Leibniz
Universität
Hannover

Hunting Down Broken SSL in Android Apps

Sascha Fahl, Marian Harbach,
Matthew Smith

Usable Security and Privacy Lab
Leibniz Universität Hannover

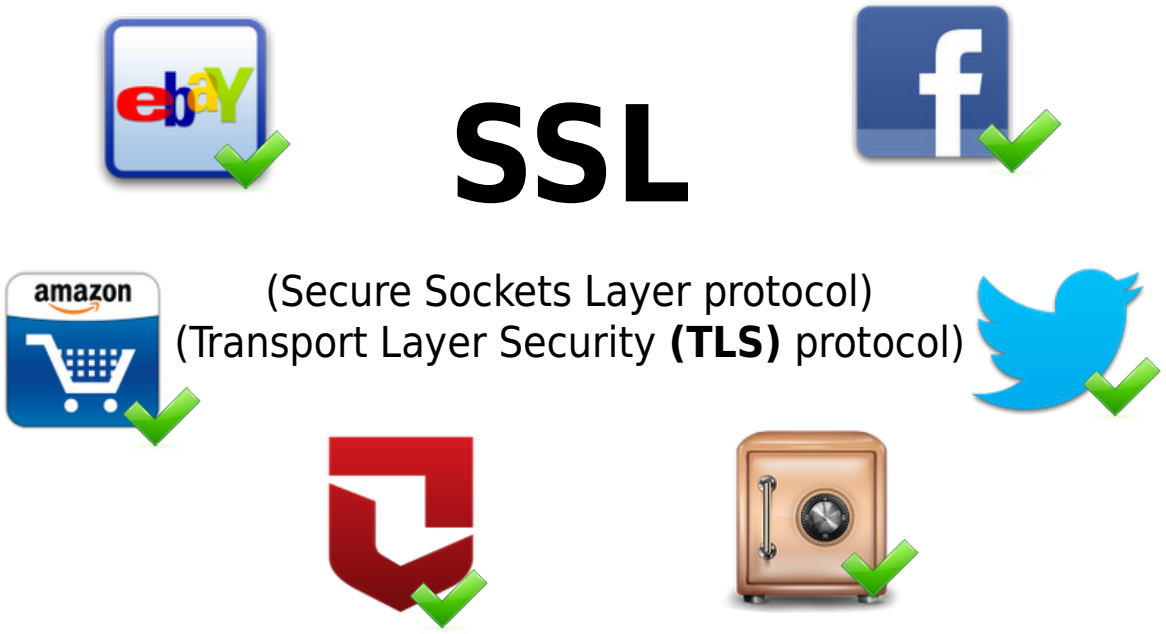
Appification

There’s an App for Everything

What do Most Apps Have in Common?

They share data over the Internet

Some of them even secure transfer using:



Secure Socket Layer

SSL is a cryptographic protocol and the mainstay of our Internet security

Why SSL?

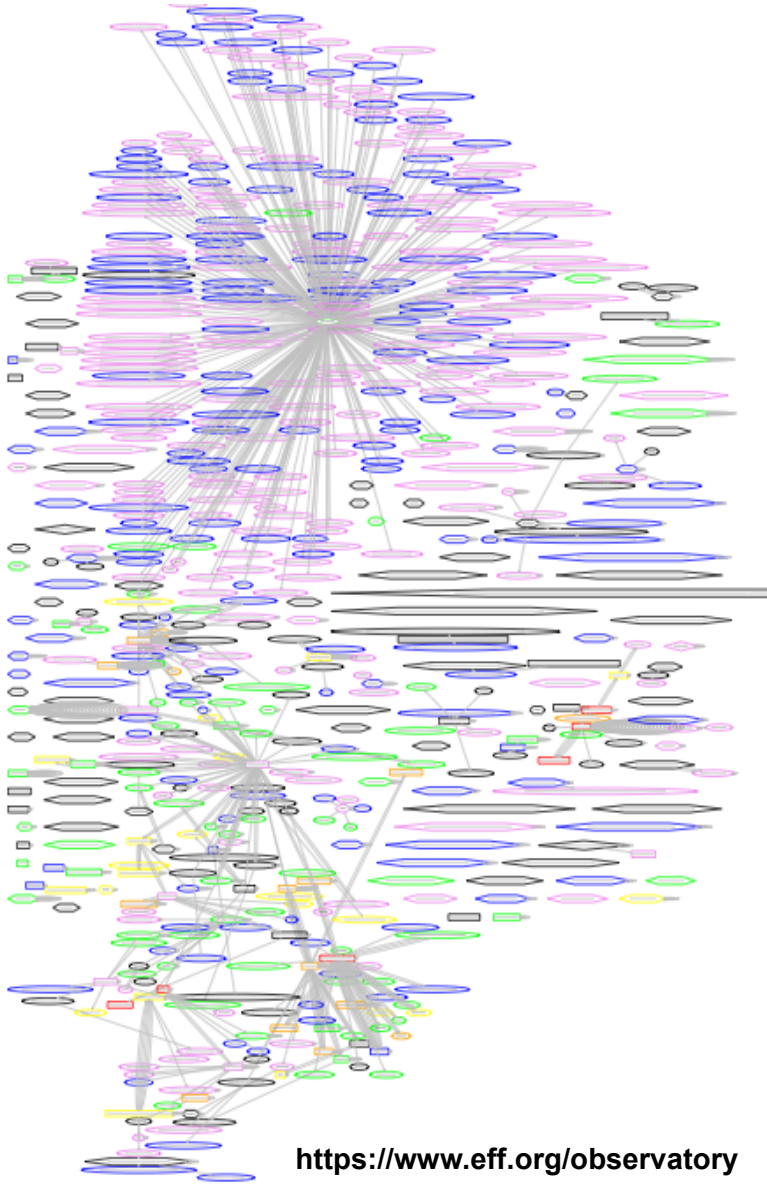
Authenticity

Integrity

Confidentiality

Problems with CAs (whom do we trust?)

- Approximately 100-200 trusted root CAs in
  - Firefox, Chrome, IE Explorer, Windows, Mac OS, Linux
  - Extended to ~650 via CA hierarchies
  - EFF Map of these organizations
- SSL / HTTPS only as strong as the weakest link
  - Weak (email-based) authentication with many CAs
  - Targeted attacks against CAs - a real world threat
  - As Comodo and Diginotar have shown



https://www.eff.org/observatory

Verifying Authenticity of a Certificate

In the current system the following checks are required:

Was the certificate signed by a „trusted“ CA?

Is the certificate expired?

Was the certificate revoked?

Does hostname verification succeed?

- i.e. is the domain name being accessed also the one specified in the certificate?

Default Android SSL Certificate API

Was the certificate signed by a „trusted“ CA?

Is the certificate expired?

Was the certificate revoked?

Does hostname match the certificate‘s CN?

Android SSL Certificate API and Revocation

Was the certificate revoked?

< Android 4.0 no straight forward way to enable certificate revocation

Android 4.0 introduces Certificate Blacklisting

- two system blacklists: for CA and end entity certificates
- the default certificate validation takes both blacklists into consideration

Android SSL Certificate API and Hostname Verification

Does hostname match the certificate‘s CN?

Hostname verification is done for all HTTPS connections

javax.net.ssl.SSLSocketFactory does no hostname verification!

- use android.net.SSLCertificateSocketFactory instead

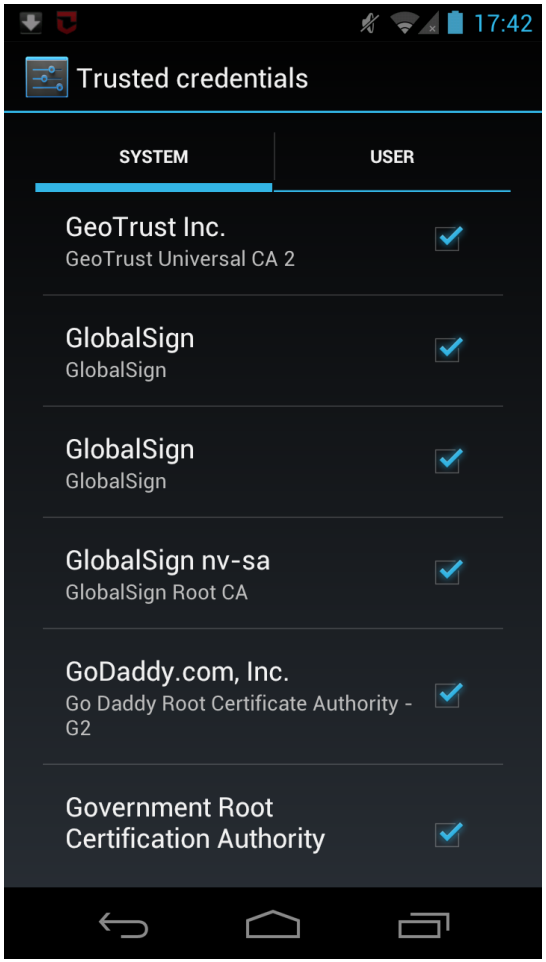
SSL Usage on Android

The default Android HTTPS API implements correct certificate validation.

What could possibly go wrong?

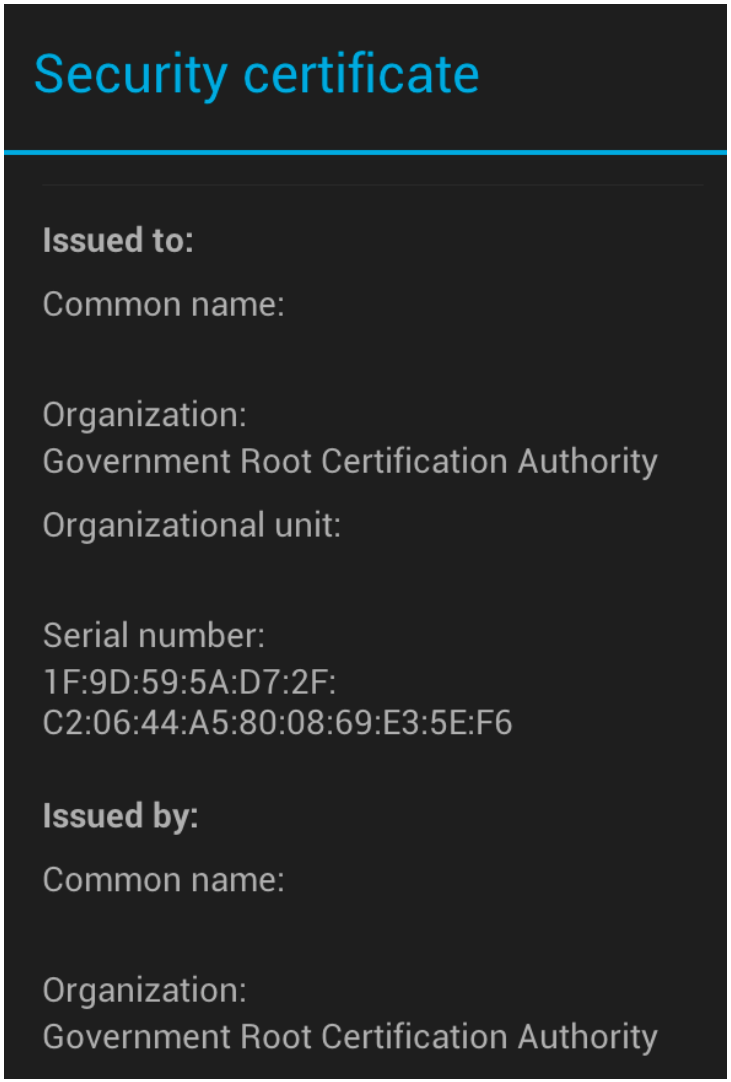
SSL Usage on Android

- A server needs a certificate that was signed by a trusted Certificate Authority (~130 pre-installed CAs)



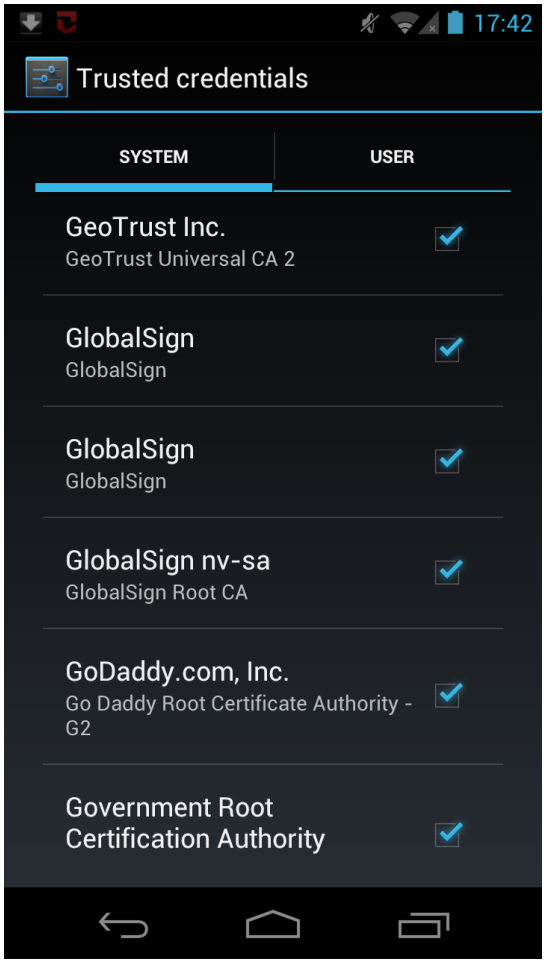
SSL Usage on Android

- A server needs a certificate that was signed by a trusted Certificate Authority (~130 pre-installed CAs)
- Some are quite strange...



SSL Usage on Android

- A server needs a certificate that was signed by a trusted Certificate Authority (~130 pre-installed CAs)
- For non-trusted certificates a custom workaround is needed
- For non-standard behaviour custom code is needed



Customizing SSL Certificate Validation on Android

Some situations require deviations from default certificate validation.

Examples:

- SSL Certificate Pinnin

- Error Handling

- Custom Certificate Authori

3 Ways to Customize Validation

Leibniz Universität Hannover

- javax.net.ssl.X509TrustManager
  - checkServerTrusted
- org.apache.http.conn.ssl.X509HostnameVerifier
  - verify
- android.webkit.WebViewClient
  - onReceivedSslError

What about using a non-trusted certificate?

Q: Does anyone know how to accept a self signed cert in Java on the Android? A code sample would be perfect.
A: Use the EasyX509TrustManager library hosted on code.google.com.

Q: I am getting an error of „javax.net.ssl.SSLException: Not trusted server certificate“. I want to simply allow any certificate to work, regardless whether it is or is not in the Android key chain. I have spent 40 hours researching and trying to figure out a workaround for this issue.
A: Look at this tutorialhttp ://blog.antoine.li/index.php/2010/10/android-trusting-ssl-certificate

stackoverflow.com

Our Analysis

1. downloaded 13,500 popular and free apps from Google's Play Market
2. static code analysis to find apps that implement customized certificate validation with MalloDroid:
   based on the androguard reverse engineering framework

   identifies apps that include custom SSL code

   tells you if an app breaks certificate validation in an obvious way

   tells you if an app contains code that code that potentially could break certificate validation

   decompiles apps to Java code for further manual analysis

Static Analysis with MalloDroid

depends on androguard (https://code.google.com/p/androguard)
very simple – only comes with few command line parameters
optional:
    xml output

    decompiles APK file for further analysis

simply type:
    ./mallodroid.py –f AppToCheck.apk –d ./javaout

available on github (https://github.com/sfahl/mallodroid)

Example

./mallodroid.py –f com.zoner.android.antivirus.apk –x
output:

```
<result package="com.zoner.android.antivirus">
    <trustmanagers>
        <trustmanager broken="True" class="com.zoner.android.antivirus.AlarmLiveThreat$3"
            <xref class="com.zoner.android.antivirus.AlarmLiveThreat" method="trustAllHosts"
        </trustmanager>
        <trustmanager broken="True" class="com.zoner.android.antivirus.AlarmUpdate$3">
            <xref class="com.zoner.android.antivirus.AlarmUpdate" method="trustAllHosts"/>
        </trustmanager>
        <insecuresslsocket/>
    </trustmanagers>
    <hostnameverifiers>
        <hostnameverifier broken="True" class="com.zoner.android.antivirus.AlarmLiveThreat$1">
            <xref class="com.zoner.android.antivirus.AlarmLiveThreat" method="&lt;clinit&gt;"/>
        </hostnameverifier>
        <hostnameverifier broken="True" class="com.zoner.android.antivirus.AlarmUpdate$1">
            <xref class="com.zoner.android.antivirus.AlarmUpdate" method="&lt;clinit&gt;"/>
        </hostnameverifier>
        <allowhostnames/>
    </hostnameverifiers>
    <onreceivedsslerrors/>
</result>
```

- MalloDroid helps you identify suspicious code
  - implementations very diverse => further manual analysis necessary

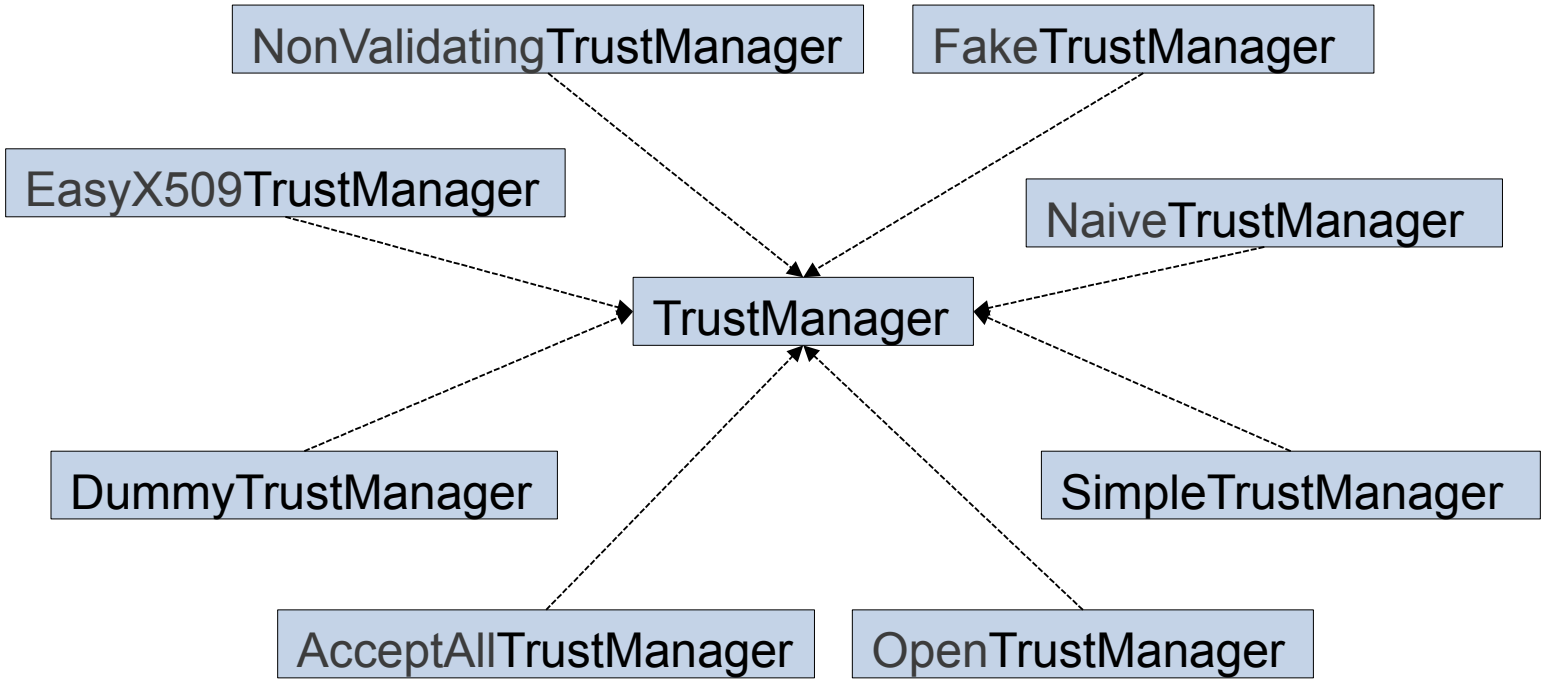
Static Code Analysis Results

92,8 % Apps use INTERNET permission
91,7 % of networking API calls HTTP(S) related
0,8 % exclusively HTTPS URLs
46,2 % mix HTTP and HTTPS
17,28 % of all Apps that use HTTPS include that fails in SSL certificate validation

- 1070 include critical code
- 790 accept all certificates
- 284 accept all hostnames

Custom TrustManagers

22 different TrustManager implementations



- and all turn effective certificate validation off
- Usability Problem: names are sometimes misleading!

Sample TrustManager Implementation 1

~90% of all custom implementations do this:

```
public void checkServerTrusted(java.security.cert.X509Certificate[] p1, String p2)
{
    return;
}
```

- attack: MITM-Attacker can inject arbitrary certificate
- if you find a customized TrustManager, chances are high it breaks certificate validation

Sample TrustManager Implementation 2

~7.5% of all custom implementations do this:

```
public void checkServerTrusted(java.security.cert.X509Certificate[] p3, String p4)
{
    p3[0].checkValidity();
}
```

- attack: MITM-Attacker can inject arbitrary certificate as long as it did not expire
- Usability Problem: name of cert.checkValidity() is misleading!

What does this mean for the apps?

Leibniz Universität Hannover

Static code analysis flags existence of dangerous code

Does not mean it is actually executed

Or that execution is actually dangerous for users!

Manual App Testing Results

Cherry-picked 100 apps

21 apps trust all certificates

20 apps accept all hostnames

Captured credentials for:

American Express, Diners Club, Paypal, bank accounts, Facebook, Twitter, Google, Yahoo, Microsoft Live ID, Box, WordPress, remote control servers, arbitrary email accounts, and IBM Sametime, among others.

Manual App Testing Results

These 41 apps had an install-base of 39 – 185 million!

Anti-Virus

Zoner AV

Textmasterformate durch Klicken bearb

- Zweite Ebene
  - Dritte Ebene
    - Vierte Ebene
      - Fünfte Ebene

- Anti-Virus App for Android
- Awarded best free Anti-Virus App for Android by av-test.org

Zoner AV

Virus signature updates via HTTPS GET
The good thing: It uses SSL
Unfortunately: The wrong way

It does not verify the hostname

And it does not check the update's authenticity!

```
static final HostnameVerifier DO_NOT_VERIFY = new HostnameVerifier()
{
    public boolean verify(String paramString, SSLSession paramSSLSession)
    {
        return true;
    }
};
```

Zoner AV - Video

BankDroid

Swedish banking app
Support for ~60 banks/payment services

- PayPal
- Steam Wallet
- Eurocard
- Swedbank
- ...

BankDroid - Aftermath

26 out of 41 broken
Deliberately broken
NO user warning

More Examples

Remote Control App

Remote Code Injection

Unlocking Rental Cars

Talking To Developers

Finding broken SSL in Android apps is good…

- …knowing what the root causes are is even better

We contacted 80 developers of broken apps

informed them ✓

offered further assistance ✓

asked them for an interview ?

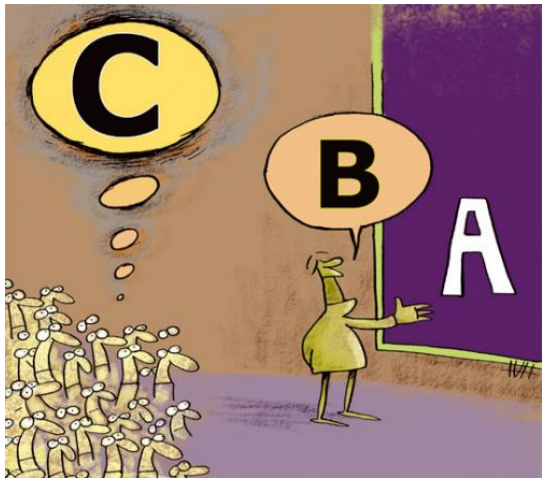
§ 15 developers agreed ✓

Statements (1)

“This app was one of our first mobile apps and when we noticed that there were problems with the SSL certificate, we just implemented the first working solution we found on the Internet. [...] We usually build Java backend software for large-scale web services.”

Statements (2)

“You said that an attacker with access to the network traffic can see the data in cleartext. I tried that and I connected my phone to a Wi-Fi hotspot on my laptop. When I used Wireshark to look at the traffic, Wireshark said that this is a proper SSL protected data stream and I could not see any cleartext information when I manually inspected the packets. So I really cannot see what the problem is here.”



Statements (3)

“The app accepts all SSL certificates because some users wanted to connect to their blogs with self-signed certs and [. . .] because Android does not provide an easy-to- use SSL certificate warning message, it was a lot easier to simply accept all self-signed certificates.”

Statements (4)

“We use self-signed certificates for testing purposes and the easiest way to make them working is to remove certificate validation. Somehow we must have forgotten to remove that code again when we released our app.“

Developer Survey Summary

Leibniz Universität Hannover

Self-Signed Certificates – Development.

Developers commonly wish to use self-signed certificates for testing purposes and hence want to turn off certificate validation during testing.

Self-Signed Certificates – Production.

A few developers wanted to use self-signed certificates in their production app for cost and effort reasons.

Code Complexity.

Developers described the code-level customization features of SSL as too complex and requiring too much effort.

Certificate Pinning / Trusted Roots.

Developers liked the idea of having an easy way to limit the number of trusted certificates and/or certificate authorities.

Global Warning Message.

Developers requested global SSL warning messages since they described building their own warning messages as too challenging.

Stop-gap Take-Aways

Broken SSL certificate validation is a serious threat for Android Apps

all kinds of apps are affected

root causes are very diverse

MalloDroid is a tool to find apps that need further inspection

checking apps for broken SSL certificate validation during security audits is highly recommended!

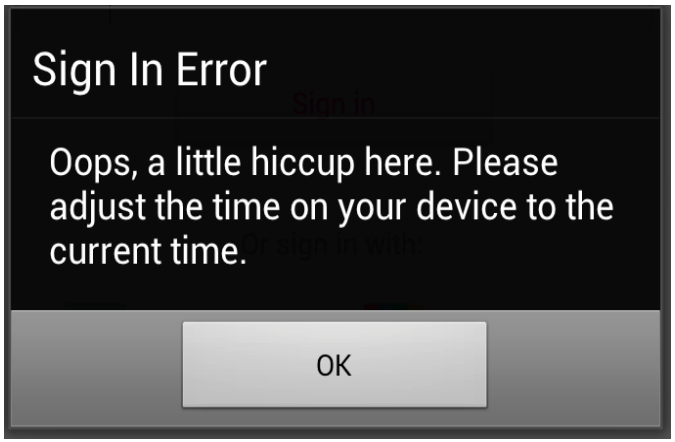
Simply type:

./mallodroid.py –f AppToCheck.apk –d ./javaout

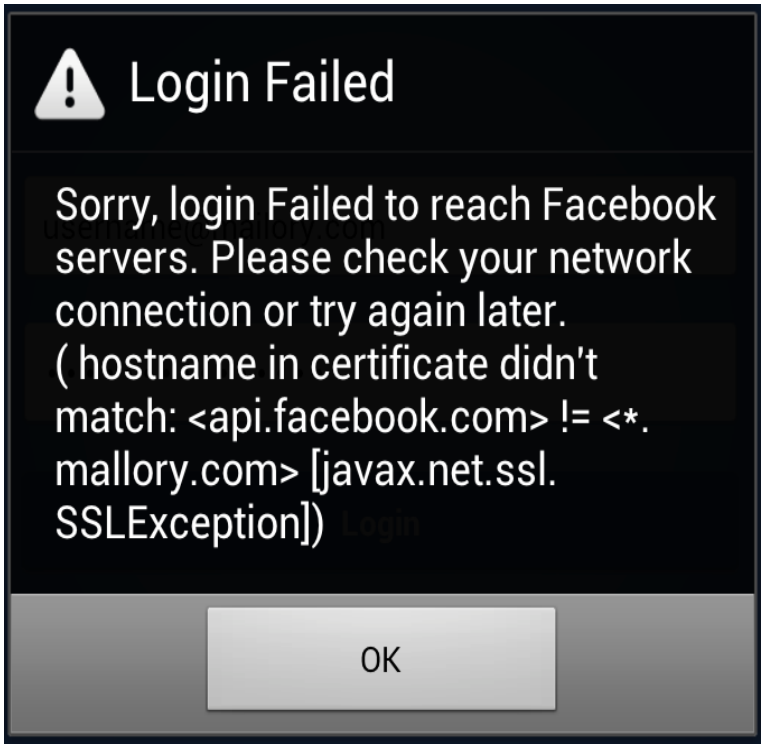
available on github (https://github.com/sfahl/mallodroid)

How Do (Good) Apps React to MITMAS

- Technically they do not endanger the user
- However they suffer from serious usability problems



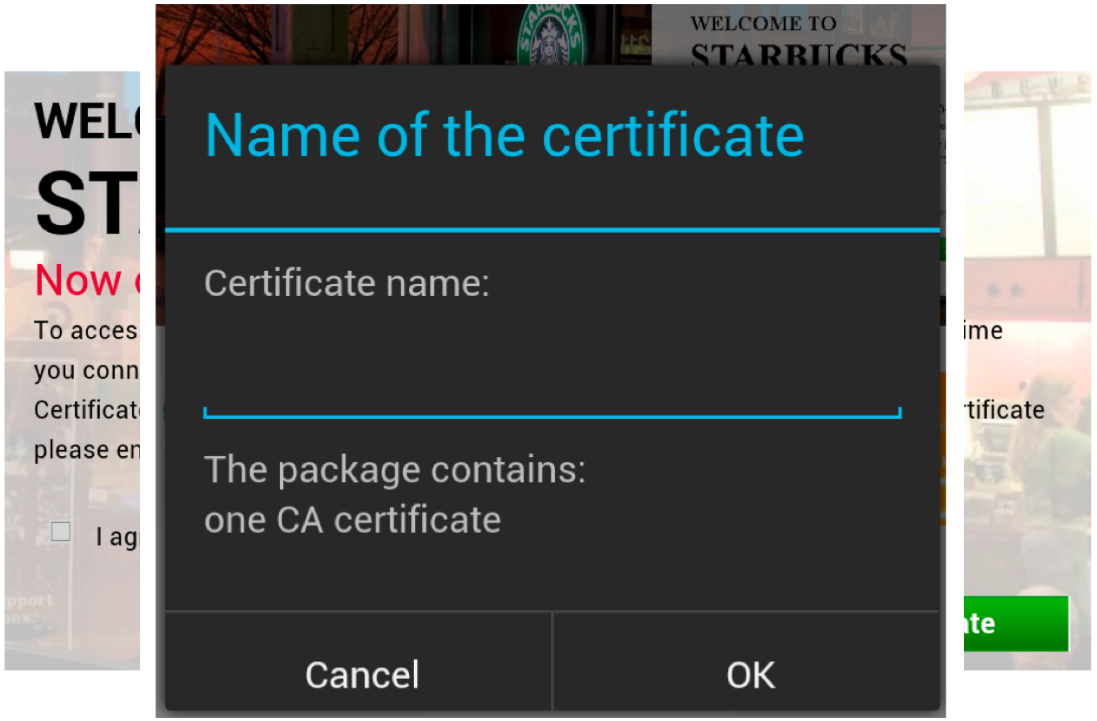
Flickr



Facebook

What if all the code is ok?

- If you don't have the resources to compromise a CA?
  - create your own and get the user to trust it
  - Android dialog to add trusted root CA has usability issues
- We conducted an Amazon MTurk user study on "Usability of Starbucks free Wi-Fi"
  - show fake captive portal
  - when users clicked "Connect and add" fake Android cert dialog is shown
  - no real actions were taken, however there was no way for a user to tell the difference



Trust Injection Attack - Results

- 143 participants completed the MTurk HIT
  - 128 passed the “check questions”
- 73% accepted the CA dialog (and thus made themselves vulnerable)
  - 77% believed that this increased their privacy protection
  - 21% believed there was no change
  - only 2% suspected that their privacy might be at risk
- Of those who clicked cancel
  - 64.7% believed it would not have effected their privacy
  - 29.4% believed it would have improve their privacy
  - only 5.9% thought it would have a negative effect.
- Usability
  - SUS usability score of 76.97 (out of 100)
- Complexity of SSL/CAs makes an excellent breeding ground for social engineering attacks.
  - This is a very easy and low risk attack to execute

A new approach to SSL on Android

Central SSL service for Android

Force SSL
- start with https-everywhere list

Force SSL Validation
- cannot be overridden

Self-Signed Certificates
for developer devices

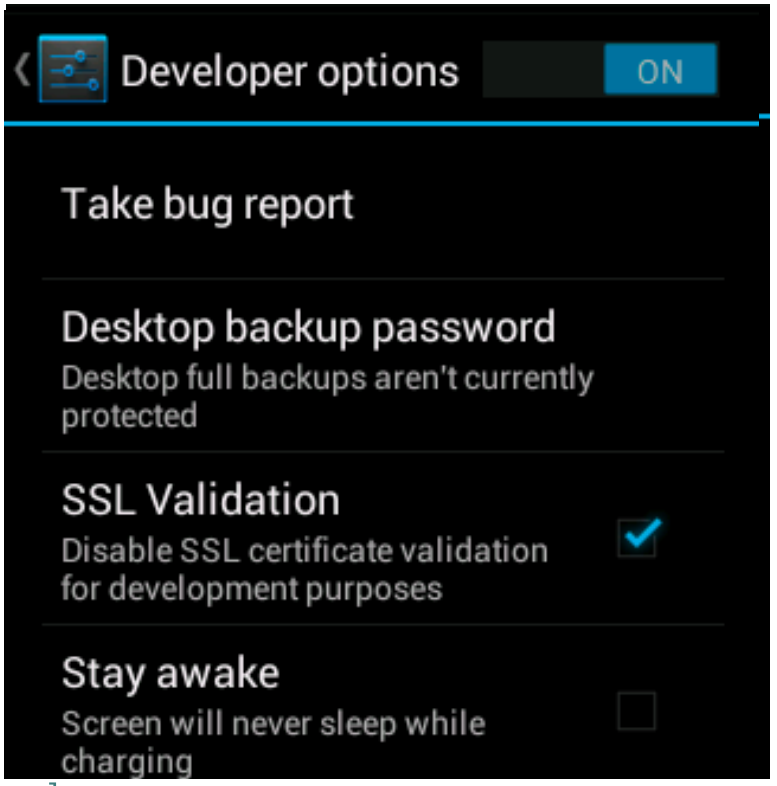
SSL Pinning
via simple config file

Standardised User Interaction
actually show user what is happening

show meaningful warnings

Alternate SSL Validation Strategies
Perspectives, Certificate Transparency, etc.

hot-pluggable




```
<uses-ssl>
    <pins host="securessl.com">
        <pin type="ca" comment="Verisign Root CA">
            8F:57:5A:C8:5B:09:63:B0:24:2B:90...
        </pin>
        <pin type="cert" comment="Self-Signed">
            18:DA:D1:9E:26:7D:E8:BB:4A:21:58...
        </pin>
    </pins>
</uses-ssl>
```

Do you want to know more?

Countermeasures will be presented at CCS in Berlin!

20th ACM Conference on Computer and Communications Security