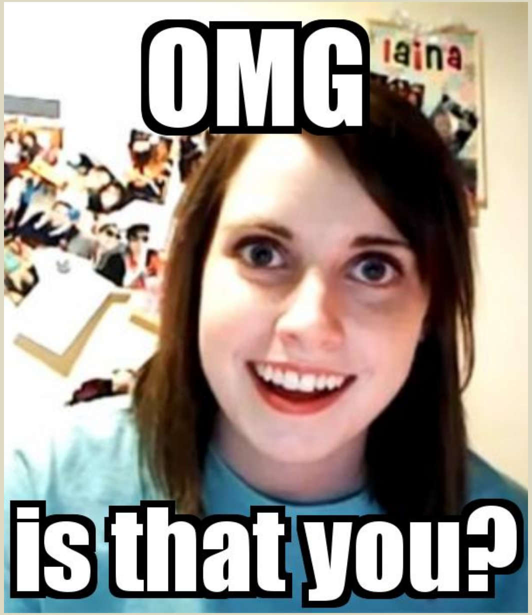
CERTIFICATE PINNING

SSL, CERTIFICATES & TRUST

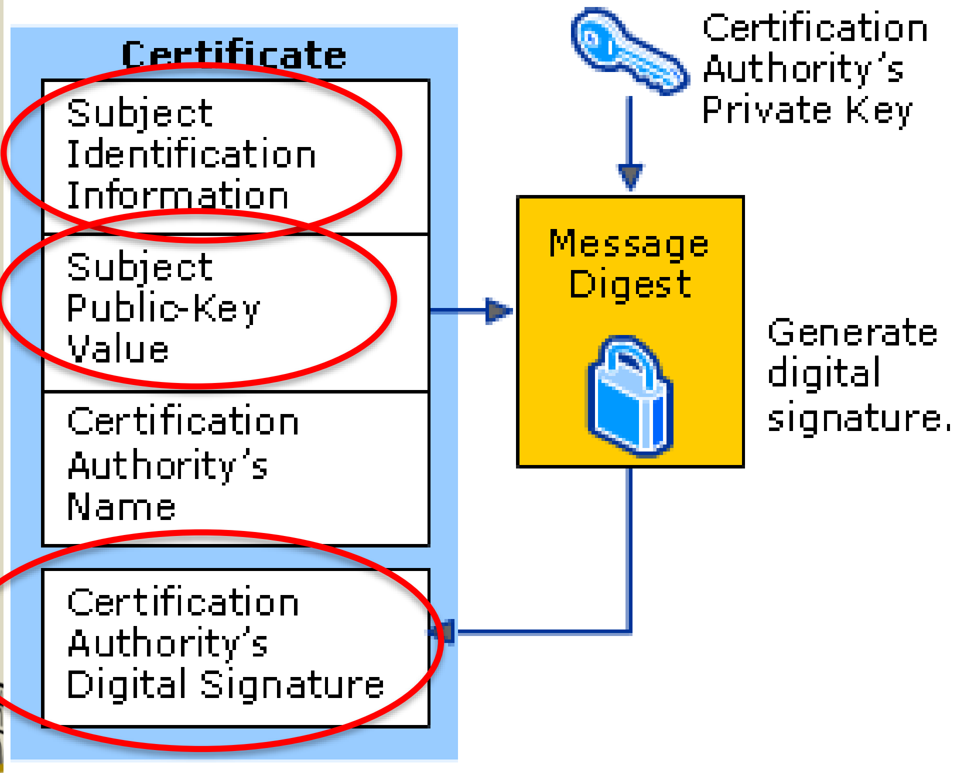
Trust problem

- How can a client trust a server?

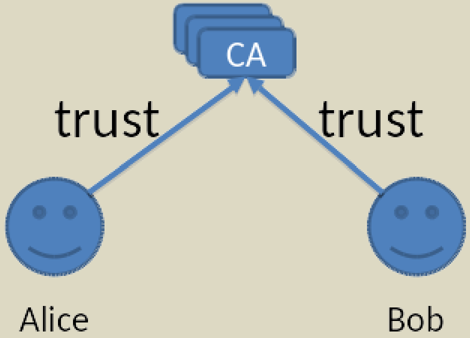
- Bind identity to Public Key
    1. CAs & X.509 chains
    2. Pinning
    3. Hybrid



X.509 & root CAs

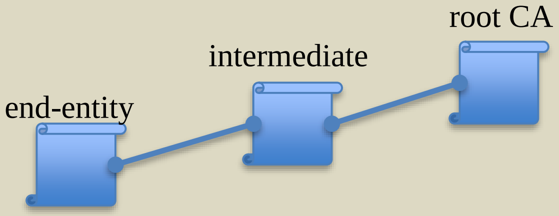

Certificate
Subject Identification Information
Subject Public-Key Value
Certification Authority's Name
Certification Authority's Digital Signature
Certification Authority's Private Key
Message Digest
Generate digital signature.

"Trusted anchors"
OS cert store



Trust evaluation

- Recursive X.509 certificate chain validation
  - Client assembles chain
    - from received end-entity cert to a **trusted anchor**
  - Checks validity of all certs (dates, constraints, signatures..)
- TLS-specific checks
  - Hostname Verification



Most mobile apps know their server

- OS anchor store solves the ‘unknown server’ problem
- But this problem does not exist for *most* mobile apps
- *a-priori* knowledge

CERTIFICATE PINNING

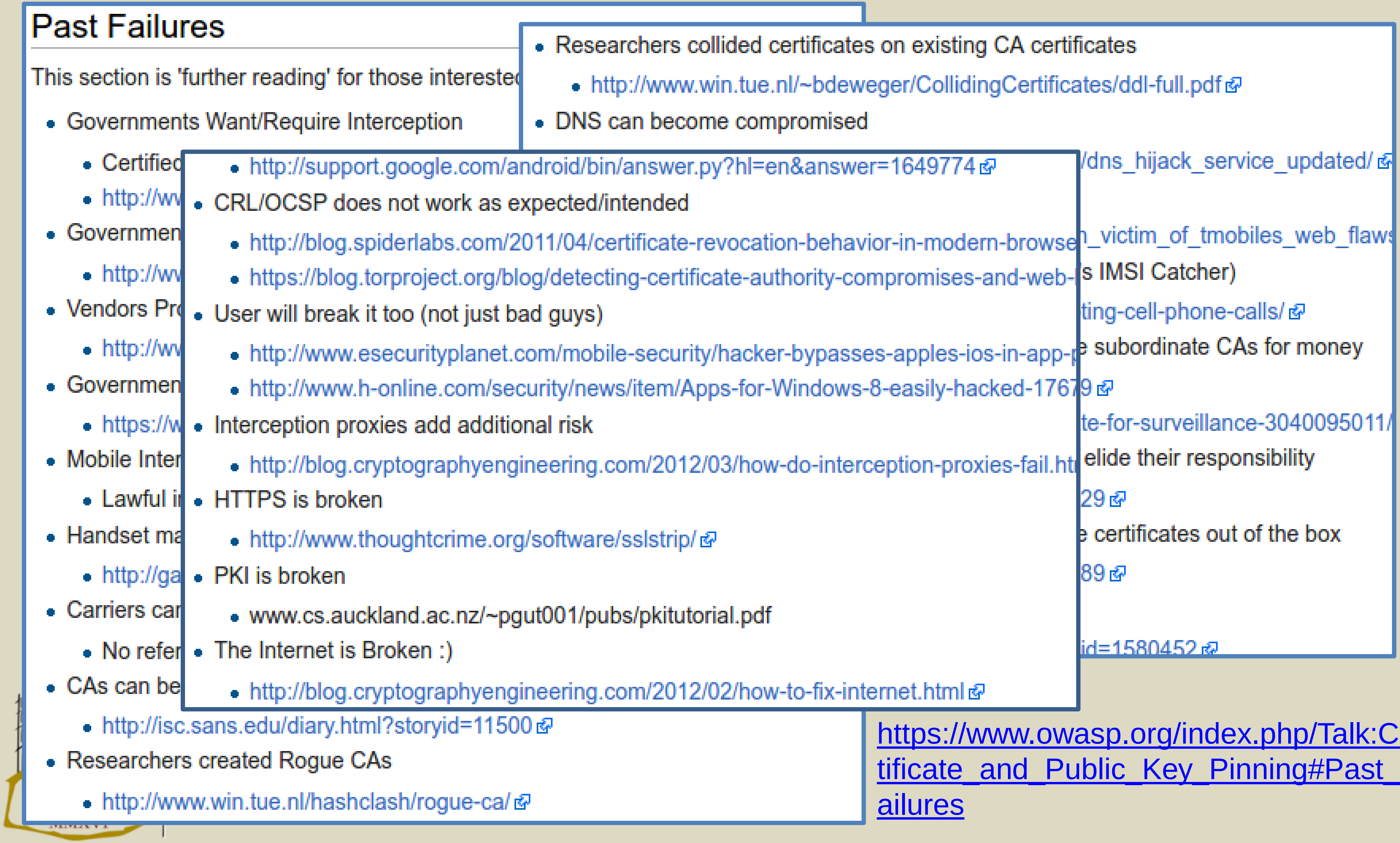
FUNDAMENTALS

What is Pinning?

- Goal: To associate an identity with a public key
  - Association process owned by developers, not CAs

Benefits over normal TLS validation

- Protection against *certificate forgery*
  - Rogue CAs
  - Compromised CAs
  - Users phished into inserting certs to device trust store



Past Failures

This section is 'further reading' for those intereste

- Governments Want/Require Interception
  - Certified
  - http://ww
- Governmen
  - http://ww
- Vendors Pr
  - http://ww
- Governmen
  - https://w
- Mobile Inter
  - Lawful i
- Handset ma
  - http://ga
- Carriers car
  - No refer
- CAs can be
  - http://isc.sans.edu/diary.html?storyid=11500
- Researchers created Rogue CAs
  - http://www.win.tue.nl/hashclash/rogue-ca/

- Researchers collided certificates on existing CA certificates
  - http://www.win.tue.nl/~bdeweger/CollidingCertificates/ddl-full.pdf
- DNS can become compromised

/dns_hijack_service_updated/

n_victim_of_tmobiles_web_flaws

s IMSI Catcher)

ting-cell-phone-calls/

e subordinate CAs for money

te-for-surveillance-3040095011/

elide their responsibility

29

e certificates out of the box

89

id=1580452

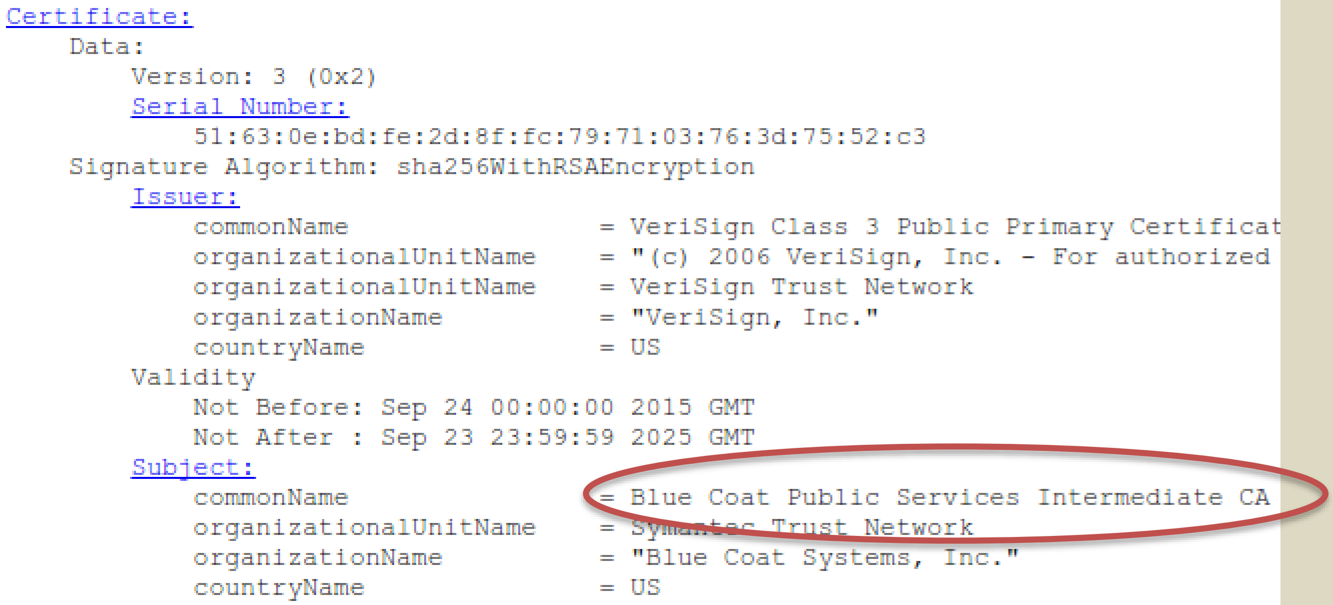
- http://support.google.com/android/bin/answer.py?hl=en&answer=1649774
- CRL/OCSP does not work as expected/intended
  - http://blog.spiderlabs.com/2011/04/certificate-revocation-behavior-in-modern-browse
  - https://blog.torproject.org/blog/detecting-certificate-authority-compromises-and-web-
- User will break it too (not just bad guys)
  - http://www.esecurityplanet.com/mobile-security/hacker-bypasses-apples-ios-in-app-p
  - http://www.h-online.com/security/news/item/Apps-for-Windows-8-easily-hacked-17679
- Interception proxies add additional risk
  - http://blog.cryptographyengineering.com/2012/03/how-do-interception-proxies-fail.ht
- HTTPS is broken
  - http://www.thoughtcrime.org/software/sslstrip/
- PKI is broken
  - www.cs.auckland.ac.nz/~pgut001/pubs/pkitutorial.pdf
- The Internet is Broken :)
  - http://blog.cryptographyengineering.com/2012/02/how-to-fix-internet.html

https://www.owasp.org/index.php/Talk:Certificate_and_Public_Key_Pinning#Past_Failures

Benefits over normal SSL validation

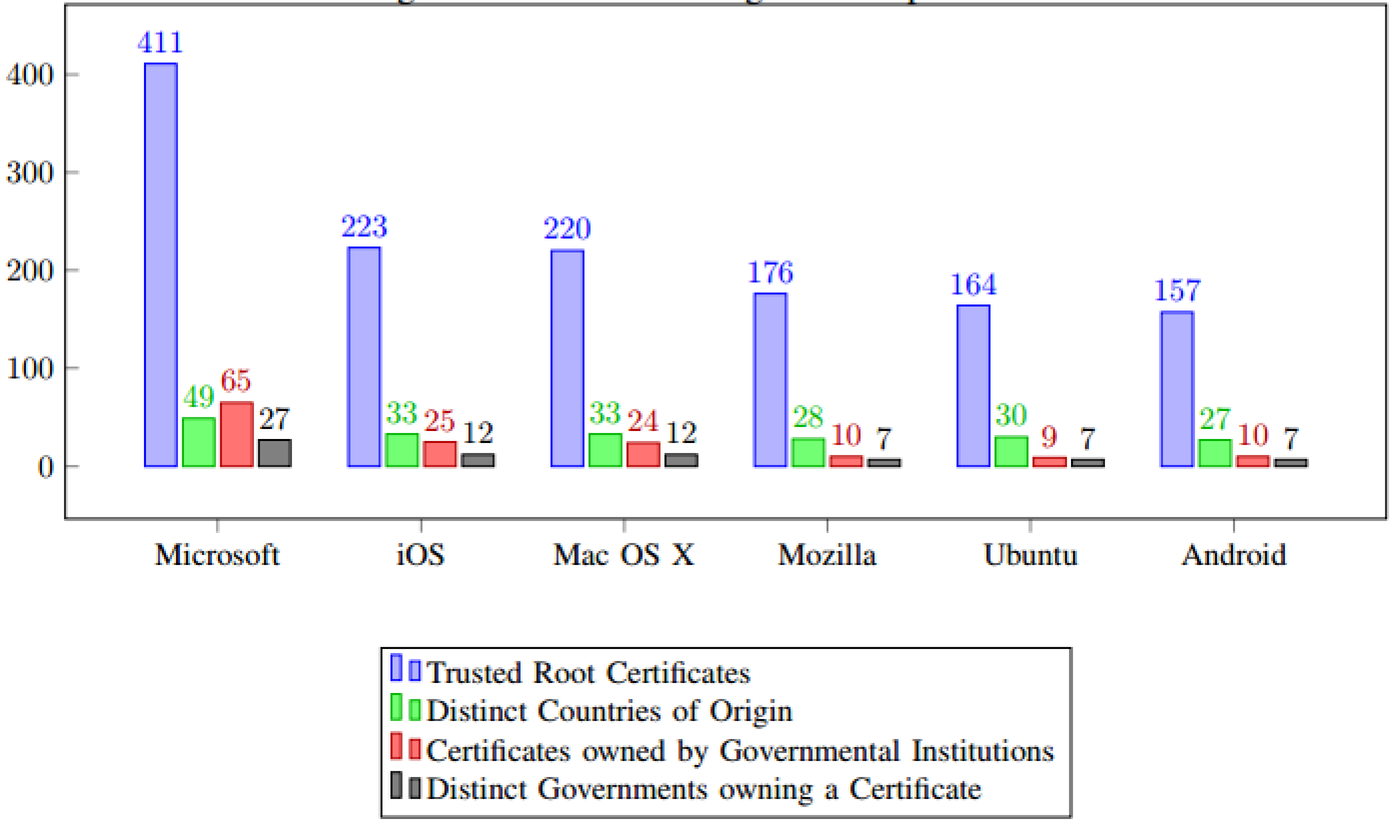
- Protection against *certificate forgery*
  - Rogue CAs
  - Compromised CAs
  - Users phished into inserting certs to device trust store
- Reduction of attack surface

Trusted authorities?



```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            51:63:0e:bd:fe:2d:8f:fc:79:71:03:76:3d:75:52:c3
    Signature Algorithm: sha256WithRSAEncryption
        Issuer:
            commonName                = VeriSign Class 3 Public Primary Certificat
            organizationalUnitName    = "(c) 2006 VeriSign, Inc. - For authorized
            organizationalUnitName    = VeriSign Trust Network
            organizationName          = "VeriSign, Inc."
            countryName               = US
        Validity
            Not Before: Sep 24 00:00:00 2015 GMT
            Not After : Sep 23 23:59:59 2025 GMT
        Subject:
            commonName                = Blue Coat Public Services Intermediate CA
            organizationalUnitName    = Symantec Trust Network
            organizationName          = "Blue Coat Systems, Inc."
            countryName               = US
```

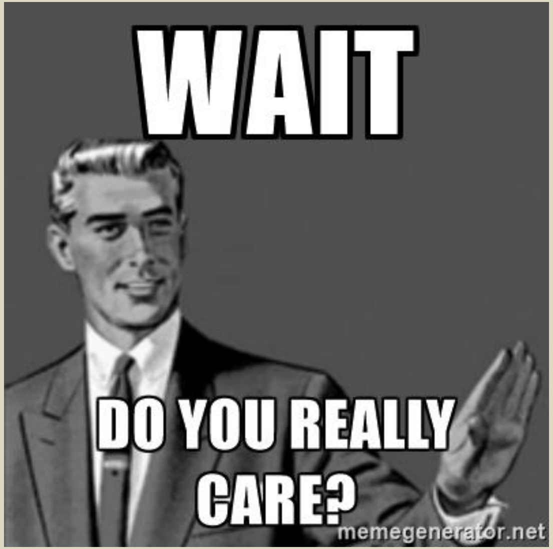
Figure 2. Root CA Program Comparison



Trust me, I'm a Root CA! Analyzing SSL Root CAs in Modern Browsers and Operating Systems. (ARES '15)
Tariq Fadai, Sebastian Schrittwieser, Peter Kieseberg, and Martin Mulazzani. 2015.

Before pinning

- Concerned about *maliciously issued certificates*?
  - Yes
    - Pinning!
  - Maybe
    - Defense in depth
  - Not really
    - *Not worth the effort* for *most*



The downside

- Will not secure connections if pinned host compromised

- Will create a single point of failure
- Will cause operational headaches
- Will require maturity/coordination
- May impact performance

Not for local attacks

- Will not stop users intercepting own traffic
- Will not stop reverse engineers & local bypass
- Will not help if device is rooted/jailbroken
- If this is a goal…
  - Use message-level asymmetric encryption
  - Binary hardening, obfuscation, move to native
  - Client-side controls: you can’t win, but can raise the bar

“Absence of Certificate Pinning”

- *Not* a security vulnerability
- May be a good practice *for some*

- “Broken pinning implementation” *IS* a security vulnerability

CERTIFICATE PINNING

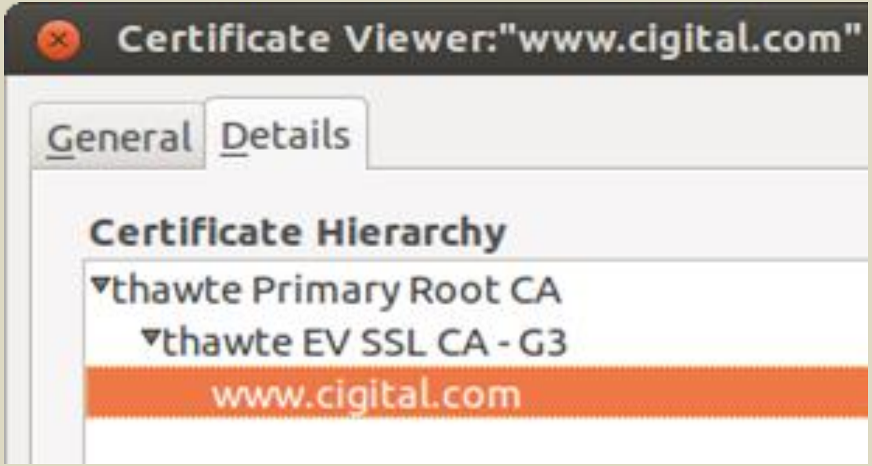
DECISION POINTS

Decisions, decisions

1. **Which identity to pin to?**
2. Pin to full cert or public key?
3. How to handle compromise?
4. How to handle rotation?
5. How to handle pin failures?
6. How to deploy the pins?

Pinning to end-entity identity

- Tiny attack surface
- No 3rd parties involved
- Easily self-signed
- No need for chain validation



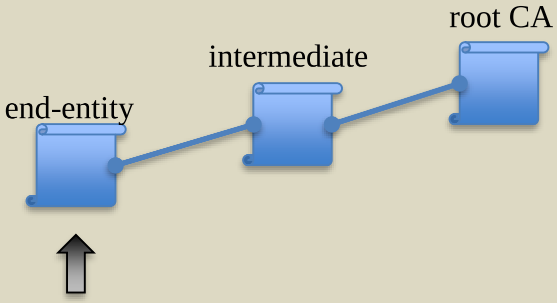
Certificate Viewer:"www.cigital.com"

General Details

Certificate Hierarchy

▾thawte Primary Root CA
 ▾thawte EV SSL CA - G3
  www.cigital.com

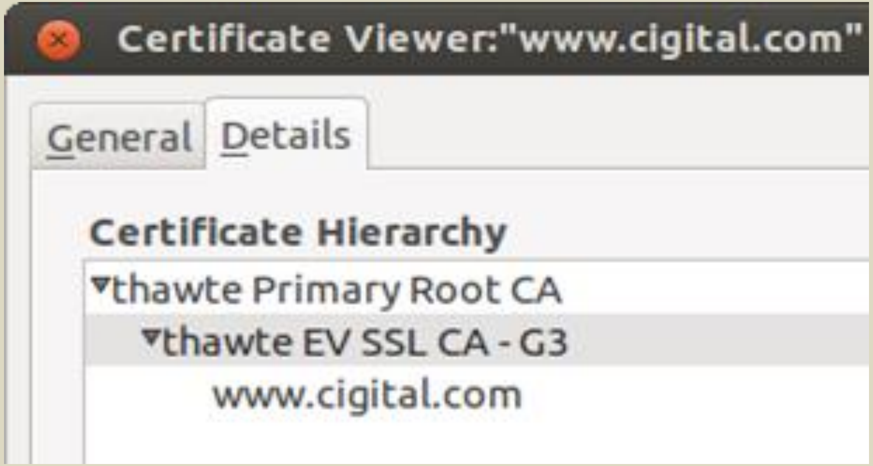
- Highly fragile
- Requires maturity



Pinning to intermediate CA identity

- More flexible

- Chain validation bugs
- Not easily self-signed
- ICA may change
- No guarantees pinned ICA is used

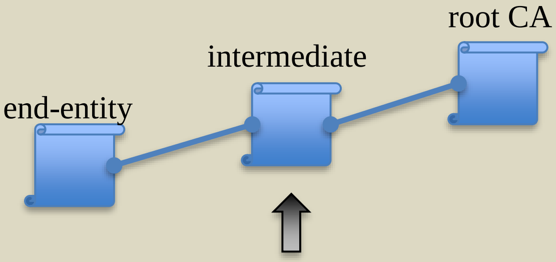


Certificate Viewer:"www.cigital.com"

General Details

Certificate Hierarchy

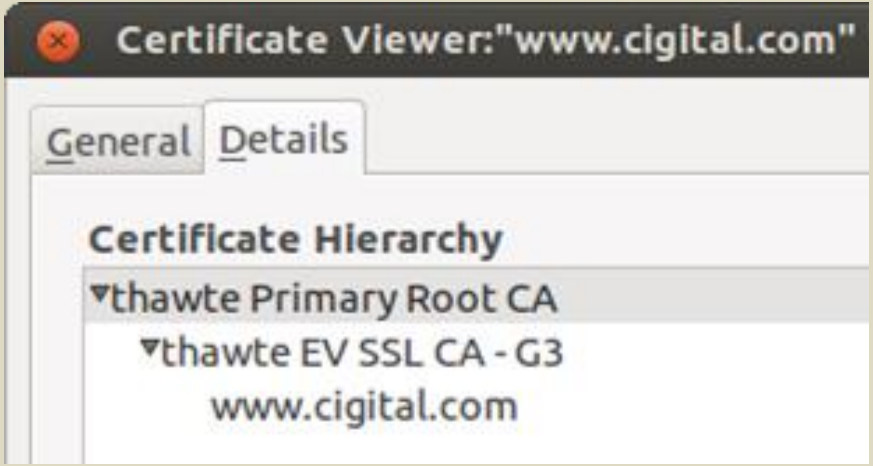
▾thawte Primary Root CA
    ▾thawte EV SSL CA - G3
        www.cigital.com



Pinning to root CA identity

- Most flexible

- Very wide attack surface
- Chain validation bugs
- Avoid cross-certified roots

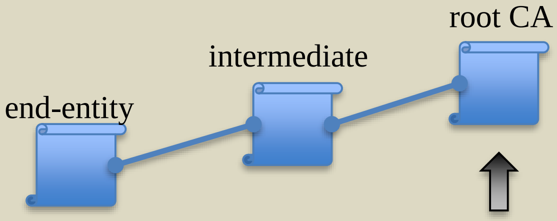


Certificate Viewer:"www.cigital.com"

General Details

Certificate Hierarchy

▾thawte Primary Root CA
  ▾thawte EV SSL CA - G3
    www.cigital.com



Pinning to internal CA identity

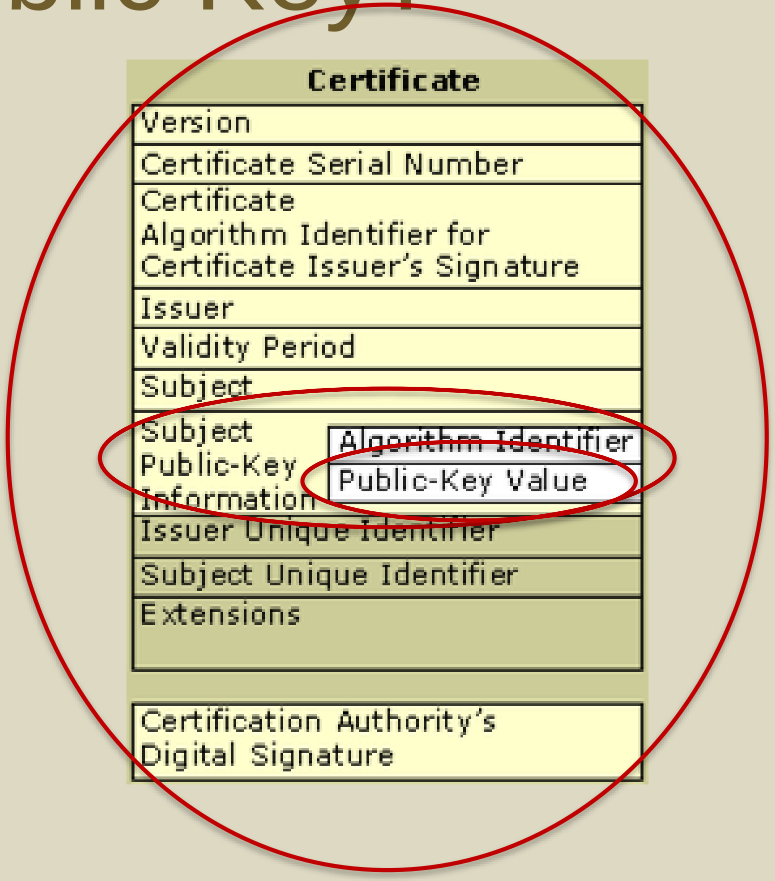
- Secure and flexible
- Possible compliance issues
- Insecure access for non-pinning clients
- Chain validation bugs
- Impossible with some pinning implementations
- Requires operational maturity

Decisions, decisions

1. Which identity to pin to?
2. **Pin to full cert or public key?**
3. How to handle compromise?
4. How to handle rotation?
5. How to handle pin failures?
6. How to deploy the pins?

Certificate or Public Key?

- Full certificate
- Public key
- SPKI



Certificate

Version

Certificate Serial Number

Certificate Algorithm Identifier for Certificate Issuer's Signature

Issuer

Validity Period

Subject

Subject Public-Key Information

Algorithm Identifier

Public-Key Value

Issuer Unique Identifier

Subject Unique Identifier

Extensions

Certification Authority's Digital Signature

Full certificate as pin

- Commonly used
- Easy pin creation
- Only option for some pinning implementations
- Only option for internal CA pinning
- Brittle
  - CA certificates often reissued/rotated
  - CAs may use multiple certs

Public key / SPKI as pin

- Trickier to get pins
- Flexible: allows key continuity
- Anonymized: pin hashes
- Several open source libraries require it
- Can’t pin to internal self-signed CA
    - Depends on system’s trust anchors

Decisions, decisions

1. Which identity to pin to?
2. Pin to full cert or public key?
3. **How to handle compromise?**
4. How to handle rotation?
5. How to handle pin failures?
6. How to deploy the pins?

How to handle compromise?

- Security != Usability
- Revocation? ☹
- Create action plan
- Fallback certs
  - Maintain an extra cert for each host off-line
  - Include fallback pin in app
- Enforce app updates for all users
  - Limit available functionality for older apps

ROMA MMXVI | APPSEC EUROPE

Decisions, decisions

1. Which identity to pin to?
2. Pin to full cert or public key?
3. How to handle compromise?
4. **How to handle rotation?**
5. How to handle pin failures?
6. How to deploy the pins?

How to handle rotation?

- Keep track of your app’s end points & pins
- Create cert rotation schedule
  - Issue new certs long before rotation
  - Do scheduled app updates
  - Review pins as part of update process
- Coordinate between PKI/servers/mobile teams
- Practice key continuity
  - Rotate certificate, not public key

Decisions, decisions

1. Which identity to pin to?
2. Pin to full cert or public key?
3. How to handle compromise?
4. How to handle rotation?
5. **How to handle pin failures?**
6. How to deploy the pins?

How to handle pin failures?

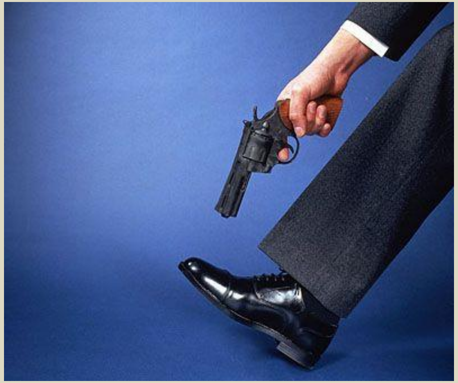
- Hard-fail: Do not establish the channel
  - Common, easy, secure
  - Inflexible, user experience issues, danger of self-induced DoS
- Soft-fail: retry without pinning
  - Tricky to get right, custom
  - Limit app functionality – lower trust mode
  - “report mode”

Decisions, decisions

1. Which identity to pin to?
2. Pin to full cert or public key?
3. How to handle compromise?
4. How to handle rotation?
5. How to handle pin failures?
6. **How to deploy the pins?**

Pin deployment: preloading

- App ships with hardcoded pin list
  - Common
  - Easy to implement
  - Complex to operate
    - Maintain version/pin map, force updates
  - Requires app updates
    - To revoke/rotate pins
  - Insecurity window
  - Self-induced DoS



Pin deployment: Trust On First Use

- Preferred if no *a-priori* knowledge of endpoints
- Easy to roll out
- Fairly complex to design
- Pin expiration - attack window
- Good for not-so-critical or unknown endpoints
  - WebView traffic
- Future? HPKP – RFC7469

Pin deployment: Over The Air

- Very flexible
- Easy to deploy
- Easy to get wrong
  - Complexity, custom protocol, expirations
- Still have to pin the ‘pin server’
- Still have to manage the pins

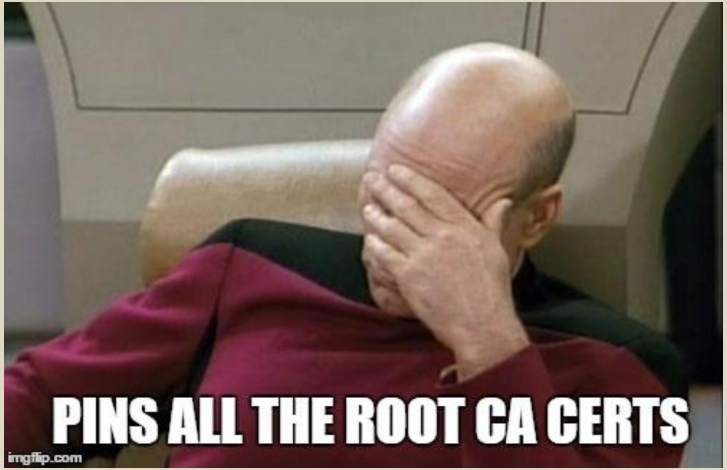
BUGS, FLAWS AND BAD DESIGNS

Avoid chain validation

- Never roll your own X.509 chain validation
- Use the system’s TLS validation routines
  - Or a 3rd party library like OpenSSL
- Using the system’s trust anchors is optional
- If pinning to CA cert
  - chain validation AND hostname verification

Don’t pin all the things

- Pinning to the 20 most popular root CAs
- Attack surface reduction?
- Worth the trouble?



Limit attack surface per host

- Want to pin connections to 10 domains?
- Host-to-pin mapping

Don’t forget half connections

- Apps may use multiple *connection handlers*
  - But only one might use of pinning
    - Seen app with 4 different networking stacks, 3 different pinning implementations, 1 broken, 1 without pins
- Pin ALL connections to pinned hosts
  - Centralise connection handling through app via library
- Try to take control of ALL connections in your app

Avoid TOCTOU bugs

- Skip pin validation if the host passed validation once?
- Secure only if SSL resumption / caching used
  - It most likely isn’t
- Pin validation should be done for every request to pinned hosts

Be careful if caching

- Skip pin validation if cert in cache?
- Insecure if you cache CA certs
  - Chain validation bypass
  - May even bypass hostname verification

Some Java APIs are dangerous

- Always check pins on validated chain
- CVE-2016-2402 (okhttp ++)

```
X509TrustManager.checkServerTrusted()
javax.net.ssl.SSLSession.getPeerCertificates()
javax.net.ssl.SSLSession.getPeerCertificateChain()
```

https://www.cigital.com/blog/ineffective-certificate-pinning-implementations/

ROMA
MMXVI

APPSEC
EUROPE

ADVANCED TOPICS

Implementation taxonomy

- pin-no-eval
  - Pure end-entity pinning: No X.509/TLS evaluation
- eval-then-pin
  1. X.509 chain evaluation by system using system’s trust anchors
  2. Check if pins inside the *validated chain*
- pin-then-eval
  - X.509 chain evaluation by system using your own trust anchors
- pin-then-custom-eval
  - X.509 chain evaluation by app using own trust anchors

Handling connections

- Invoke handler API for each pinned connection
  - Create custom “pinned” API in app
  - Use a pinning networking library
    - okhttp and others
- Automatically direct most* connections to your API
  - iOS: `NSURLprotocol` swizzling
  - Android: `URL.setURLStreamHandlerFactory()`

* excludes webviews, non-httsurlconnection…

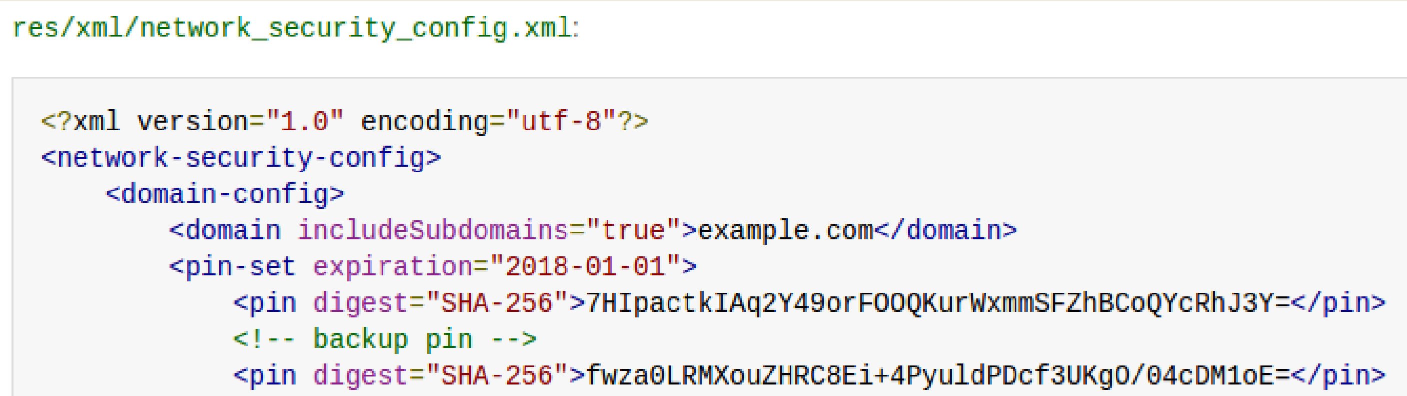
Cert pinning implementation

- Android:
  - Careful: `X509TrustManager.checkServerTrusted()`
  - API 17+: `X509TrustManagerExtensions.checkServerTrusted()`
  - API 24+: Custom `X509ExtendedTrustManager`
- iOS:
  - custom `NSURLConnectionDelegate: SecTrustEvaluate()`
- System’s OpenSSL library
  - Don’t. Not great benefit, also restricted in Android API 24+
- Other libraries: okHttp, TrustKit, AndroidPinning….
- Statically compile OpenSSL (or other)
  - Much more resistant to local attacks but tricky to get right

Pinning & WebViews

- WebViews: used to render web pages in app
  1. Connection handler -> no native pinning support
  2. Rendering engine
- Android:
  - Intercept requests using `shouldInterceptRequest()`
  - load using own handler, feed response back to WebView
- iOS
  - Intercept connections using `NSURLprotocol:startLoading()`
  - load using own handler, feed response back to protocol
  - Pinning & WKWebView = only on iOS9 (`didReceiveAuthenticationChallenge`)

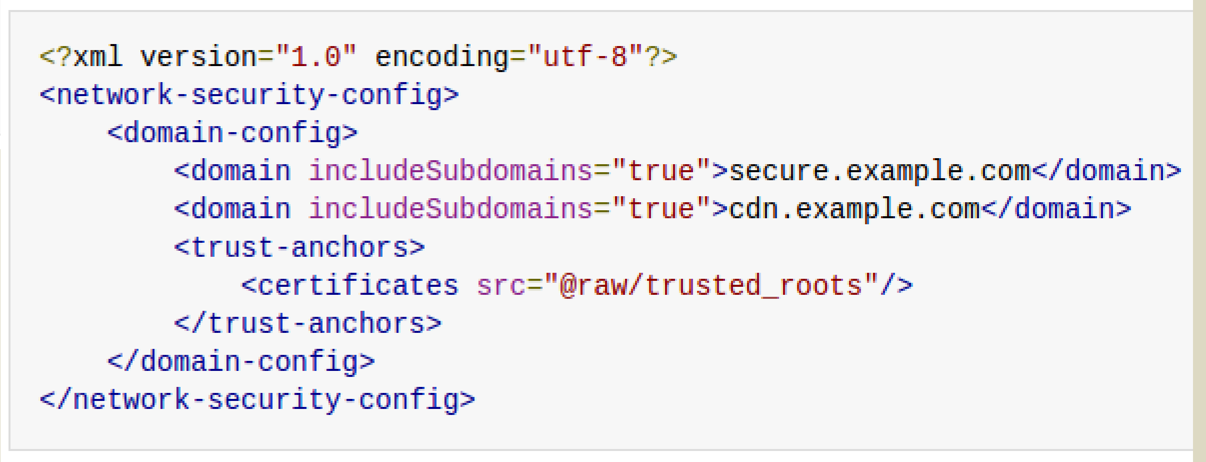
Android Network Security Config



res/xml/network_security_config.xml:

```xml
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config>
        <domain includeSubdomains="true">example.com</domain>
        <pin-set expiration="2018-01-01">
            <pin digest="SHA-256">7HIpactkIAq2Y49orFOOQKurWxmmSFZhBCoQYcRhJ3Y=</pin>
            <!-- backup pin -->
            <pin digest="SHA-256">fwza0LRMXouZHRC8Ei+4PyuldPDcf3UKgO/04cDM1oE=</pin>
        </pin-set>
    </domain-config>
</network-security-config>
```

Android Nougat



```xml
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config>
        <domain includeSubdomains="true">secure.example.com</domain>
        <domain includeSubdomains="true">cdn.example.com</domain>
        <trust-anchors>
            <certificates src="@raw/trusted_roots"/>
        </trust-anchors>
    </domain-config>
</network-security-config>
```

Summary

- Pinning is a headache
- Best: pin to end-entity
- Second best: pin to internal CA
- Preload pins for most sensitive connections
- Never validate X.509 chains manually
- Get your implementation reviewed

Questions?

ONE DOES NOT SIMPLY
IMPLEMENT CERTIFICATE PINNING
imgflip.com