Advanced Android Security APIs

- KeyStore and Crypto
- VPN
**KeyChain and Crypto APIs**

- Like any other OS: support for crypto operations
  - SecureRandom: generate cryptographically secure random data
    - E.g., seeding key generation, etc.
  - KeyGenerators and key factories: generate key for selected algorithms
  - Password-based encryption support and key derivation
    - PKCS#5 and PKCS#12

- Supports unified key storage: Keystore and keychain
  - Keystore: app-private credentials; keychain: system-wide credentials
  - Uses platform hardware-support for storage (if platform supports) and allows binding material to the platform
    - Trusted Execution Environment (TEE), Secure Element (SE)
  - Allows using key material without need to release it to the app process
  - Specifying key usage authorization
  - Key attestation can be used to verify key is stored in hardware (locally or to remote party)
Additional feature for user authentication that can be used in conjunction with the keystore system:

- Confirm credentials:
  - authenticate users based on how recently they last unlocked their device
  - Only if (re-)authenticated credentials in keystore can be used
- Fingerprint authentication
  - User can enroll fingerprints
  - Fingerprint must authorize a specific cryptographic operation associated with one key

Both give a kind of “physical presence” of the user
PROBLEMS WHEN USING CRYPTO API [66]

- Using cryptographic primitives in the wrong way can undermine the strong security guarantees crypto can offer
  - Using crypto the right is not always obvious and can be very hard
PROBLEMS WHEN USING CRYPTO APIs [66]

- Rule 1: Do not use ECB mode for encryption
  - Identical message -> identical cipher text
- Rule 2: Do not use a non-random IV for CBC encryption
  - constant IV results in a deterministic, stateless cipher (not IND-CPA)
- Rule 3: Do not use constant encryption keys
  - constant key hard-coded in publicly available software is not a secret key
- Rule 4: Do not use constant salts for PBE
  - Reduces to a PBE with no salt at all
- Rule 5: Do not use fewer than 1,000 iterations for PBE
  - More susceptible to brute-force attacks
- Rule 6: Do not use static seeds to seed SecureRandom()
  - PRNG seeded with a constant seed will produce a constant, known output across all implementations
PROBLEMS WHEN USING CRYPTO APIs [66]

- Study of 11,748 Android applications that make use of the crypto APIs
  - 10,327 programs (88%) use cryptography inappropriately

<table>
<thead>
<tr>
<th># apps</th>
<th>violated rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>5,656</td>
<td>Uses ECB (BouncyCastle default) (R1)</td>
</tr>
<tr>
<td>3,644</td>
<td>Uses constant symmetric key (R3)</td>
</tr>
<tr>
<td>2,000</td>
<td>Uses ECB (Explicit use) (R1)</td>
</tr>
<tr>
<td>1,932</td>
<td>Uses constant IV (R2)</td>
</tr>
<tr>
<td>1,636</td>
<td>Used iteration count &lt; 1,000 for PBE(R5)</td>
</tr>
<tr>
<td>1,629</td>
<td>Seeds SecureRandom with static (R6)</td>
</tr>
<tr>
<td>1,574</td>
<td>Uses static salt for PBE (R4)</td>
</tr>
<tr>
<td>1,421</td>
<td>No violation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># apps</th>
<th>rules violated</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,905</td>
<td>Rule 1 &amp; Rule 3</td>
</tr>
<tr>
<td>1,588</td>
<td>Rule 1 &amp; Rule 6</td>
</tr>
<tr>
<td>1,247</td>
<td>Rule 4 &amp; Rule 5</td>
</tr>
<tr>
<td>866</td>
<td>Rule 2 &amp; Rule 3</td>
</tr>
<tr>
<td>109</td>
<td>Rule 1 &amp; Rule 2</td>
</tr>
<tr>
<td>24</td>
<td>Rule 1 &amp; Rule 5</td>
</tr>
<tr>
<td>11</td>
<td>Rule 3 &amp; Rule 5</td>
</tr>
<tr>
<td>5</td>
<td>Rule 2 &amp; Rule 5</td>
</tr>
<tr>
<td>2</td>
<td>Rule 1 &amp; Rule 4</td>
</tr>
<tr>
<td>2</td>
<td>Rule 3 &amp; Rule 4</td>
</tr>
</tbody>
</table>
PROBLEMS WHEN USING CRYPTO APIs [66]

- **Examples Rule 1 (ECB):**
  - DES/ECB to encrypt personal identifiers that exceed the DES block size -> one can learn identical identifiers
  - One password manager stores passwords encrypted using AES in ECB mode

- **Example Rule 3 (const. enc. key):**
  - AdMob encrypts the phone’s location data using a constant key and sends it over the network
  - Apps storing the user’s Google credentials on disk encrypted using a static key
  - One game app with 100M installs for encrypting communication with backend servers
Libraries as attributing factor [62]

- Checked Ad libs for cryptographic misuse

<table>
<thead>
<tr>
<th>Property</th>
<th>#Libs/Ver</th>
<th>Verified</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1: ECB mode for encryption</td>
<td>5/25</td>
<td>5/25</td>
</tr>
<tr>
<td>R2: constant IV for CBC</td>
<td>7/32</td>
<td>4/20</td>
</tr>
<tr>
<td>R3: constant symmetric keys</td>
<td>13/60</td>
<td>3/7</td>
</tr>
<tr>
<td>R4: static salts for PBE</td>
<td>2/2</td>
<td>2/2</td>
</tr>
<tr>
<td>R5: &lt;1000 iterations for PBE</td>
<td>2/2</td>
<td>2/2</td>
</tr>
<tr>
<td>R6: static seed for SecureRandom</td>
<td>3/7</td>
<td>2/5</td>
</tr>
</tbody>
</table>

- In total 2,667 app versions of 296 distinct apps with a cumulative install-base of 3.7bn were affected by those ad libs with verified crypto misuse
- Android comes with an on-board VPN service

- Since Android 4.0: Application-based VPNs using the VpnService API
Application-based VPNs

- **VpnService**: Default Use-case

  - **VpnService.Builder**:
    - Allows selection of that can/cannot use the VPN tunnel (e.g., protect Intranet at endpoint)
      - `addAllowedApplication`, `addDisallowedApplication`
“Abusing” VPN API

- Traffic inspection / injection
  - Malicious intentions (e.g., stealing credentials, injection malware,...)
  - Detecting privacy leaks [67]
- Ad filter / application firewall
  - Filter packages that connect to or come from well-known advertisement services
    - BSc thesis in the InfSec group (2013)
Course Project

- Topic
- Team building
- Scoreboard
Course Project

Topic
### UdS Chipper App

- **Very simple Twitter-like app:**
  - Users can sign up, post messages, manage their account

![Diagram showing Chipper Android App (Sign up/in, post/view/delete message) connected to Chipper Server (Message dashboard, RESTful API)]
**YOUR TASK**

- Goal of the project is to identify, exploit, fix security vulnerabilities of the Chipper App (not the server!)
  - All attacks are based on what you have seen during the first lecture week
    - But: May depend on each other (i.e., Vuln1 needed to come to Vuln2)
    - Project description will give you guidance which categories of vulnerabilities we build on purpose
      - Vulnerabilities are either implementation errors or inherent to the Android system
  - Develop for each discovered vulnerability a proof-of-concept and propose (not implement!) a bug fix with short explanation why this is a fix
  - All means allowed to discover the vulnerabilities
    - Reverse engineering, static/dynamic analysis, Mitm attacks, attack apps,...
Course Project

Team Building
Course Project

Scoreboard
**Gamification of project**

- Small competition between teams:
  - Entirely voluntarily! (Just for fun & games!)
  - Submit your exploit code by email when you have it to earn points
    - Earlier submission than other teams = more points
  - Submit your bug fixes when you have them to earn points
    - Earlier submission than other teams = more points

- Small awards to be gained at end of project period for top teams