Chained TLS Cert Verification

Problem

The central challenge is that as we have more application repos appearing on both servers and peer devices on local networks, we need to handle the fact that the majority of these will not have certificates signed by a Root CA and cannot be pinned. As an example, the Kerplapp app mentioned runs a tiny HTTPS server on your device, and we need a way to verify that cert in a dynamic way.

TLS secret key pinning is great when you have a finite amount of known, centralized servers. We aren't doing that, so we need something more flexible. We've taken the first step with ChatSecure (which incorporates AndroidPinning and Memorizing Trust Manager, without a Root CA store), but we want to standardize this a bit more for Bazaar / F-Droid.

Definitions

- **SPKI** - "Subject Public Key Identifier", the public key, key size, and key type in a single X509 record
- **fingerprint** - the hash over a standard chunk of a key
- **pin** - a hostname, SPKI, CA-signed boolean, and optional expire date to compare all connections to
- **TOFU** - the process of prompting the user whether to add a pin or not

Flowchart

(source)

https://raw.githubusercontent.com/guardianproject/GuardianProjectPublic/master/Chained%20TLS%20Verification/Chained%20TLS20Verification%20flowchart.png

Discussion and Resources

- Rethinking SSL Development in an Appified World
- https://www.imperialviolet.org/2011/05/04/pinning.html
- http://www.thoughtcrime.org/blog/authenticity-is-broken-in-ssl-but-your-app-ha/
- http://tack.io/
- https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning
- [[2013-12-13 IRC conversation]]
- [[2014-08-01 IRC conversation]]

Certificate vs. Secret Key

One question is whether to use the specific certificate or just the site's key that signed the certificate in the verification. Another option is having both included in the verification chain. First, cert pins would checked, then private key pins.

If the certificate is what is checked, it is easier to implement:

- byte-by-byte comparison of the locally stored certificate versus the presented remote certificate
- Android checks APK signatures this way
- F-Droid checks index.jar signatures this way

If the secret key is what is checked, then it is more flexible:

- transitions to new certificates is easier when once expires or is revoked
- Google Chrome pins the secret key
- this is much more vulnerable: if the secret key is compromised, then the attacker could issue certs that would be trusted by the pin/tofu

Implementation

Merge existing code into one project.

- AndroidPinning: https://github.com/moxie0/AndroidPinning
- MemorizingTrustManager: https://github.com/ge0rg/MemorizingTrustManager

    - still fails miserably at hostname validation (can't be used to accept a cert for the wrong hostname)
    - MTM needs to respect expiration dates
- a version of this idea was implemented in F-Droid: https://gitorious.org/f-droid/fdroidclient/merge_requests/56