SYNOPSYS®
Silicon to Software™

Pinning

Not as simple as it sounds

John Kozyrakis

Android Security Symposium – Vienna – 9 March 2017

$ whoami

- John Kozyrakis
  - London, UK
  - Technical strategist
    - Synopsys Software Integrity Group
    - Cigital
  - Mobile application security
    - Secure Design
    - Static Analysis
    - Binary Protection
    - Security Research

Agenda

Trust and PKIX

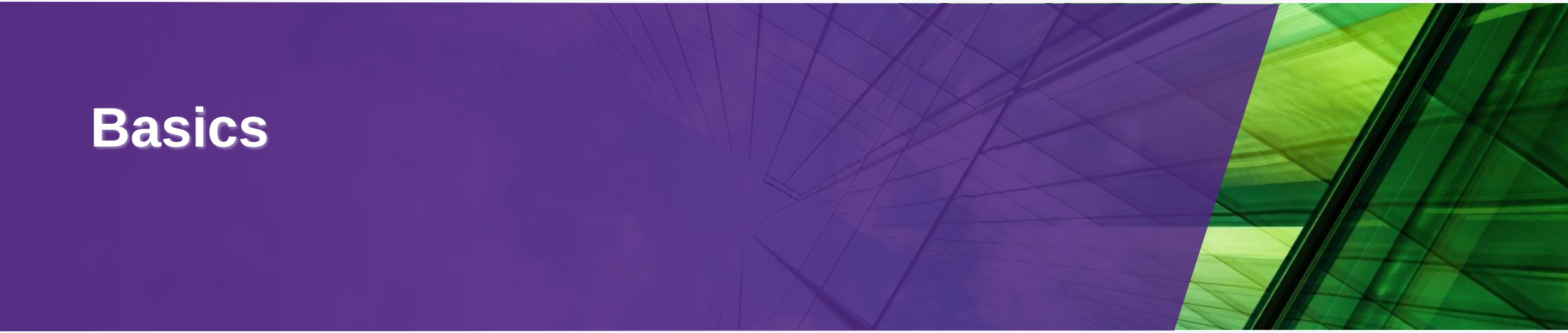
Pinning fundamentals

Common implementation mistakes

New platform support

Questions

Basics



Protecting communications

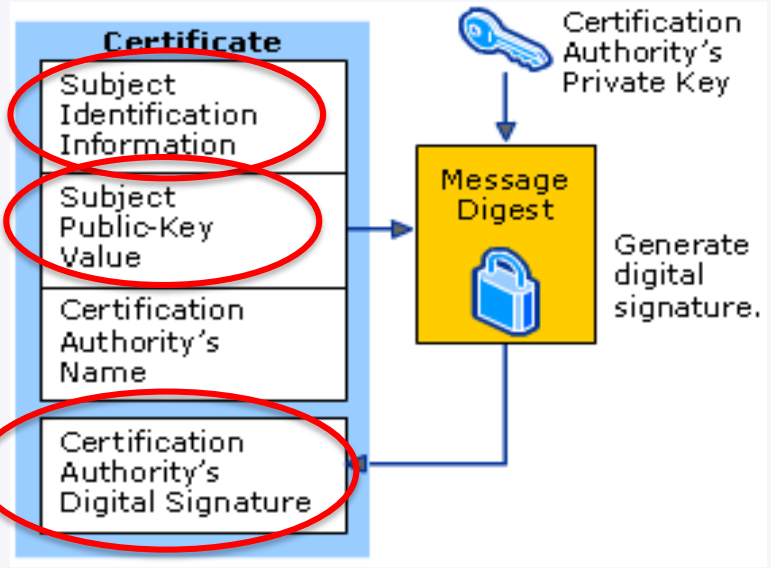
- Sensitive traffic?
  - Banking credentials, credit card details, PII, emails etc
  - Use TLS
- Non-sensitive traffic?
  - Use TLS!

- If no TLS…
  - MitM can modify traffic, inject exploits, replace content, track you etc

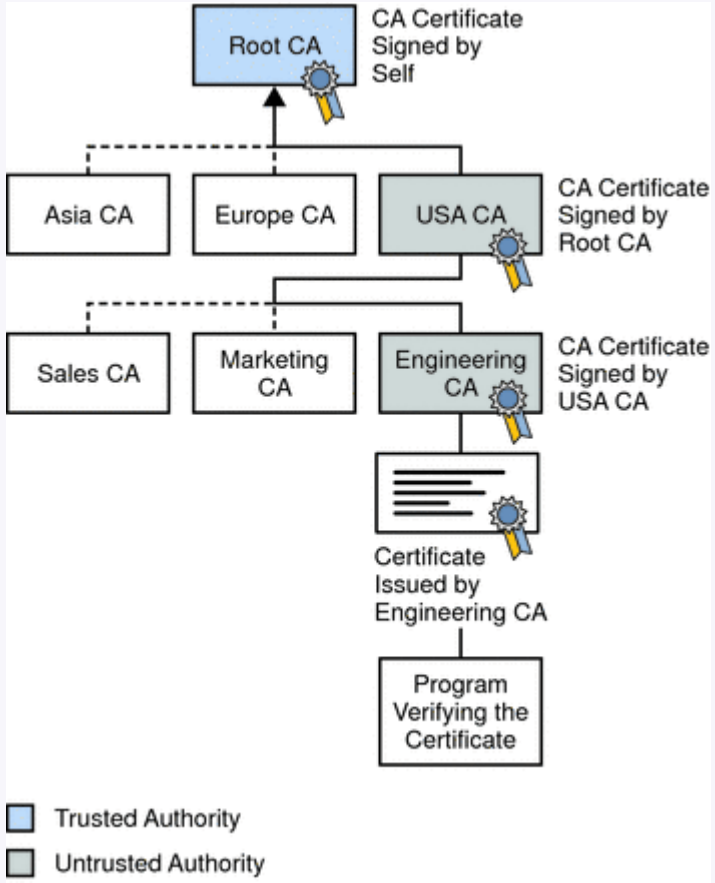
- TLS
  - Confidentiality, Integrity, peer authentication…
  - HTTPS: secure by default
  - SSL Sockets: not secure by default

Certificates & X509 chains

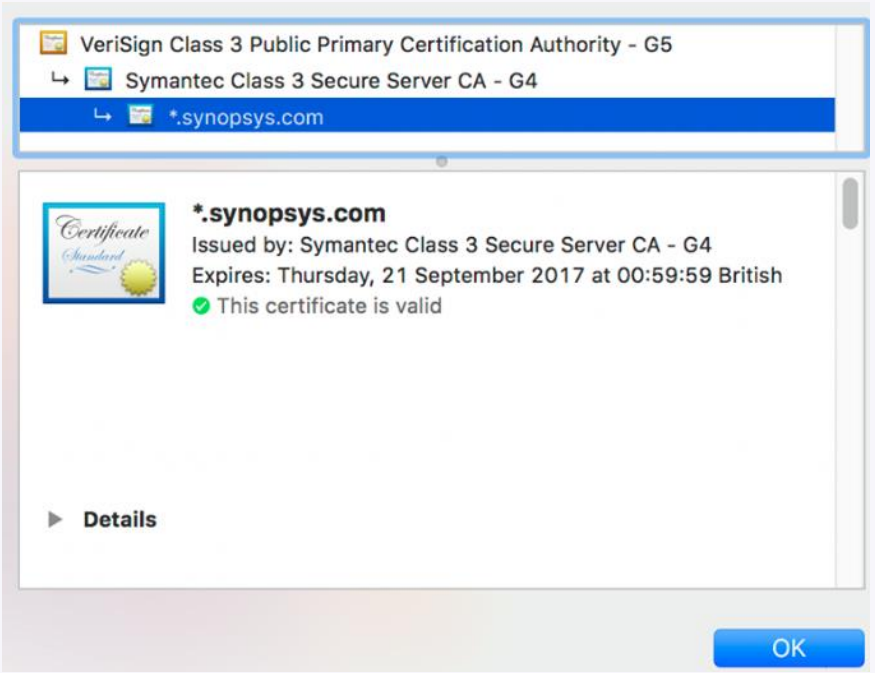
Certs bind an identity to Public Key



https://technet.microsoft.com/cc776447.7611c23d-baf0-4398-bea2-e2d7972f3f17(en-us,WS.10).gif



https://docs.oracle.com/cd/E19316-01/820-2765/images/chn.gif



Trust evaluation

1. Verify presented certificate is valid and chains to trusted anchor
2. Verify presented certificate was issued for the host you want
3. Verify backend holds private key corresponding to certified public key

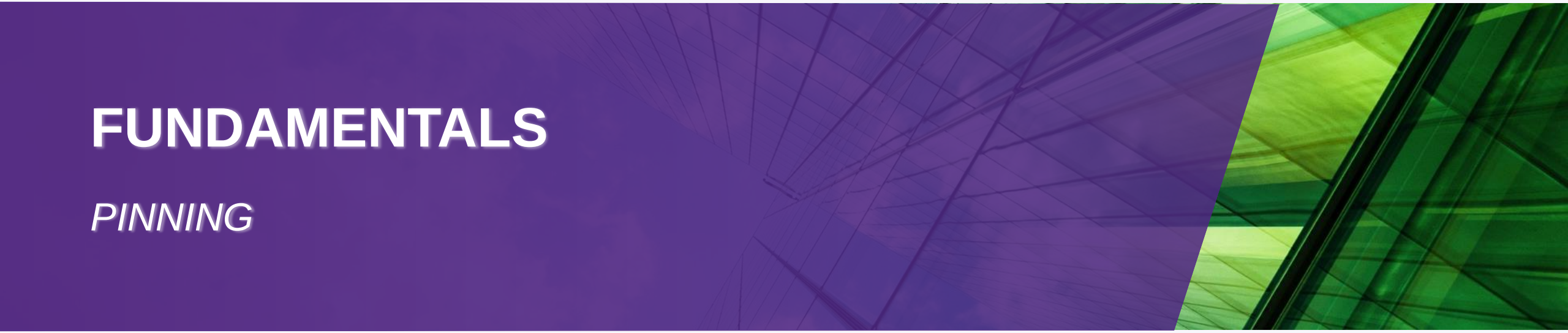
- Recursive X.509 certificate chain validation
  - Client has trusted anchors store
  - Client receives 1..N certs from server
  - Client assembles valid chain from received end-entity cert to a **trusted anchor**
  - Checks constraints & other cert fields of every cert

But.. mobile apps know their server

- Problem PKIX solves: Client does not know server's identity
- *But.. Most mobile apps do know the identity of their server*
- Can we do better than PKIX?

FUNDAMENTALS

PINNING



Why Pinning?

- Pinning goals
  - Control the process yourself, not depend on PKIX / CAs
  - Raise the bar above PKIX security guarantees
  - Shrink attack surface

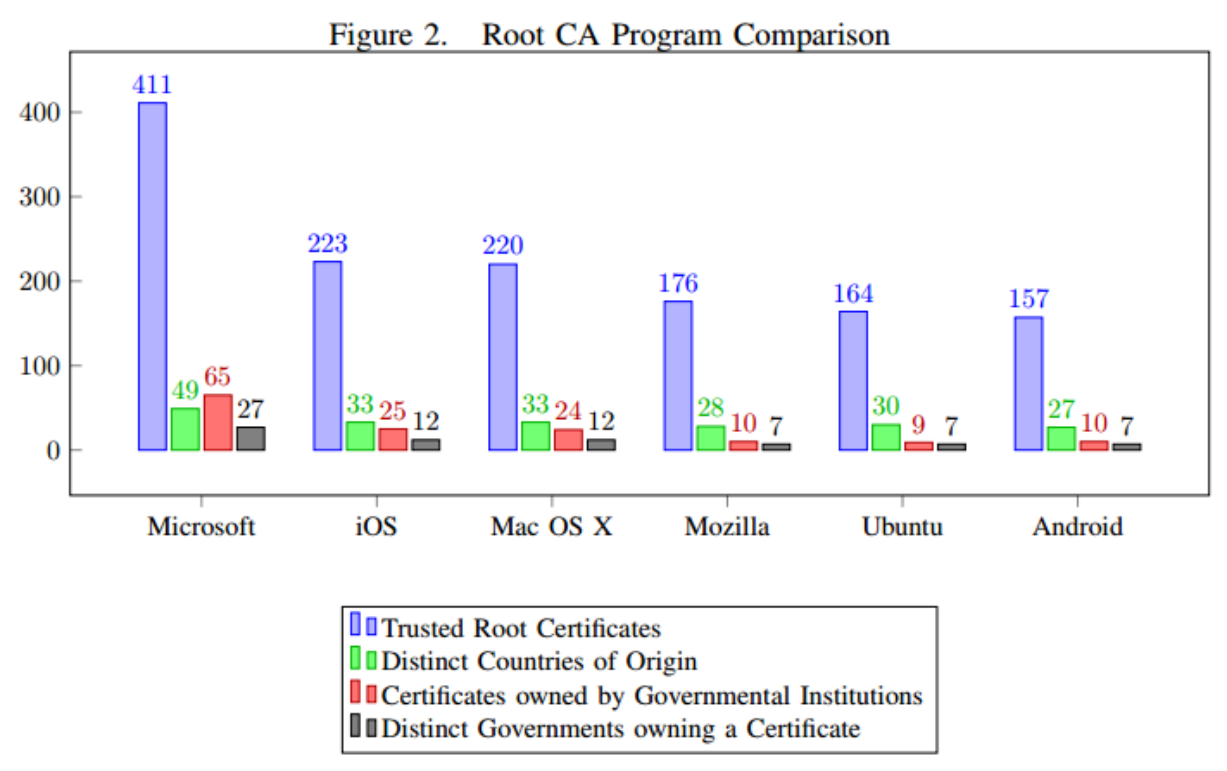
- Protection against *certificate forgery*
  - Rogue CAs
  - Compromised CAs
  - Mistakes by CAs
  - Users phished into inserting certs to device trust store
    - Android 7.0 fix

Trusted authorities?

Fake & mis-issued certificates

“Questioning the chain of trust”
report & series of talks by Andrew Blaich

- Government entities
- Small organizations
- CAs controlling other CAs
- Android device vendors add extra certs
- Removing bad CA from devices is not easy



Trust me, I'm a Root CA! Analyzing SSL Root CAs in Modern Browsers and Operating Systems. (ARES '15)

```
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
```

Android 5: 162
Android 6: 158
Android 7: 148
Android 7.1: 156

iOS 8: 201*
iOS 9: 187*
iOS 10: 165*

* Fully trusted

Past Failures

This section is 'further reading' for those interested

- Governments Want/Require Interce
  - Certified
- 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/

- http://support.go
- Interception proxies
  - http://blog.crypto
- HTTPS is broken
  - http://www.thoug
- PKI is broken
  - www.cs.aucklan
- The Internet is Broken :)
  - http://blog.cryptographyengineering.com/2012/02/how-to-fix-internet.html

- Researchers collided certificates on existing CA certificates
  - http://www.win.tue.nl/~bdweger/CollidingCertificates/ddl-full.pdf

- **Highlights in failures of trust:**
  - GlobalSign [2016]
  - DigiNotar [2011]
  - GlobalSign [2011]
  - India C
  - RapidS

Andrew Blaich –
https://sector.ca/
an-investigation-

9774

/dns_hijack_service_updated/

nodern-browse

n_victim_of_tmobiles_web_flaws

ises-and-web-

's IMSI Catcher)

ting-cell-phone-calls/

es-ios-in-app-p

e subordinate CAs for money

y-hacked-1767

9

te-for-surveillance-3040095011/

Symantec employees fired for issuing rogue HTTPS certificate for Google

Unauthorised credential was trusted by all browsers, but Google never authorised it.

DAN GOODIN (US) - 22/9/2015, 06:15

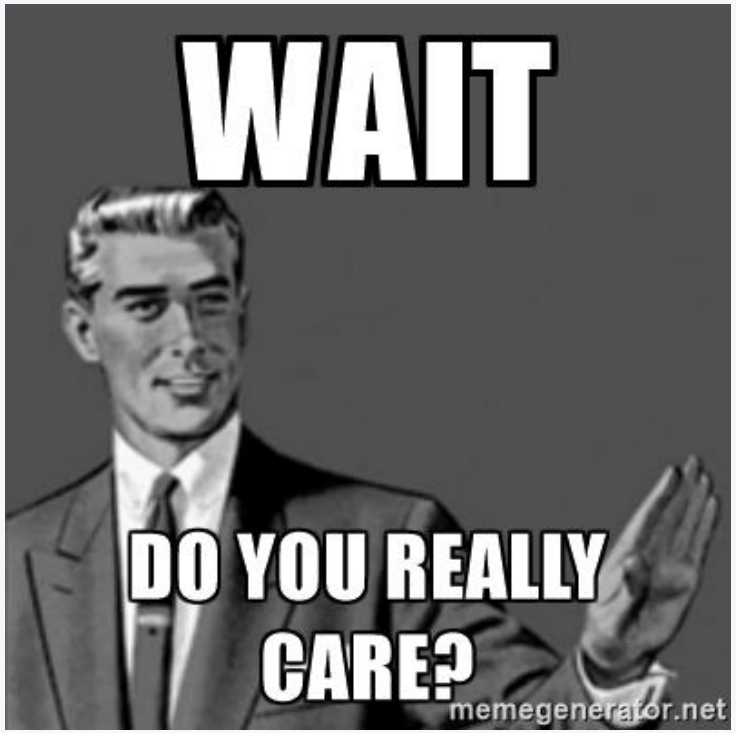
Already on probation, Symantec issues more illegit HTTPS certificates

At least 108 Symantec certificates threatened the integrity of the encrypted Web.

DAN GOODIN (US) - 20/1/2017, 22:45

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

Before pinning



- Concerned about *maliciously issued certificates*?
  - Yes: pin it
  - Maybe: good idea
  - Not really: PKIX good enough

The downside

- Will not protect against a compromised *pinned* certificate

- Will create a single point of failure
- Will require a mature process to avoid operational headaches
- May cause security issues like broken SSL validation
- May impact performance

Barclays Bank – Black Friday

On November 24, 2016 (Thanksgiving Day), an emergency situation arose whereby mobile application users of Barclays Bank would no longer be able to conduct transactions due to the pinning of an obsolete intermediate certificate in the application. The bank, through its application provider Axsy, urgently contacted Symantec to request a new certificate for *.payliquid.com <http://payliquid.com/> chained to the older intermediate

"The recent change to the intermediate certificate negatively impacted Barclay's SSL pinning solution. As a result, connection to our mobile application will fail for all users imminently. The only other option to fix this issue is underway and requires us to modify our existing iOS and Android mobile application code. This will take several weeks, including security testing, app store submission, approval and rollout.

https://cabforum.org/pipermail/public/2016-November/008989.html

Not against local attacks

- Will not stop reverse engineers
  - Frida, Xposed modules to unpin, debugging, repackaging all work
- Will not help if device is rooted/jailbroken
- Client-side controls: you can’t win, but can raise the bar
  - Look into binary hardening, tamper detection, obfuscation, move to native
  - Look into SafetyNet Attestation

What's a vulnerability?

- "Certificate Pinning bypassed using XXX local technique"
  - NOT a vulnerability
- "Absence of Certificate Pinning"
  - NOT a vulnerability, unless mandated by policy
- "Broken pinning implementation"
  - *IS* a vulnerability
    - Remote pinning bypass (low severity)
    - Remote TLS validation bypass (critical severity)

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?

OWASP AppSecEU 2016 talk

https://koz.io/certificate-pinning-owasp-appseceu16

Bugs, flaws and bad designs

“I see broken code”

Apps doing “custom” chain validation

- Please do not roll your own X.509 chain validation
  - Extremely complex
  - Use the system’s validation routines or 3rd party library like OpenSSL
  - Do not roll your own “chain cleaning” function either

Custom trust managers

```
customtrustManager =  new X509TrustManager() {
    @Override
    public void checkClientTrusted( final X509Certificate[] chain, final String authType ) {
    }
    @Override
    public void checkServerTrusted( final X509Certificate[] chain, final String authType ) {
    }
    @Override
    public X509Certificate[] getAcceptedIssuers() {
        return null;
    }
};
```

- Don’t do this – not even in debug builds

checkServerTrusted – payment processor SDK

```
private static class X509CertPinningTrustManager implements X509TrustManager
{
    public void checkServerTrusted(X509Certificate[] chain, String authType) throws CertificateException
    {
        if ((chain == null) || (chain.length == 0)) {
            throw new CertificateException("No X509Certificates found");
        }
        List pinsInfo = AppConfig.getPinningInfo();
        boolean validCertFound = false;
        if (pinsInfo != null) {
            for (pinnedCert : pinsInfo)
            {
                if ((pinnedCert.fingerprint == null) || (pinnedCert.fingerprint.length == 0)) {
                    throw new CertificateException("Invalid X509Certificate info provided.");
                }
                byte[] serverCertFingerprint = this.messageDigest.digest(chain[pinnedCert.chainPosition].getEncoded());
                this.messageDigest.reset();
                if (Arrays.equals(serverCertFingerprint, pinnedCert.fingerprint))
                {
                    validCertFound = true;
                    break;
                }
            }
        }
        if (!validCertFound) {
            throw new CertificateException("Invalid X509Certificate used.");
        }
    }
}
```

Chain validation??

Pinning checks

Chain can be unordered

Chain can have extra certs

checkServerTrusted – mobile payments app

```
public class PinningTrustManager implements X509TrustManager {
    private final List mPins;
    private final TrustManager[] mSystemTrustManagers;

    public void checkServerTrusted(X509Certificate[] certChain, String authType) throws CertificateException {
        if(certChain != null && certChain.length != 0) {
            for(i = 0; i < this.mSystemTrustManagers.length; ++i) {
                this.mSystemTrustManagers[i].checkServerTrusted(certChain, authType);
            }
            for(i = 0; i < certChain.length; ++i) {
                if (this.isValidPin(certChain[i])){
                    return;
                }
            }
            throw new CertificateException("No valid pins found in chain!");
        }
        throw new IllegalArgumentException("checkServerTrusted: X509Certificate is empty");
    }
}
```

System SSL validation

Pinning checks

Chain can have extra certs

checkServerTrusted

- Food ordering app

System SSL validation

Pinning check

```
@Override
public void checkServerTrusted(X509Certificate[] chain, String authType) throws CertificateException {
    boolean isValidPin = false;
    if (chain == null)
        throw new IllegalArgumentException("checkServerTrusted: X509Certificate array is null");

    if (chain.length < 0)
        throw new IllegalArgumentException("checkServerTrusted: X509Certificate is empty");

    try {
        TrustManagerFactory tmf = TrustManagerFactory.getInstance("X509");
        tmf.init((KeyStore) null);

        for (TrustManager trustManager : tmf.getTrustManagers()) {
            ((X509TrustManager) trustManager).checkServerTrusted(chain, authType);
        }
    } catch (Exception e) {
        throw new CertificateException(e);
    }
    RSAPublicKey pubkey = (RSAPublicKey) chain[1].getPublicKey();
    final byte[] conexionPin = pubkey.getEncoded();


    for (byte[] pin : authPins) {
        if (Arrays.equals(pin, conexionPin)) {
            isValidPin = true;
        }
    }

    if (!isValidPin) {
        throw new CertificateException("Invalid PIN");
    }
}
```

OkHttp library

```
// Check that the certificate pinner is satisfied by the certificates presented.
route.address.certificatePinner.check(route.address.uriHost,
    sslSocket.getSession().getPeerCertificates());
```

Peer Certificates?

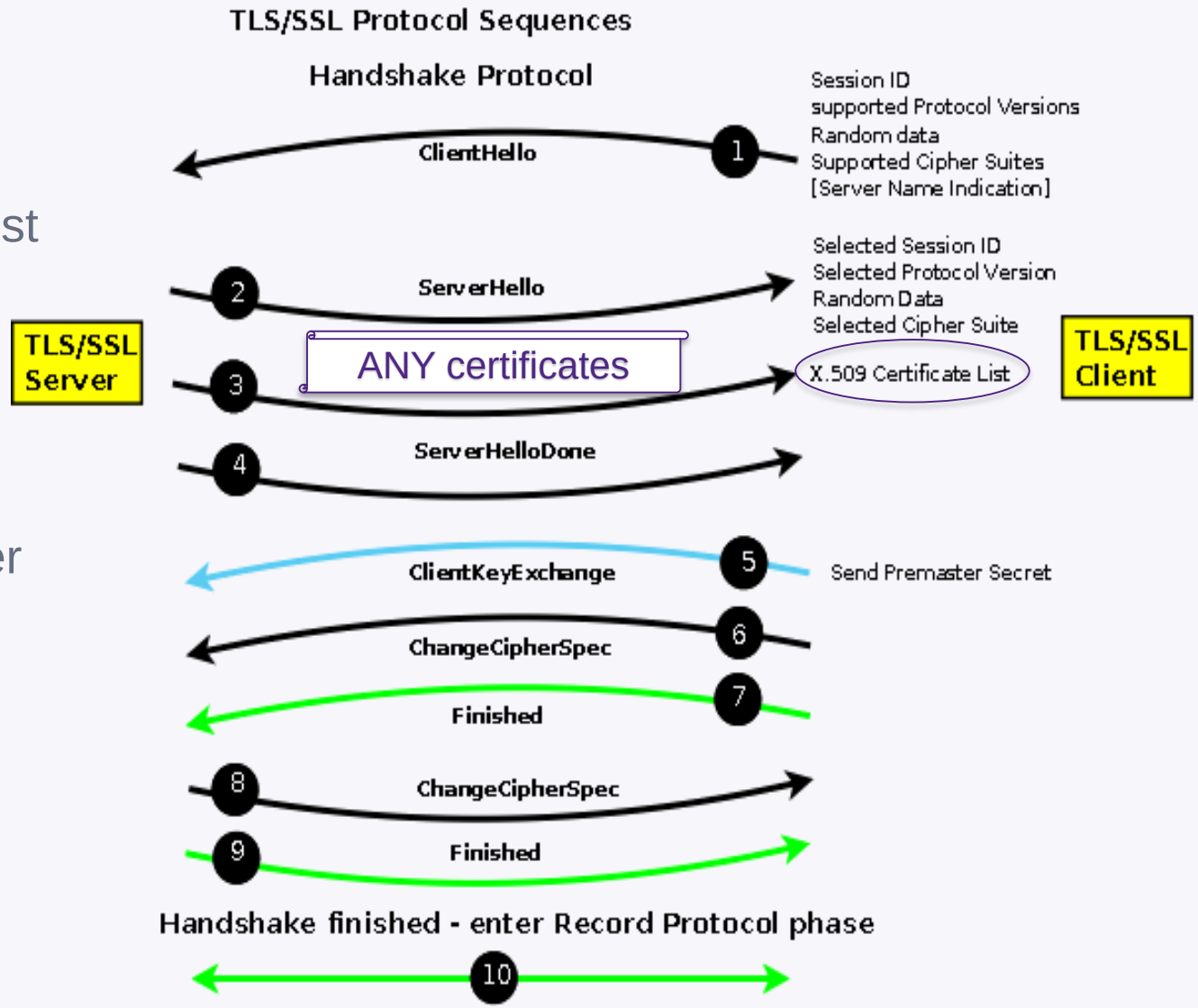
```
public void check(String hostname, Certificate... peerCertificates)
    throws SSLPeerUnverifiedException {
  List<ByteString> pins = hostnameToPins.get(hostname);
  if (pins == null) return;

  for (Certificate c : peerCertificates) {
    X509Certificate x509Certificate = (X509Certificate) c;
    if (pins.contains(sha1(x509Certificate))) return; // Success!
  }
```

What's going on?

- Server can send ANYTHING in its certificate list
- **`getPeerCertificates()`** and **`checkServerTrusted()`** return all certs AS SENT by server

- MitM attacker can send all certs the real server would send to client – including pinned certs



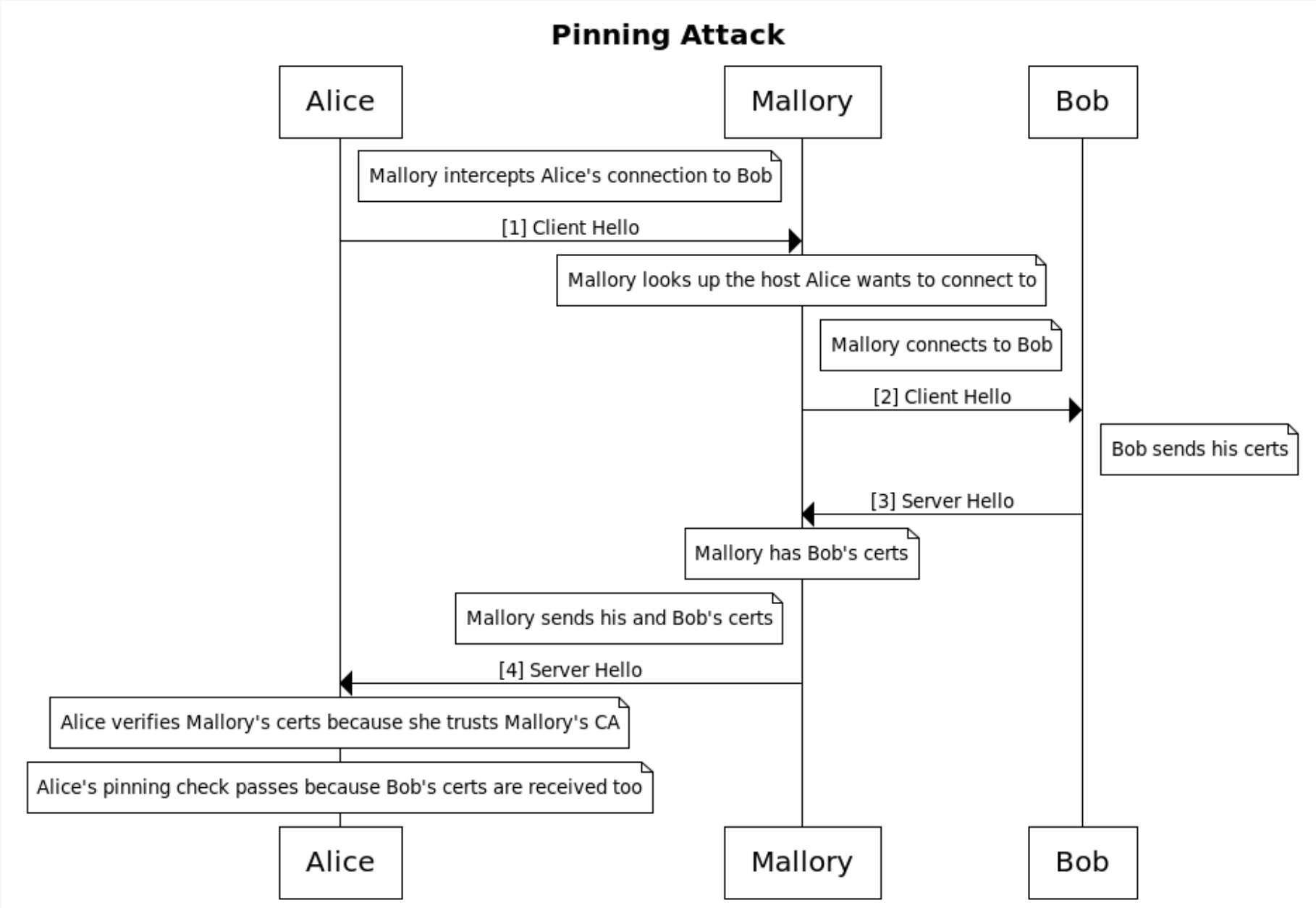
http://www.zytrax.com/tech/survival/ssl.html

getPeerCertificates() bug impact

- Pinning rendered completely ineffective

- 6 US & UK banks, 4 retailers, 2 payment apps
- 10s other less well known apps I've checked
- 6 libraries & SDKs used by 100s of apps
  - OkHttp < 3.1.2 & < 2.7.5 – CVE-2016-2402
  - SSLCertificateChecker-PhoneGap-Plugin < 4.0.0 if checkInCertChain=True
  - Including 2 commercial Android obfuscation products
- Several Java apps

- https://koz.io/pinning-cve-2016-2402/ [testing tools]
- https://www.cigital.com/blog/ineffective-certificate-pinning-implementations/


Pinning Attack
Alice
Mallory
Bob
Mallory intercepts Alice's connection to Bob
[1] Client Hello
Mallory looks up the host Alice wants to connect to
Mallory connects to Bob
[2] Client Hello
Bob sends his certs
[3] Server Hello
Mallory has Bob's certs
Mallory sends his and Bob's certs
[4] Server Hello
Alice verifies Mallory's certs because she trusts Mallory's CA
Alice's pinning check passes because Bob's certs are received too
Alice
Mallory
Bob

Fixing custom implementations

- Pins MUST checked on VALID chain, unless pinning to end-entity cert (check chain[0])
- Android & Java did not expose good APIs for this in the past
- Android
  - Since API 17: `X509TrustManagerExtensions` returns clean valid chain
    - Corner cases fixed in API 24
- Oracle Java
  - Bug status: “Issue fixed in main codeline, scheduled for a future Critical Patch Update”
  - “`SSLSession.getPeerCertificates()` should be sanitized”
  - Java SE 9 `getPeerCertificates()` documentation updated

> Note: The returned value may not be a valid certificate chain and should not be relied on for trust decisions.

Apps skipping X509 chain validation

- Cleaning / reordering cert chain? ✓
- Checking clean chain against pins? ✓
- Certificate chain validation? ✗

```
public void checkServerTrusted(X509Certificate[] chain, String authType)
{
    cleanChain = CertificateChainCleaner.getCleanChain(chain);
    for (X509Certificate certificate : cleanChain) {
        if (isValidPin(certificate)) {
            return;
        }
    }
}
```

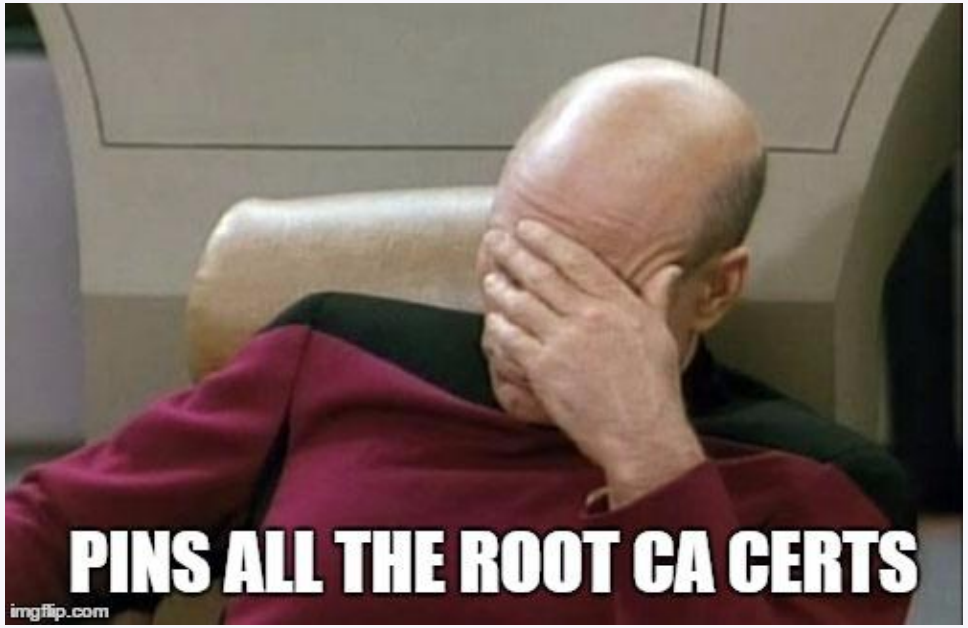
Always do this

```
public void checkServerTrusted(X509Certificate[] chain, String authType)
{
    cleanChain = CertificateChainCleaner.getCleanChain(chain);
    for (TrustManager systemTrustManager : systemTrustManagers) {
        systemTrustManager.checkServerTrusted(cleanChain, authType);
    }
    for (X509Certificate certificate : cleanChain) {
        if (isValidPin(certificate)) {
            return;
        }
    }
}
```

Apps pinning all the things

- App replaces system's trust store with custom trust store
- Custom trust store holds the 20 most popular root CA certificates

- DO: Pin only to certs on your chain
  - Possibly add one off-chain backup



Apps using any pin for any host

- App connects to 10 different hosts
- App holds pins for 10 different hosts
- App uses any stored pin for any host

- DO: Pin – to – Host mapping

TOCTOU bugs

- App uses pin validation for first connection to $HOSTNAME
- App skips pin validation for all new connections to $HOSTNAME

- Pin validation should be done for every full SSL handshake to pinned hosts
  - Connection pooling & SSL caching not always used

Broken caching

- App skips pin validation if ping previously checked (cert found in cache)
- OK if caching end-entity cert. Broken if caching CA certs.

```
public void checkServerTrusted(X509Certificate[] certChain, String authType) {
    for (pinnedCert : this.cache)
    {
        if(certChain.contains(pinnedCert)) {
            return;
        }
    }
    cleanChain = this.certChainCleaner(certChain);
    this.checkSystemTrust(cleanChain, authType);
    pinnedCert = this.checkPinTrust(cleanChain);
    this.cache.add(pinnedCert);
}
```

attacker can include any pinned cert

Broken Hostname Verification

- App does pinning correctly, but…

```
HttpsURLConnection.setDefaultHostnameVerifier(new HostnameVerifier() {
    public boolean verify(String hostname, SSLSession session){
        return true;
    }
});
```

- OR…

```
HttpsURLConnection.setDefaultHostnameVerifier(new HostnameVerifier() {
    public boolean verify(String hostname, SSLSession session){
        return hostname.equals("www.oursafebank.com");
    }
});
```

- Look closely: Verification always succeeds, SSLSession never checked
- Also please don't use `ALLOW_ALL_HOSTNAME`…

Multiple connection handlers

- Apps consist of several libraries and classes
  - Each class or library might have its own connection handling & pinning code
  - Have seen app with 4 networking stacks
    - 3 out of 4 did pinning
    - 1 pinning implementation was broken
    - 1 didn't have any pins

- Sometimes people *think* they use pinning, but they really don't

- Pin *all* connections to important hosts
- Try to take control of ALL connections in your app
- Centralize your implementation

Insecure fallback design

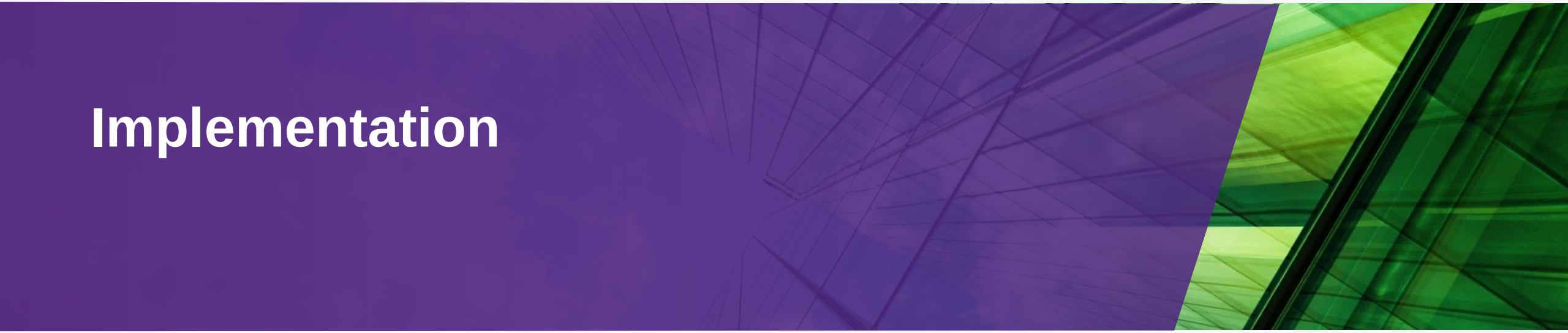
- App decides to use fallback certificate to avoid self-DoS
- Normal pin: end-entity certificate
- Fallback pin: CA certificate

- Flexibility: You can reissue a certificate using the same CA without DoS

- But: system as secure as pinning just to CA certificate
- No gain by pinning to end-entity
  - just added complexity
  - Could pin just to CA instead

- Optimal strategy: fallback pins should maintain the same security guarantees

Implementation



Taxonomy

- TYPE I
  - **System** SSL validation using **custom** trust anchors
- TYPE II
  - **Custom** SSL validation using **custom** trust anchors
- TYPE III
  - Pins checked against **end-entity** certificate – **no SSL validation** performed
- TYPE IV
  - **Pins** checked after **system** validation using **system** trust anchors
- TYPE V
  - **Pins** checked after **custom** validation using **system** trust anchors
- TYPE VI
  - **Pins** checked after **custom** validation using **custom** trust anchors (overkill)

Handling pinned connections

- Use Android Network Security Config (API 24+)
- Use a 3rd party networking library with a pinning feature
- Centralize connection handling in *one* place within the app
- Invoke your library API for *each* connection

- Automatically direct most* connections to your API
  - iOS: `NSURLprotocol` swizzling
  - Android: `URL.setURLStreamHandlerFactory()`
    - Not the easiest API to use

* excludes WebViews, non-HttsUrlConnection APIs etc

Android Libraries

- OkHttp
  - By Square – Jesse Wilson (@swankjesse) and others
  - Full featured, fast, efficient connection handling
  - Cert pinning API in ConnectionBuilder
- CWAC-NetSecurity
  - By Mark Murphy (@commonsguy)
  - Limited backport of Network Security Config up to API 17 - supports TOFU mode
- TrustKit-Android
  - By Datatheorem - Alban Diquet (@nabla-c0d3) and others
  - Limited backport of Network Security Config to up to API 17 – supports reporting mode
  - Less invasive than CWAC-NetSecurity, can be deployed down to API 15 in disabled mode
- AndroidPinning
  - By Moxie Marlinspike (@moxie)
  - GPLv3, dated, includes 4 year old trust anchor store

Custom implementation

- Android:
  - `X509TrustManager.checkServerTrusted()`
    - Avoid – if used, always do cert chain cleaning & call system trust managers using clean chain
  - `X509TrustManagerExtensions.checkServerTrusted()` (API 17+)
    - Check pins on returned validated cert chain
  - For SSLSockets: `X509ExtendedTrustManager` (API 24+)
- iOS:
  - `SecTrustEvaluate(SecTrustRef trust, SecTrustResultType *result);`

- System's OpenSSL library
  - Don't. Not great benefit, also restricted in API 24+
- Statically compile OpenSSL (or other)
  - More resistant to local attacks, tricky

Pinning & WebViews

- WebViews have two components
  1. Connection handler
  2. Rendering engine
- Android API 24+
  - Use Network Security Config ! Pinning works, `usesCleartextTraffic` may work in API 26
- Android API < 24
  - Intercept outbound requests using `shouldInterceptRequest()`
  - Load request using own handler, feed response data back into WebView
  - Not clean, synchronous, issues with POST requests
- iOS
  - Intercept connections using `NSURLprotocol:startLoading()`
  - load using own handler, feed response data back to protocol
  - Pinning & WKWebView = complicated – `didReceiveAuthenticationChallenge()`

Native OS pinning support

Things are getting better

Strict Mode & usesCleartextTraffic

- StrictMode
  - For app debugging..
  - `detectCleartextNetwork()` since API 23
  - Deep packet Inspection
    - Checks outgoing connections for TLS Client Hello bytes

```
StrictMode.setVmPolicy(
    new StrictMode.VmPolicy.Builder()
        .detectCleartextNetwork()
        .penaltyDeathOnCleartextNetwork()
        .build()
    );
```

- `usesCleartextTraffic` manifest flag

```
<application android:usesCleartextTraffic="false" />
```

  - API 23+ but ignored in API 24+ if Network Security Config exists
  - BEST EFFORT: Just an indication
    - for some system-provider connection APIs and popular 3rd party libraries
    - WebViews do NOT respect this (yet)

Android Network Security Configuration

res/xml/network_security_config.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>
```

```
<?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>
```

Android Network Security Configuration

- Policies
  - Which anchors to use
    - default: system & user API <24, only system API 24+
  - Static certificate pinning, with backup pins & optional expiration time
  - What to do with cleartext traffic
- Flexible: app-wide, per-domain or debug-only config

- Hidden features
  - “Debug only” config files! `/res/xml/my_config_file_debug.xml`
  - HSTS enforcement – NOT quite there yet…
  - CertificateTransparancy enforcement – SOON? ☺

HSTS enforcement

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config hstsEnforced="true">
        <domain includeSubdomains="true">secure.example.com</domain>
    </domain-config>
</network-security-config>
```

As of API 25, not used anywhere yet ¯_(ツ)_/¯

- But WebView supports HSTS (chromium)

Android Network Security Configuration deep dive

- Surprisingly similar to solution provided by the “Pin it!” paper by SBA-Research
- App gets set up at runtime with an Android Network Security Policy Provider (`AndroidNSSP`)

```
int pos = Security.insertProviderAt(new NetworkSecurityConfigProvider(), 1);
```

```
public NetworkSecurityConfigProvider() {
    // TODO: More clever name than this
    super("AndroidNSSP", 1.0, "Android Network Security Policy Provider");
    put("TrustManagerFactory.PKIX", PREFIX + "RootTrustManagerFactorySpi");
    put("Alg.Alias.TrustManagerFactory.X509", "PKIX");
}
```

- Creates custom `NetworkSecurityTrustManager`
  - Extends `X509ExtendedTrustManager` which implements `X509TrustManager`
  - Creates new keystore, adds all configured or default anchors
  - Creates a new `trustManagerImpl` for keystore
  - Implements `checkServerTrusted()` properly, checking pins in *trusted path*
    - Also checks pinSet expiration time and if pinning is enabled
- `NetworkSecurityTrustManager` is installed as the default `TrustManager` of the runtime

New APIs - NetworkSecurityPolicy

- `NetworkSecurityPolicy` class – since API 23
- `isCleartextTrafficPermitted()` since API 23
- `isCleartextTrafficPermitted(String hostname)` since API 24
- `isCertificateTransparencyVerificationRequired(String hostname)` – API 26?
  - False by default, but conscrypt support is there
  - Controlled using system properties, e.g. `conscrypt.ct.enforce.com.google.www`
- Problem: Some apps modify AppConfig at runtime using reflection. Fixed upstream.

New APIs - X509TrustManagerExtensions

- `X509TrustManagerExtensions` since API 17
- `checkServerTrusted()`
  - returns validated trusted path
- `isUserAddedCertificate(cert)` – since API 21
  - Checks if cert inside `/data/misc/user/0/cacerts-added`
- `isSameTrustConfiguration(host1, host2)`
  - SystemAPI for now, checks if two hosts share the same network security config

New APIs - X509ExtendedTrustManager

- `X509ExtendedTrustManager` since API 24
- Before: chain checks in TLS layer, hostname verification at layer above (HTTPS/LDAPS etc)
- Problem: Too many developers using `SSLSocket` without hostname verification

- Now: `X509ExtendedTrustManager` does both checks for SSL Sockets

- New `checkServerTrusted()` and `checkClientTrusted()` APIs
  - 3rd argument is `Socket` or `SSLEngine`
  - checks peer's identity in `SSLParameters` vs end-entity X509 certificate
  - Checks `SSLParameters` algorithm constraints for every cert in path
    - subject public key, signature algorithm, key usage, extended key usage etc

- conscrypt's `TrustManagerImpl` changed to this

Apple App Transport Security

- App Transport Security (ATS) – introduced in iOS 9
  - Blocks all non-HTTPS connections, similar to Android's StrictMode
- WWDC (June 2016)

> *At the end of 2016, Apple will make ATS mandatory for all developers who hope to submit their apps to the App Store.*

- August 2016

> Google gives developers code to disable iOS 9 app security to continue to serve ads

- 21 December 2016:
  - ATS enforcement postponed

> App Transport Security (ATS), introduced in iOS 9 and OS X v10.11, improves user security and privacy by requiring apps to use secure network connections over HTTPS. At WWDC 2016 we announced that apps submitted to the App Store will be required to support ATS at the end of the year. To give you additional time to prepare, this deadline has been extended and we will provide another update when a new deadline is confirmed.

Android ideas

- Extend policy to cipher selection?
  - Enforce TLS1.2 or TLS1.3, drop SHA1 certs etc
- Runtime policy configuration is needed by some apps
  - e.g. for runtime pin updates
- Configurable policies per content source
  - allow non-HTTPS media content, allow insecure content in WebViews etc
- HSTS
- HPKP (?)
- Certificate Transparency

Summary

Not everyone needs pinning

Pinning doesn’t stop local attacks

Pinning is an operational headache – design it carefully

Too easy to get custom implementation wrong

Use Android Network Security Configuration

..or a good 3rd party library

SYNOPSYS®
Silicon to Software™

Thank You

Questions?

Contact Information

- **John Kozyrakis**
  - Twitter: @ikoz
  - Email: John.Kozyrakis@synopsys.com
  - https://koz.io


SYNOPSYS®
Silicon to Software™