The Savage Curtain: Mobile SSL Failures
Who are these guys?

Tony Trummer - Staff Security Engineer aka “SecBro”

Tushar Dalvi - Sr. Security Engineer & Pool Hustler
Our employer generally does not have prior knowledge of, condone, support or otherwise endorse our research.
Apps are mash-ups of native and web code
Java, Objective C, Swift, etc.
Developers control SSL/TLS security settings
This is probably not the site you are looking for!

TLS provides several security features:

- Encryption
- Authenticity
- Integrity

In apps, unlike browsers, whether you see a certificate warning is up to the app developer.
TLS is really the **ONLY** protection against Man-in-the-middle (MitM) attacks

MitM is significantly easier to perform against mobile devices
Before dismissing the idea of large-scale or supply-chain attacks...

- Recent reports of pre-installed trojans on low-end Android devices
- In 2013, Nokia was found to be performing MitM on customer traffic, reportedly for performance reasons
- In 2013, reports surfaced claiming that the NSA and GCHQ (“Flying Pig”) were actually performing real-world MitM attacks
Infosec folks often roll their eyes when they read statements on sites or in apps that tout TLS use and how big their keys are.
One night, after a few drinks, we decided to test some apps, starting with proxying their web requests.
The app or OS must verify the certificate is cryptographically signed by the private key of a trusted Certificate Authority.
Proper certificate validation

Certificate is signed by the private key of a trusted CA?

Is this an intermediate certificate?

Trusted Root CA
DOUBLE FACEPALM
FOR WHEN ONE FACEPALM DOESN'T CUT IT
Vulnerability Note VU#582497

Multiple Android applications fail to properly validate SSL certificates

Credit

This vulnerability was reported by Will Dormann of the SEIFUG. Additional reporters of the concept of Android apps that fail to validate SSL certificates include Tony Trummer, Tushar Dalvi, and Kuo Chiang. Other individuals that publicly reported this issue include: Sascha Fahl, Marián Hrabec, Thomas Widders, Matthew Smith, Lars Baumgärtner, and Bernd Freisleben.

SSL Vulnerabilities in the Google Play 1,000 Most Downloaded Applications
A Taste of Armageddon
The Trouble with Tribbles
Testing for CA validation

- Configure device to use proxy
- Configure BurpSuite's proxy listener to "Generate a CA-signed per-host certificate"
- **DO NOT** install the proxy's CA certificate on the test device
- Verify you see a certificate warning in the native mobile browser
- Step through each section of the app
- If you see HTTPS traffic, in Burpsuite, the app failed
Second aspect of validation

Does the Subject Common or Alternative name match the hostname of the site you're visiting?
Proper certificate validation

Does the Common or Subject Alternative Name Match the hostname?

Traces back to Trusted Root CA
By any other name
By any other name
The Apple

```c
static OSStatus
SSLVerifySignedServerKeyExchange(SSLContext *ctx, bool enable, bool crlCheck)
{
    OSStatus err;
    SSLBuffer hashOut, hashCtx, clientRandom, serverRandom;
    uint8_t *signature, UInt16 signatureLen;
    sizes_t signedHashes;
    *dataToSign;
    dataToSignLen;

    ... if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom))
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom))
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams))
        goto fail;
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
        goto fail;

    err = sslRawVerify(ctx,
        ctx->peerPubKey,
        dataToSign,
        /* plaintext */
        dataToSignLen,
        /* plaintext len */
        signature,
        signatureLen);

    if(err) {
        sslErrorLog("SSLDecodeSignedServerKeyExchange: sslRawVerify FAILED: %s", err);
        goto fail;
    }
}
```
And the Children Shall Lead

AFNetworking 1.3.4
matt released this on Apr 15, 2014 • 797 commits to master since this release
• Fix potential non-terminating loop in connect
• Fix SSL certificate validation to assert that no internal (Maximilian Dorsofl)
• Fix SSL certificate validation to provide a human
• Fix to add explicit cast to NSInteger in format
• Fix to call SecTrustEvaluate before calling certificate validation (Josh Chung)

AFNetworking 2.2.2
matt released this on Apr 15, 2014 • 319 commits to master since this release
• Add unit test for checking content type (Diego Torres)
• Add boundary property to AFHTTPBodyPart (copyWithZone)
• Add removeKeysWithNullValues property to AFJSONResponseSerializer to automatically
  remove NSNull values in dictionaries serialized from JSON (Matt Thompson)
• Change to accept id parameter type in HTTP manager convenience methods (Matt Thompson)
• Change to deprecate setAuthorizationHeaderFSaltWithToken; in favor of users
  specifying an Authorization header field value themselves (Matt Thompson)
• Change to use long long type to prevent a difference in stream size caps on 32-bit and 64-
  bit architectures (Yung-Luen Lan, Cédric Luthi)
• Fix calculation of Content-Length in TaskDoSendBodyData (Christos Vasilakis)
• Fix for comparison of image view request operations (Matt Thompson)
• Fix for SSL certificate validation to check status codes at runtime (Dave Anderson)

AFNetworking 2.5.2
matt released this a day ago • 4 commits to master since this release
• Add guards for unsupported features in iOS 8 App Extensions
• Add missing delegate callbacks to UIWebView category
• Add test and implementation of strict default certificate validation

Change to use case sensitive compare when sorting keys in query string
(Thompson)
Change to use xcpretty instead of xctool for automated testing (Kyle Full
• e to use @selector values as keys for associated objects (Matt
• o property synthesis warnings (Oliver Letterer)

Future certs being treated as valid in 2.5.1 #2573

THURSDAY, MARCH 26, 2015
SSL MitM attack in AFNetworking 2.5.1 • Do NOT use it
in production!

During a recent mobile application security analysis for one of our clients, we
identified a quite unobvious behaviour in apps that use the
AFNetworking library.

It turned out that because of a logic flaw in the latest version of the library,
SSL MitM attacks are feasible in apps using AFNetworking 2.5.1.
Black Box Analysis Results

The following tools were used for the black box analysis:

- otool (object file displaying tool)[1]
- Burp pro (proxy tool)[2]
- ssh (Secure Shell)

40% of the audited apps did not validate the authenticity of certificates presented. This makes them susceptible to Man-in-the-Middle (MiTM) attacks.[3]
By any other name
By any other name
Testing for proper hostname validation

- Install Portswigger CA cert on device
- Configure your device to use a proxy
- Configure proxy listener to “Generate a CA-signed certificate with a specific hostname”
- Set the hostname to foobar.com
- Verify you see a certificate warning in the native mobile browser
- Step through each section of the mobile app
- If you see HTTPS traffic, the app failed
Proper certificate validation

Does the Common or Subject Alternative Name Match the DNS hostname?

Not expired? Not revoked?

Traces back to Trusted Root CA
Damn it, Jim!
Credit card numbers, passwords, and/or session cookies
Dagger of the mind

- Unencrypted credit card information
- Tier 1 PCI merchant
- 10 million+ installations
FTC vs. Fandango & Credit Karma

- One of the major flaws cited in the suit was failure to validate SSL certificates on mobile applications.
- Agreed to “establish comprehensive security programs”
- Agreed to “undergo independent security assessments every other year for 20 years”
- Scolded publicly for not keeping “their privacy promises to consumers”
SSL session caching

- During the initial handshake the certificate is validated
- Subsequent client requests re-use the previous handshake and do not re-validate the certificate
- TOFU (Trust On First Use)
How would a bad guy get my phone?
Why is it more likely on mobile?
If I have physical access, couldn't I just...

- Install malicious app
- Access your data
Since SSL session caching only checks the certificate once, you only need it on the device for as long it takes you to make the first connection, after which you can delete it.
Server decides how long to accept the cached session

In other words, the bad guy gets to decide how long to accept the cached session...

We refer to this feature as “EverPWN”
Review your code
Implement policy
Test pre-release
Train developers
In Android, investigate these:
- TrustManager
- SSLSocket
- SSLSocketFactory.getInsecure
- HostNameVerifier

In iOS, investigate these areas:
- Don't use AFNetworking < v. 2.5.3
- _AFNETWORKING_ALLOW_INVALID_SSL_CERTIFICATES_
- SetAllowsAnyHTTPSCertificate
- kCFStreamSSLAllowsAnyRoot
Certificate Pinning

Dev and prod signing certificates are **required** to be different in both iOS and Android

Build validation models based on which certificate is used to sign the app
Contact and testing instructions:
http://www.secbro.com

Tony Trummer:
http://www.linkedin.com/in/tonytrummer
@SecBro1

Tushar Dalvi:
http://www.linkedin.com/in/tdalvi
@TusharDalvi

R.I.P Reggie Destin