
Marten Oltrogge, Nicolas Huaman, Sabrina Amft
Yasemin Acar, Michael Backes, Sascha Fahl
Android and TLS - Prior Research

Long History of research on the urgent state of custom TLS in Android apps rendering them vulnerable for MITMAs

- FAHL et al. Why Eve and Mallory love Android: An analysis of Android SSL (in)security [54]
- FAHL et al. Rethinking SSL Development in Appified World [56]
- GEORGIEV et al. The most dangerous code in the world: validating SSL certificates in non-browser software [58]
- CHOTHIA et al. Why banker bob (still) can't get tls right: A security analysis of tls in leading uk banking apps [47]
- CONTI et al. Mind the hand you shake-protecting mobile devices from SSL usage vulnerabilities [48]
- OLTROGGE et al. To Pin or Not to Pin - Helping App Developers Bullet Proof Their TLS Connections [74]
- ONWUZURIKE et al. Danger is my middle name: experimenting with SSL vulnerabilities in Android apps [77]
- SOUNTHRIRAJU et al. SMV-Hunter: Large scale, automated detection of SSL/TLS man-in-the-middle vulnerabilities in android apps [84]
- TENDULKAR et al. An Application Package Configuration Approach to Mitigating Android SSL Vulnerabilities [85]
- OLTROGGE et al. The rise of the citizen developer: Assessing the security impact of online app generators [75]

Some proposed countermeasures

[56] and [85] propose a declarative configuration-driven approach letting developers implement custom TLS without code

```java
@Override
public boolean verify(
    String host,
    SSLSession session) {
    return true;
}
```

```java
@Override
public void checkServerTrusted(
    X509Certificate[] chain, String authType)
throws CertificateException {
}
```

```java
@Override
public void onReceivedSslError (
    WebView view,
    SslErrorHandler handler,
    SslError error) {
    handler.proceed();
}
```
To address this urgent state, new countermeasures and novel mechanisms have been introduced:

- **Network Security Configuration (NSC)** added in Android 7 to keep developers from implementing possibly insecure custom certificate validation code
- **Safe defaults** to make mounting of MITMAs harder
  - Not trust for user CAs by default in Android 7
  - Enforcing HTTPS by default in Android 9
- **Safeguards** to block apps containing vulnerable certificate validation code

<table>
<thead>
<tr>
<th>Date</th>
<th>Android Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015-10-05</td>
<td>Android 6 (API 21)</td>
<td>Android introduces the “androidUsesClearseTraffic” flag for Manifest files, and removes the Apache HTTP Client library [33, 79, 64].</td>
</tr>
<tr>
<td>2016-05-17</td>
<td></td>
<td>Google Play blocks apps containing unsafe implementations of the X509TrustManager interface [67].</td>
</tr>
<tr>
<td>2016-08-22</td>
<td>Android 7 (API 24)</td>
<td>Android introduces NSC, distrusts user-installed certificates, and ignores the “androidUsesClearseTraffic” flag in case a NSC file is available [44, 94].</td>
</tr>
<tr>
<td>2016-11-25</td>
<td></td>
<td>Google Play blocks apps containing unsafe implementations of the onReceivedSslError method in WebView [66].</td>
</tr>
<tr>
<td>2017-03-01</td>
<td></td>
<td>Google Play blocks apps containing unsafe implementations of the HostnameVerifier interface [68].</td>
</tr>
<tr>
<td>2017-08-21</td>
<td>Android 8 (API 26)</td>
<td>Android adds support for the “clearesitTrafficPermitted” flag for the WebView class [81].</td>
</tr>
<tr>
<td>2018-08-01</td>
<td></td>
<td>New apps need to target at least Android 8; makes new safe defaults introduced with Android 7 (2016-08-22) and Android 8 (2017-08-21) [49, 95] available to those apps.</td>
</tr>
<tr>
<td>2018-08-08</td>
<td>Android 9 (API 28)</td>
<td>Sets “clearesitTrafficPermitted” to false; enforces HTTPS connections by default. Developers can revert this for specific domains or globally in NSC settings [63].</td>
</tr>
<tr>
<td>2018-11-01</td>
<td></td>
<td>App updates need to target at least Android 8; makes new safe defaults introduced with Android 7 (2016-08-22) and Android 8 (2017-08-21) [49, 95] available to existing apps.</td>
</tr>
<tr>
<td>2019-08-01</td>
<td></td>
<td>New apps need to target at least Android 9; makes new safe defaults introduced with Android 9 (2018-08-08) [49, 95] available to those apps.</td>
</tr>
<tr>
<td>2019-11-01</td>
<td></td>
<td>Updates need to target at least Android 9; existing apps benefit from new safe defaults introduced with Android 9 (2018-08-08) [49, 95].</td>
</tr>
</tbody>
</table>

Affects Android OS & NSC – Affects Google Play security policy & safeguards.
Network security configuration - Features

- Allows to implement custom network security behaviour without writing code
  - Allow or disallow cleartext traffic

```xml
<?xml version="1.0" ?>
<network-security-config>
  <base-config cleartextTrafficPermitted="[true|false]"/>
  <domain-config cleartextTrafficPermitted="[true|false]">
    <domain includeSubdomains="true">example.com</domain>
  </domain-config>
</network-security-config>
```
Network security configuration - Features

- Allows to implement custom network security behaviour without writing code
  - Allow or disallow cleartext traffic
  - Configure trust anchors

```xml
<?xml version="1.0" ?>
<network-security-config>
  <domain-config>
    <domain includeSubdomains="true">example.com</domain>
    <trust-anchors>
      <certificates src="system"/>
      <certificates src="user"/>
      <certificates src="@raw/ca"/>
    </trust-anchors>
  </domain-config>
</network-security-config>
```
Network security configuration - Features

- Allows to implement custom network security behaviour without writing code
  - Allow or disallow cleartext traffic
  - Configure trust anchors
  - Implement Pinning without any code

```xml
<?xml version="1.0" ?>
<network-security-config>
  <domain-config>
    <domain includeSubdomains="true">example.com</domain>
    <pin-set expiration="...">
      <pin digest="SHA-256">7HlpactkIAq2Y49orFOOQKuWxmmSFZbCoQYcRhJ3Y=</pin>
      <pin digest="SHA-256">fwza0LRMouZHRC8Ei+4PyuldPDcf3UKgO/04cDM1oE=</pin>
    </pin-set>
  </domain-config>
</network-security-config>
```
Network security configuration - Security Analysis

- Collected 1,335,322 Apps (between 2016/08/22 and 2020/03/18)
- Detected NSC in 99,212 Apps
- Could analyze 96,400 custom NSC files

- Adoption correlates with enforcement of Android 9 which deactivates cleartext traffic by default

### Target SDK

<table>
<thead>
<tr>
<th>Target SDK</th>
<th>Total Apps</th>
<th>Apps w. NSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; Android 7*</td>
<td>236,843</td>
<td>68</td>
</tr>
<tr>
<td>&gt;= Android 7</td>
<td>1,098,479</td>
<td>96,332</td>
</tr>
<tr>
<td>&gt;= Android 8</td>
<td>963,750</td>
<td>95,826</td>
</tr>
<tr>
<td>&gt;= Android 9</td>
<td>565,910</td>
<td>88,854</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>1,335,322</td>
<td>96,400</td>
</tr>
</tbody>
</table>

* Though NSC is only supported for Android 7 and higher, apps with lower target SDKs can use backport-libraries (e.g., TrustKit [25]) to implement NSC.
Cleartext traffic configuration
- Allowing cleartext traffic again is most prominent use case of NSC (>98%)
- Undermine safe defaults (introduced in Android 9)
- In-depth analysis of affected domains
  - Cleartext traffic allowed for 24,653 domains
  - 8,935 of these domains support HTTPS → cleartext traffic not necessary
- We find especially third party libraries to instruct developers to allow cleartext traffic
  - Local caching & Ad-Libraries
Network security configuration - Security Analysis

- Custom Trust Anchors configuration
  - 8,606 Apps re-enable trust for user CAs (8.67%)
  - Undermine safe defaults (introduced with Android 7)

- Gathered data on distribution of Trust Anchor configurations
- In-depth analysis of Custom Trusted CAs
  - 759 (partially) only trust their own set of CAs (30 globally, 744 for domains)
  - 836 apps added supplementary certificates (784 globally, 58 for domains)
- Find Mis-Use of debug configuration
  - At least for 41 cases, we find “Proxy” Certificates registered outside <debug-overrides>
Network security configuration - Security Analysis

- Pinning configuration
  - Only 663 Apps (0.67%) make use pinning
  - Limited use of NSC for security enhancements

  - Conducted an in-depth Pin Analysis for domains in pinning configurations
    - Fetched certificates for domains to match against pins
    - Analysis of backup pins
    - Analysis of pins’ expiration

```
<domain-config>
  <domain includeSubdomains="true">example.com</domain>
  <pin-set expiration="...">
    <pin digest="SHA-256">7HIpactIAq2Y49orFOOQKurWxmmSFZhBCoQYcRhJ3Y=</pin>
    <pin digest="SHA-256">fwza0LRMXouZHRC8Ei+4PyuldPDCF3UKgO/04cDM1oE=</pin>
  </pin-set>
</domain-config>
```
Network security configuration - Security Analysis

- Malformed NSC configurations
  - Flawed parameter values for `<domain>` → `<domain-config>` has no effect
    - e.g. using URLs (e.g. https://...) instead of domains
  - Ambiguous Pinning configurations → Pinning has no effect
    - 6 Apps with trust anchors with `overridePins="true"`
    - Apps that allow cleartext traffic for pinned domains
    - Obviously non-functional pins

- Root causes: C/P, limited documentation and tool support

```xml
<domain-config>
  <domain cleartextTrafficPermitted="true">https://example.com</domain>
  <pin-set expiration="...">
    <pin digest="SHA-256">BBBBBBBBBBBBBBBBBBBBBBBBBBBBBB...BBBBBBBBBBBBBBBBBBBBBBB</pin>
  </pin-set>
  <trust-anchors>
    <certificates src="system" overridePins="true"/>
  </trust-anchors>
</domain-config>
```
Efficacy of Google Play Safeguards

- Evaluating the efficacy of Google Play Safeguards:
  - Built various insecure custom TLS samples
  - Published on Play

```java
@Override
public boolean verify(String host, SSLSession session) {
    return true;
}

@Override
public void checkServerTrusted(X509Certificate[] chain, String authType) throws CertificateException {
}

@Override
public void onReceivedSslError(WebView view, SslErrorHandler handler, SslError error) {
    handler.proceed();
}
```
Efficacy of Google Play Safeguards

- Built various insecure custom TLS samples
- Published on Play
- Violating policies

```java
@override
public boolean verify(String host,
                      SSLSession session) {
    return true;
}

@override
public void checkServerTrusted(  
        X509Certificate[] chain, String authType)
      throws CertificateException {
}

@Override
public void onReceivedSslError(  
        WebView view,
        SslErrorHandler handler,
        SslError error) {
    handler.proceed();
}
```
Efficacy of Google Play Safeguards

- Evaluating the efficacy of Google Play Safeguards:
  - Built various insecure custom TLS samples
  - Published on Play
  - Violating policies

- Result
  - Only in two cases app was rejected
    - WebViewClient only calling proceed() on TLS error
    - Insecure version of Acra library → likely fingerprinted as parts are not blocked
  - Limited capabilities for detecting dangerous implementations
  - Only trivial cases caught
Security impact of our findings

- Assessing impact of NSC settings
  - NSC impacts security on all levels of popularity
  - Manual analysis of apps allowing HTTP
    - threats for eavesdropping
    - sensitive data transmitted via HTTP
      - including Login Data

- Assessing prevalence of insecure TLS code in Apps:
  - Replicated Fahl et al. [54]
  - Used CryptoGuard for 15,000 recent Apps
  - Checked for insecure HostnameVerifier and TrustManager implementations
  - Result
    - Still find insecure TLS code in 5,511 Apps
    - Results in line with prior work
Discussion and Takeaways

Customization is still Harmful:
- New safe defaults enhanced security
  - HTTPS and no trust for user CAs by default
  - Reduce attack surface for MITMAs
- NSC mostly used to undermine safe defaults (>88%)
- Sparse use to actually enhancing security as only 663 use pinning

Safeguards are insufficient:
- New policies to avoid common pitfalls of insecure certificate validation implementations
- Enforcement still insufficient
- Insecure implementation still present in the wild

Customization is Error-Prone:
- New forms of misconfiguration & misconception
- Even if intended to enhance security
  - Can cause the opposite
  - If tools are not used correctly
- Undiscerning use of C/P code samples
- Limited (tool) support (e.g. LINT)

Recommendation:
- Need for better (tool) support and documentation
- Enhanced capabilities of Safeguards
Questions?