Mobile Security

14-829 – Fall 2013

Yuan Tian

Class #25 – Security Misuse in Mobile

Carnegie Mellon University
Silicon Valley

Outline

Misuse of SSL in mobile development

Misuse of encryption in mobile development

Misuse of SSL in Mobile

[1]Fahl, Sascha, et al. "Why Eve and Mallory love Android: An analysis of Android SSL (in) security." *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012.

[2] Fahl, Sascha, et al. "Rethinking SSL development in an appified world."*Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013.

Background

SSL is widely used in non-browser software whenever a secure Internet connection is needed

Examples:

(1) sending local data to cloud-based storage

(2) sending customers' payment details from E-Commerce servers to payment processors (ex. PayPal and Amazon)

(3) logging IM clients into online services

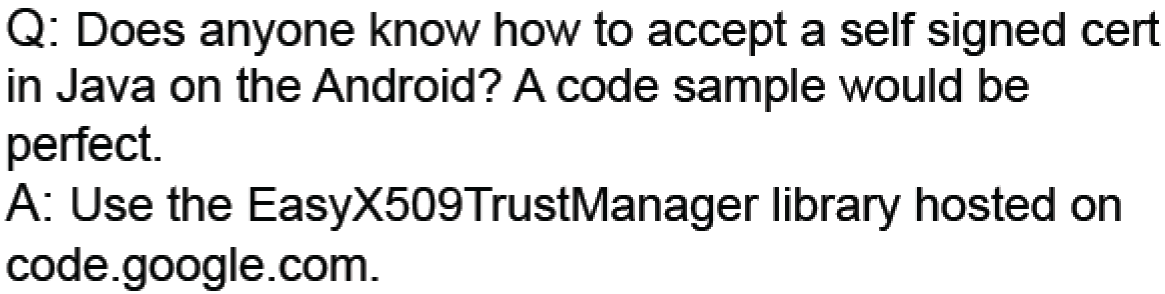
(4) authenticating servers to mobile applications on Android and iOS.

SSL Usage on Android

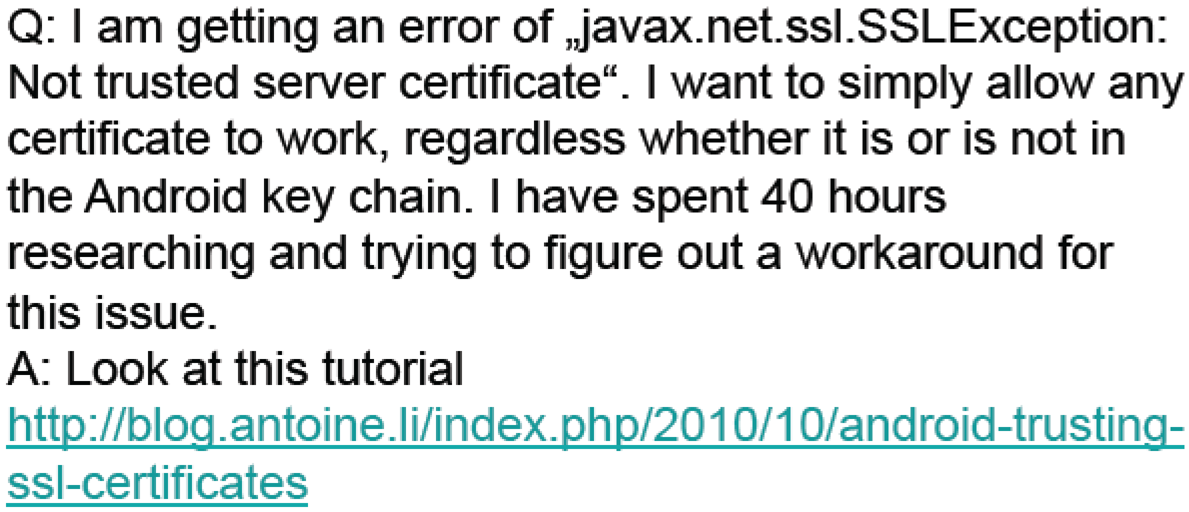
A server needs a certificate which is signed by a trusted party

For non-trusted certificate, a workaround is needed

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.

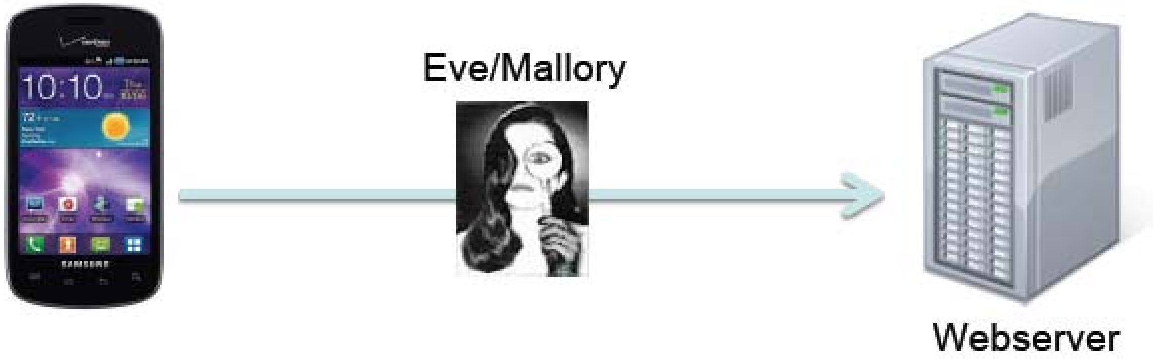


Q: I am getting an error of „javax.net.ssl.SSLException: Not trusted server certificate“. I want to simply allow any certificate to work, regardless whether it is or is not in the Android key chain. I have spent 40 hours researching and trying to figure out a workaround for this issue.
A: Look at this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

Analysis of Misuse

Static Analysis for possible problems:

- Broken TrustManager Implements
- Accept all hostnames



Slides from Sascha Fahl

Analysis Result

Out of 13500 popular and free apps in Google Play, 17.28% of Apps which use SSL fails to Verify the certificate:

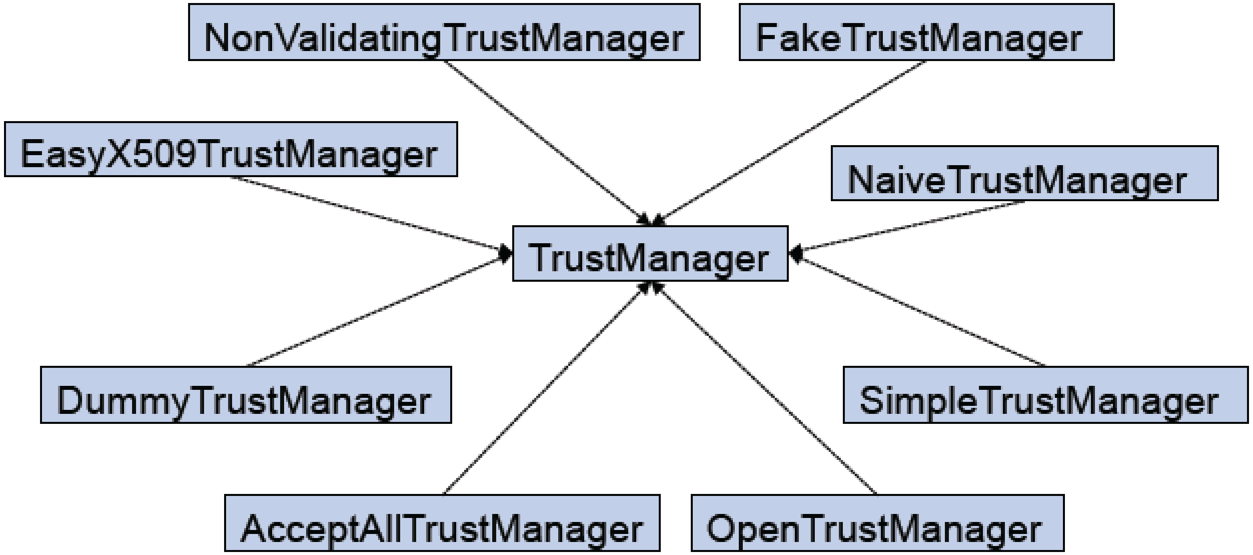
1070 include critical code

790 accept all certificates

284 accept all hostnames

Trust Manager Implementation

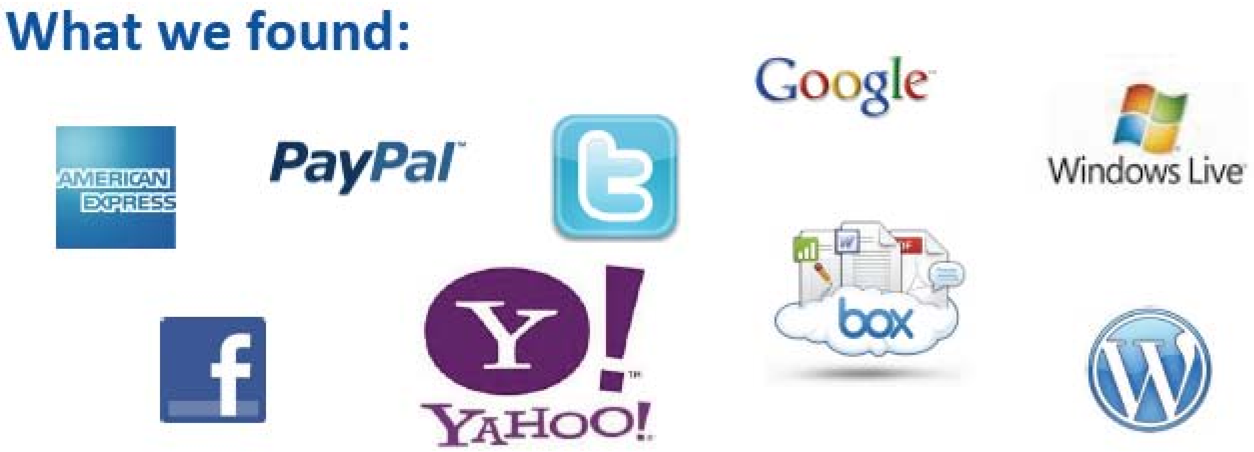
All 22 implementations of trust manager, all fails for effective certificate verification



Slides from Sascha Fahl

Carnegie Mellon University
Silicon Valley

Affected Apps



What we found:

Slides from Sascha Fahl

Case Study- Zoner AV

Anti-Virus App for Android

Award as one of the best AV for app for Android by av-test.org

Zoner AV

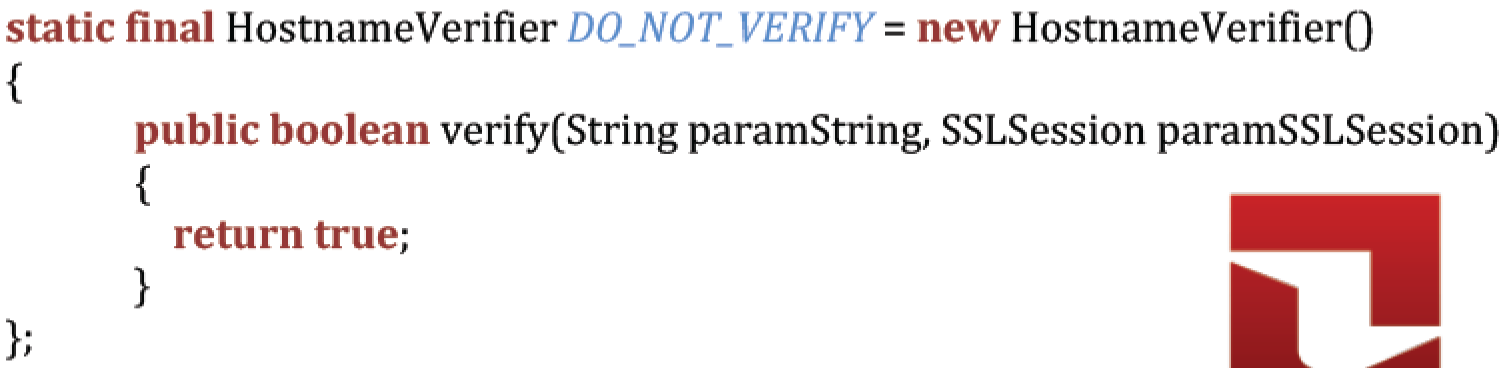
AV TEST
The Independent IT-Security Institute
Magdeburg Germany

Slides from Sascha Fahl

Zoner AV

Virus signature update by HTTPS

No check for the authenticity!

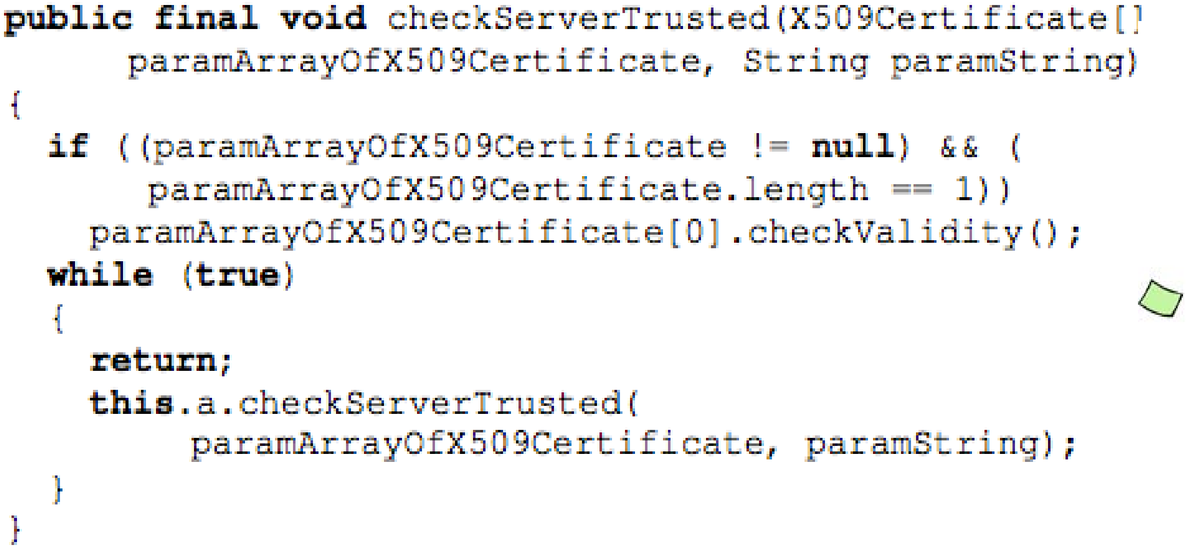


```
static final HostnameVerifier DO_NOT_VERIFY = new HostnameVerifier()
{
        public boolean verify(String paramString, SSLSession paramSSLSession)
        {
          return true;
        }
};
```

Slides from Sascha Fahl

Case Study- Chase

Allows a network attacker to capture username and password and rest of session customer using the app



```
public final void checkServerTrusted(X509Certificate[]
      paramArrayOfX509Certificate, String paramString)
{
  if ((paramArrayOfX509Certificate != null) && (
       paramArrayOfX509Certificate.length == 1))
    paramArrayOfX509Certificate[0].checkValidity();
  while (true)
  {
    return;
    this.a.checkServerTrusted(
         paramArrayOfX509Certificate, paramString);
  }
}
```

How to Fix the Mess?

It’s all developers’ fault!

Slides from Sascha Fahl

Why this is wrong-
Talk to Developers

The author contacted 80 developers of broken apps

- informed them
- offered further assistance
- asked them for an interview

Slides from Sascha Fahl

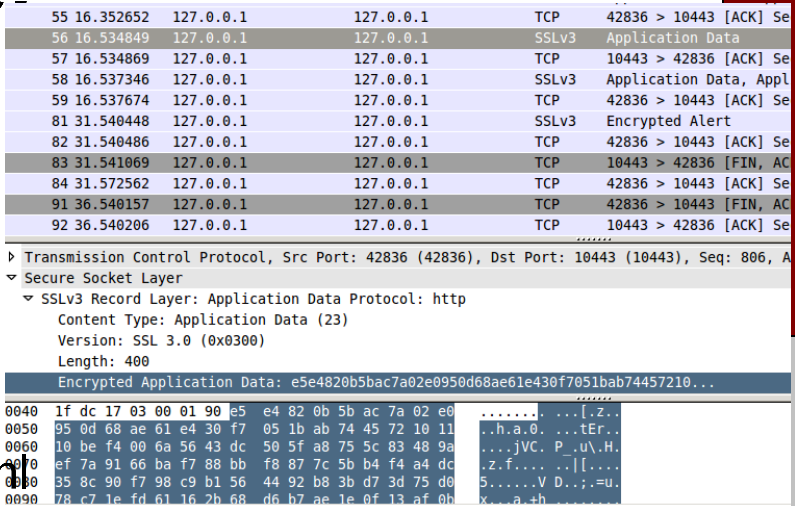
Statement 1

*“**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.**”*

Slides from Sascha Fahl

Statement 2

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.***



```
   55 16.352652   127.0.0.1           127.0.0.1           TCP      42836 > 10443 [ACK] Se
   56 16.534849   127.0.0.1           127.0.0.1           SSLv3    Application Data
   57 16.534869   127.0.0.1           127.0.0.1           TCP      10443 > 42836 [ACK] Se
   58 16.537346   127.0.0.1           127.0.0.1           SSLv3    Application Data, Appl
   59 16.537674   127.0.0.1           127.0.0.1           TCP      42836 > 10443 [ACK] Se
   81 31.540448   127.0.0.1           127.0.0.1           SSLv3    Encrypted Alert
   82 31.540486   127.0.0.1           127.0.0.1           TCP      42836 > 10443 [ACK] Se
   83 31.541069   127.0.0.1           127.0.0.1           TCP      10443 > 42836 [FIN, AC
   84 31.572562   127.0.0.1           127.0.0.1           TCP      42836 > 10443 [ACK] Se
   91 36.540157   127.0.0.1           127.0.0.1           TCP      42836 > 10443 [FIN, AC
   92 36.540206   127.0.0.1           127.0.0.1           TCP      10443 > 42836 [ACK] Se

▷ Transmission Control Protocol, Src Port: 42836 (42836), Dst Port: 10443 (10443), Seq: 806, A
▽ Secure Socket Layer
  ▽ SSLv3 Record Layer: Application Data Protocol: http
      Content Type: Application Data (23)
      Version: SSL 3.0 (0x0300)
      Length: 400
      Encrypted Application Data: e5e4820b5bac7a02e0950d68ae61e430f7051bab74457210...

0040  1f dc 17 03 00 01 90 e5  e4 82 0b 5b ac 7a 02 e0   ........ ...[.z..
0050  95 0d 68 ae 61 e4 30 f7  05 1b ab 74 45 72 10 11   ..h.a.0. ...tEr..
0060  10 be f4 00 6a 56 43 dc  50 5f a8 75 5c 83 48 9a   ....jVC. P_.u\.H.
0070  ef 7a 91 66 ba f7 88 bb  f8 87 7c 5b b4 f4 a4 dc   .z.f.... ..|[....
0080  35 8c 90 f7 98 c9 b1 56  44 92 b8 3b d7 3d 75 d0   5......V D..;.=u.
0090  78 c7 1e fd 61 16 2b 68  d6 b7 ae 1e 0f 13 af 0b   x...a.+h ........
```

Slides from Sascha Fahl

Statement 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****."*

Slides from Sascha Fahl

Statement 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****.“*

Copyright © Ron Leishman * http://ToonClips.com/6768

Slides from Sascha Fahl

Developer’s Wish list

Self-Signed Certificates – Development

Self-Signed Certificates – Production

Less SSL Coding

Certificate Pinning / Trusted Roots

Easy-to-use Warning Message

Slides from Sascha Fahl

The Dilemma

Current Situation:

Developers have the freedom to customize certificate validation

Developers mostly are not security experts

Developers find the current situation too inflexible

Future Situation:

Protect the user!

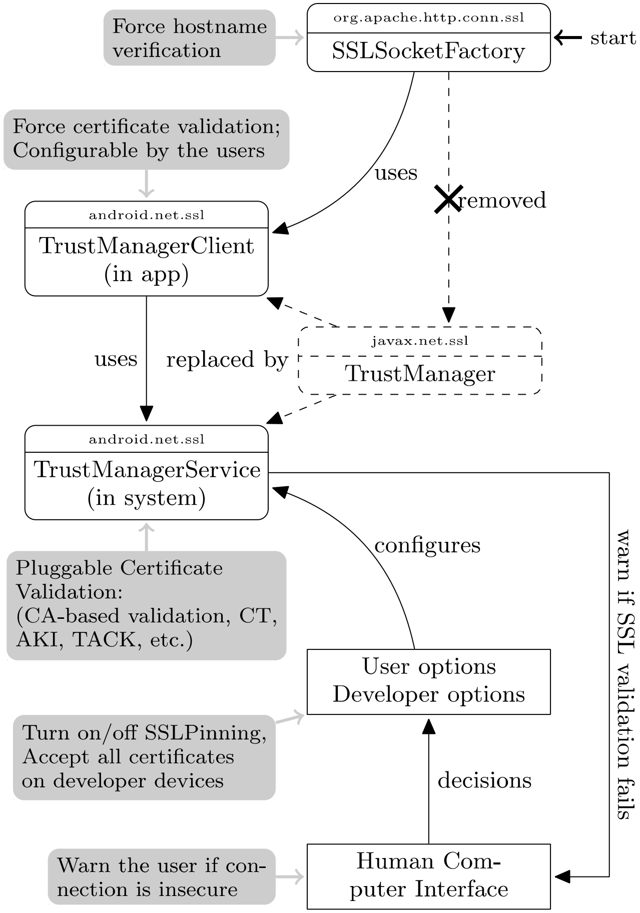
Make the common use cases easy

Adapt certificate handling to the developers' needs

Solution: Improve usability of certificate handling for developers!

Slides from Sascha Fahl

Patching Android OS



Force hostname verification

org.apache.http.conn.ssl

SSLSocketFactory

start

Force certificate validation; Configurable by the users

uses

removed

android.net.ssl

TrustManagerClient (in app)

uses

replaced by

javax.net.ssl

TrustManager

android.net.ssl

TrustManagerService (in system)

Pluggable Certificate Validation: (CA-based validation, CT, AKI, TACK, etc.)

configures

warn if SSL validation fails

User options
Developer options

Turn on/off SSLPinning, Accept all certificates on developer devices

decisions

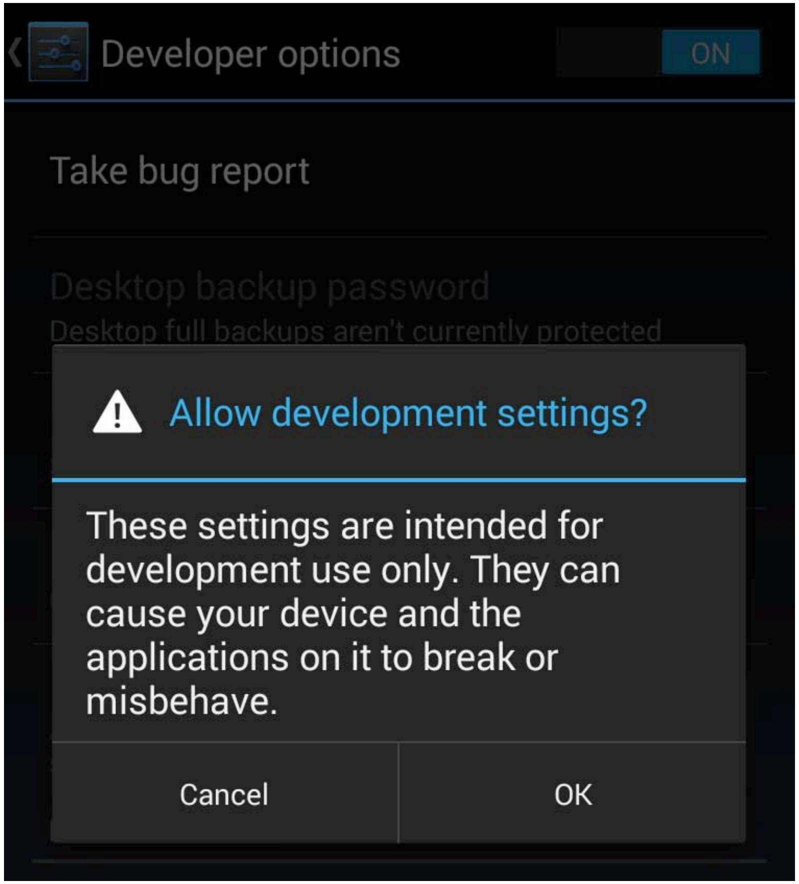
Warn the user if connection is insecure

Human Computer Interface

Slides from Sascha Fahl

Self-signed Certificate

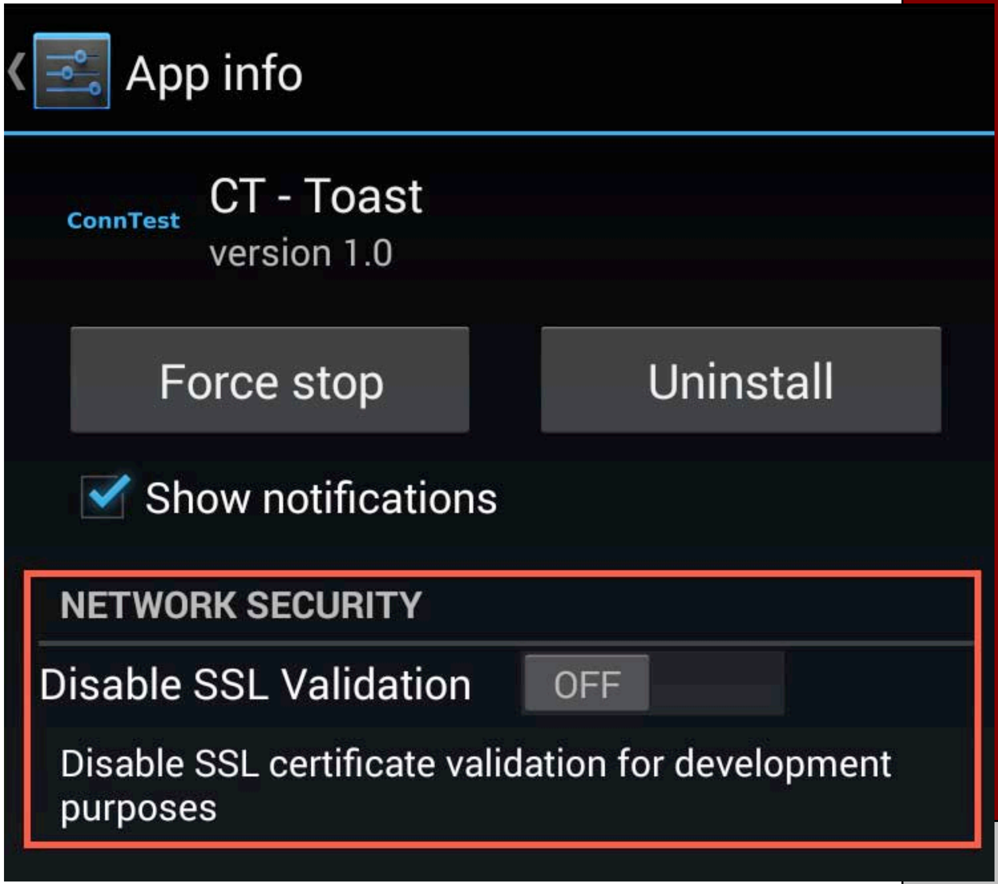
enable developer options



Developer options ON

⚠ Allow development settings?

These settings are intended for development use only. They can cause your device and the applications on it to break or misbehave.



CT - Toast
version 1.0

☑ Show notifications

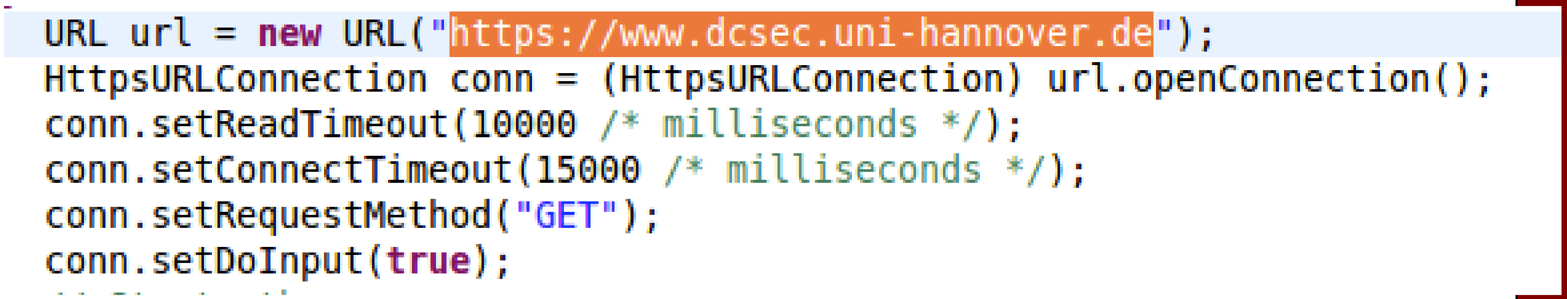
NETWORK SECURITY

Disable SSL Validation OFF

Disable SSL certificate validation for development purposes

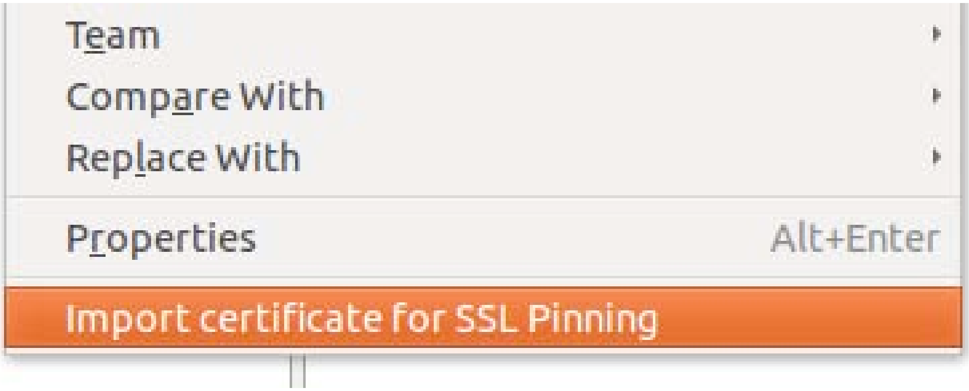
Slides from Sascha Fahl

Certificate Pining



```
URL url = new URL("https://www.dcsec.uni-hannover.de");
HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();
conn.setReadTimeout(10000 /* milliseconds */);
conn.setConnectTimeout(15000 /* milliseconds */);
conn.setRequestMethod("GET");
conn.setDoInput(true);
```

This is easy!



Slides from Sascha Fahl

Conclusion

- ✔ Eve and Mallory no longer love Android
- ✔ Backwards compatible – no broken apps, except
  - ✘ apps that implemented pinning (19 in 13500 tested Android apps)
  - ✔ updating them to the new pinning sytem is very easy
- ✔ New features for Android
  - ✔ Easy to use self-signed certs for development
  - ✔ Easy to use pinning / custom CAs
  - ✔ Central and easy to use warning messages
  - ✔ Central place to plug in new validation strategies – such as CT, TACK, etc
- ✔ Contacted developers –
  - ✔ got positive feedback

Slides from Sascha Fahl

Misuse of Encryption in Mobile

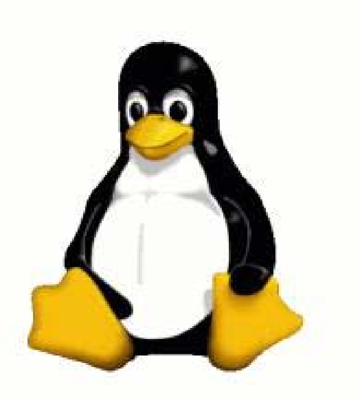
[3]Egele, Manuel, et al. "An empirical study of cryptographic misuse in android applications." *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013.

Motivation

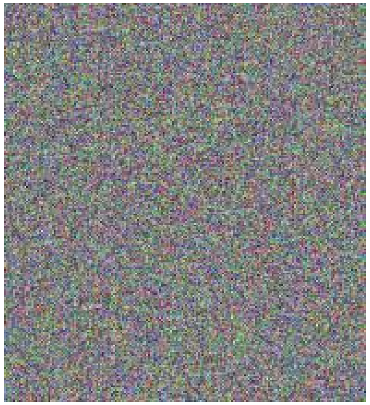
> 800,000 Android applications

Apps handle sensitive information (e.g., passwords)

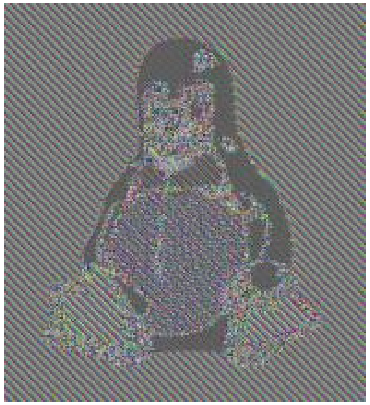
Developers are not security experts



Plaintext



AES/CBC



AES/ECB

Slides from Manuel Egele

Crypto APIs in Android

Cryptographic service providers (CSP) are interfaces to:

- (A-) symmetric crypto
- MAC algorithms
- Key generation
- TLS, OpenPGP, etc.

Android uses BouncyCastle as CSP

BouncyCastle is compatible to Java Sun JCP

Slides from Manuel Egele

Commonly Used Crypto Primitives

Symmetric encryption schemes

IND-CPA

- Block ciphers: AES/[3]DES
- Encryption modes: ECB/CBC/CTR

Password-based encryption

Cracking resistance

- Deriving key material from user passwords

Pseudo random number generators

Secure seed

- Random seed

Slides from Manuel Egele

Common Rules

1) Do not use ECB mode for encryption
2) Do not use a static IV for CBC mode
3) Do not use constant symmetric encryption keys
4) Do not use constant salts for PBE
5) Do not use fewer than 1,000 iterations for PBE
6) Do not use static seeds to seed `SecureRandom()`

Slides from Manuel Egele

Cryptolint

Static program analysis techniques

1. Extract a super control flow graph from app
2. Identify calls to cryptographic APIs
3. Static backward slicing to evaluate security rules

Automatically detect if developers do not use crypto correctly!

Slides from Manuel Egele

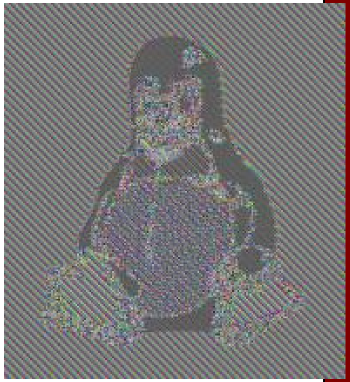
Rule 1: Thou Shalt Not Use ECB

Transformation string specifies:

Algorithm

Block Cipher Mode (optional)

Padding (optional)



```
Cipher.getInstance("AES/ECB/PKCS7Padding", "BC");
```

Default for block ciphers: ECB (undocumented)

Problem: Bad defaults

Carnegie Mellon University
Silicon Valley

Slides from Manuel Egele

Rule 2: Thou Shall Use Random IVs

CBC$ algorithm specifies random IV

```
c = Cipher.getInstance(“AES/CBC/PKCS7Padding”);
c.getIV();
```

Developer can specify IV herself

```
public final void init (int opmode, Key key,
   AlgorithmParameterSpec params)
IvParameterSpec(byte[] iv)
```

Problem: Insufficient Documentation

Carnegie Mellon University
Silicon Valley

Slides from Manuel Egele

Rule 3: Thou Shalt Not Use Static Symmetric Encryption Keys

Key embedded in application ⇒ not secret

Symmetric encryption schemes often specify a randomized key generation function

To instantiate a key object:

```
SecretKeySpec(byte[] key, String algorithm)
```

Problem: Developer Understanding

Carnegie Mellon University
Silicon Valley

Slides from Manuel Egele

Rule 4: Thou Shalt Not Use Constant Salts for Password Based Encryption

RFC2898 (PKCS#5):

"*4.1 Salt … producing a large set of keys … one is selected at random according to the salt.*"

```
PBEParameterSpec(byte[] salt,
   int iterationCount)
```

Problem: Poor Documentation

Slides from Manuel Egele

Rule 5: Thou Shalt Not Use Small Iteration Counts for PBE

RFC2898 (PKCS#5):

> “*4.2 Iteration Count: For the methods in this document, a minimum of 1,000 iterations is recommended.*”

```
PBEParameterSpec(byte[] salt,
   int iterationCount)
```

Problem: Poor Documentation

Rule 6: Thou Shalt not Seed `SecureRandom()` With Static Values

Android documentation for `SecureRandom()` PRNG:

> "*This class generates cryptographically secure pseudo-random numbers. It is best to invoke `SecureRandom` using the default constructor.*"

...

> "*Seeding `SecureRandom` may be insecure*"

`SecureRandom()` vs. `SecureRandom(byte[] seed)`

Problem: Developer Understanding

Carnegie Mellon University
Silicon Valley

Slides from Manuel Egele

Evaluation

145,095 Apps downloaded from Google Play

Only Apps that use

- `javax/crypto`
- `java/security`
- Filter popular libraries (advertising, statistics, etc.)

11,748 Apps analyzed

Slides from Manuel Egele

Evaluation

11,748 apps use crypto

13% use static salt for passwords

65% use ECB

31% use static symmetric key

13% use small iteration counts

16% use known IV for CBC

14% misuse `SecureRandom`

Slides from Manuel Egele

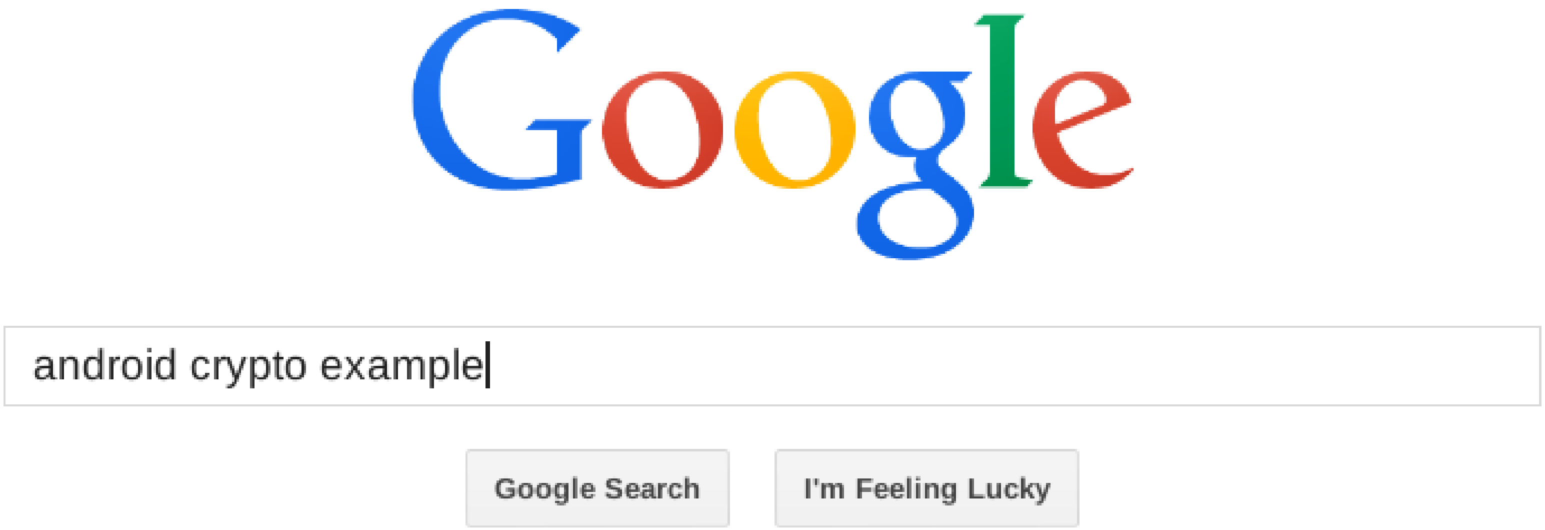
Password Manager (+6 days)

```
private String encrypt(byte [] key, String clear) {
  byte [] encrypted;
  byte [] salt = new byte[2];
  ...
  Random rnd = new Random();
  Cipher cipher =
    Cipher.getInstance("AES/CBC/PKCS7Padding", "BC");
  byte [] iv = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
  IvParameterSpec ivSpec = new IvParameterSpec(iv);
  cipher.init(Cipher.ENCRYPT_MODE, skeySpec, ivSpec);
  rnd.nextBytes(salt);
  cipher.update(salt);
  encrypted = cipher.doFinal(clear.getBytes());
```

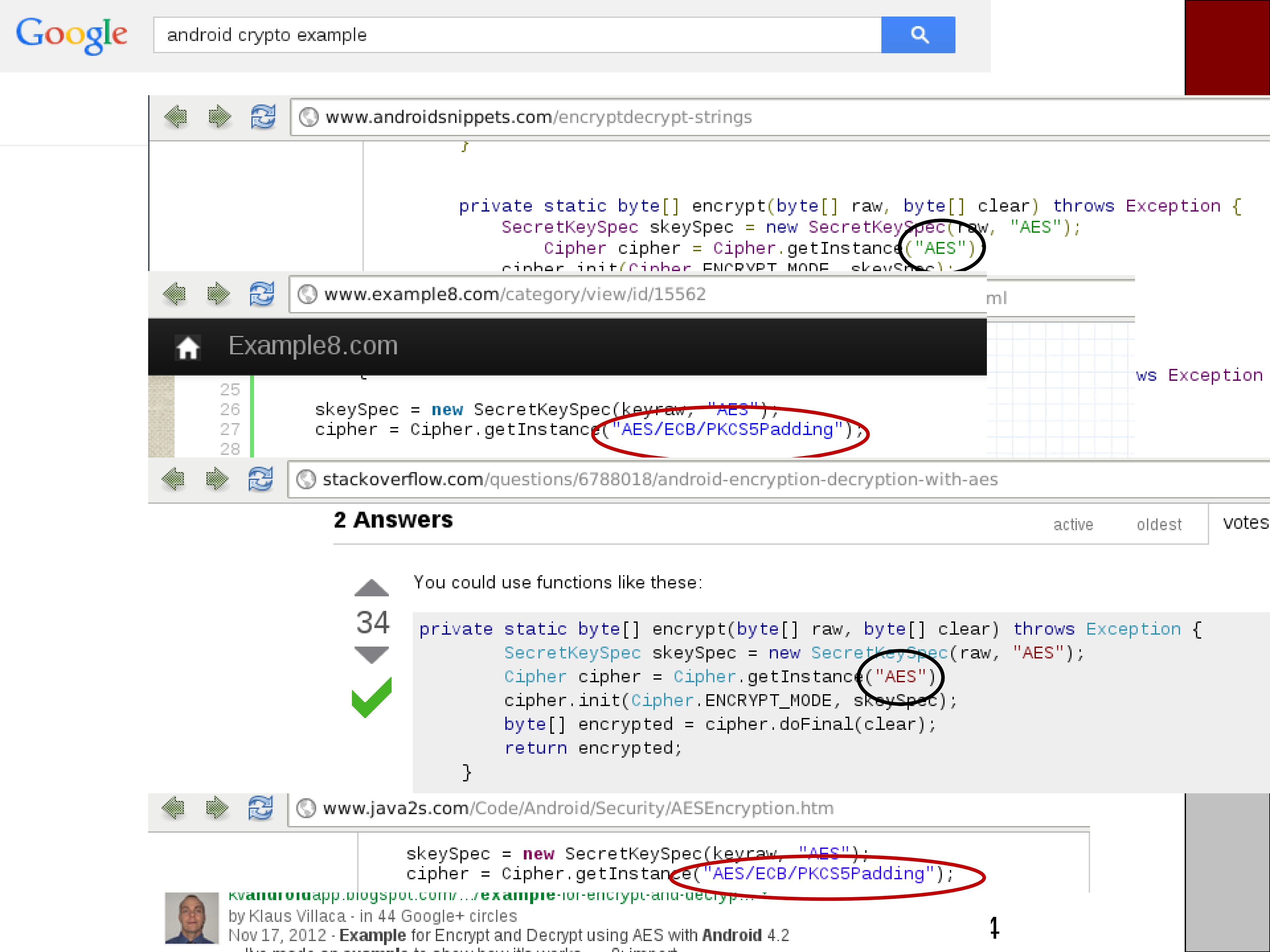
Password Manager (key)

```
public static byte []
hmacFromPassword(String password) {
  byte [] key = null;
  ...
  Mac hmac = Mac.getInstance("HmacSHA256");
  hmac.init (new SecretKeySpec
    ("notverysecretiv".getBytes("UTF-8"), "RAW"));
  hmac.update(password.getBytes("UTF-8"));
  key = hmac.doFinal();
  ...
  return key;
```

How Do Developers Learn Crypto?



android crypto example



android crypto example

www.androidsnippets.com/encryptdecrypt-strings

```
    }

    private static byte[] encrypt(byte[] raw, byte[] clear) throws Exception {
        SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
            Cipher cipher = Cipher.getInstance("AES");
        cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
```

www.example8.com/category/view/id/15562

Example8.com

```
        {
    skeySpec = new SecretKeySpec(keyraw, "AES");
    cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
```

ml

ws Exception

stackoverflow.com/questions/6788018/android-encryption-decryption-with-aes

2 Answers

active oldest votes

34

You could use functions like these:

```
private static byte[] encrypt(byte[] raw, byte[] clear) throws Exception {
        SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
        Cipher cipher = Cipher.getInstance("AES");
        cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
        byte[] encrypted = cipher.doFinal(clear);
        return encrypted;
    }
```

www.java2s.com/Code/Android/Security/AESEncryption.htm

```
    skeySpec = new SecretKeySpec(keyraw, "AES");
    cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
```

kvandroidapp.blogspot.com/.../example-for-encrypt-and-decryp...

by Klaus Villaca - in 44 Google+ circles

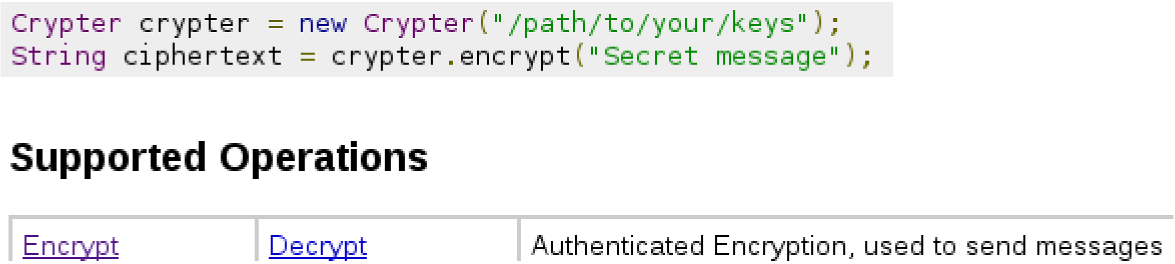
Nov 17, 2012 - **Example** for Encrypt and Decrypt using AES with **Android** 4.2

I've made an example to show how it's works ... 0; import

1

keyCzar

“Developers should not be able to inadvertently expose key material, use weak key lengths or deprecated algorithms, or improperly use cryptographic modes.”



```
Crypter crypter = new Crypter("/path/to/your/keys");
String ciphertext = crypter.encrypt("Secret message");
```

Supported Operations

Encrypt	Decrypt	Authenticated Encryption, used to send messages

Conclusions

Developers are not security or crypto experts

It is too easy to use crypto incorrectly

- Bad default values
- Lacking documentation
- Developer misunderstanding

Improved APIs & Documentation necessary

- Authenticated encryption (e.g., GCM)
- Security discussion for crypto APIs