



# CISPA

HELMHOLTZ CENTER FOR  
INFORMATION SECURITY

# Why Eve and Mallory Still Love Android: Revisiting TLS (In)Security in Android Applications

Marten Oltrogge, Nicolas Huaman, Sabrina Klivan  
Yasemin Acar, Michael Backes, Sascha Fahl

## 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 an 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]

SOUNTHIRARAJ 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

```
@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();
}
```

# Android and TLS - New Countermeasures

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

	Date	Android Version	Description
	1 2015-10-05	Android 6 (API 23)	Android introduces the "android:usesCleartextTraffic" flag for Manifest files, and removes the Apache HTTP Client library [33,59,64].
	2 2016-05-17		Google Play blocks apps containing unsafe implementations of the X509TrustManager interface [67].
	3 2016-08-22	Android 7 (API 24)	Android introduces NSC, distrusts user-installed certificates, and ignores the "android:usesCleartextTraffic" flag in case a NSC file is available [44,60].
	4 2016-11-25		Google Play blocks apps containing unsafe implementations of the onReceivedSslError method in WebViews [66].
	5 2017-03-01		Google Play blocks apps containing unsafe implementations of the HostnameVerifier interface [68].
	6 2017-08-21	Android 8 (API 26)	Android adds support for the "cleartextTrafficPermitted" flag for the WebView class [61].
	7 2018-08-01		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) [49,65] available to those apps.
	8 2018-08-08	Android 9 (API 28)	Sets "cleartextTrafficPermitted" to false; enforces HTTPS connections by default. Developers can revert this for specific domains or globally in NSC settings [63].
	9 2018-11-01		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) [49,65] available to existing apps.
	10 2019-08-01		New apps need to target at least Android 9; makes new safe defaults introduced with Android 9 (2018-08-08) [49,65] available to those apps.
	11 2019-11-01		Updates need to target at least Android 9; existing apps benefit from new safe defaults introduced with Android 9 (2018-08-08) [49,65].

 Affects Android OS & NSC  Affects Google Play security policy & safeguards.

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

```
@Override
public void onReceivedSslError(
    X509Certificate[] chain, String authType)
throws IOException {
    // ...
}
```

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

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

```
<?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>
```

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

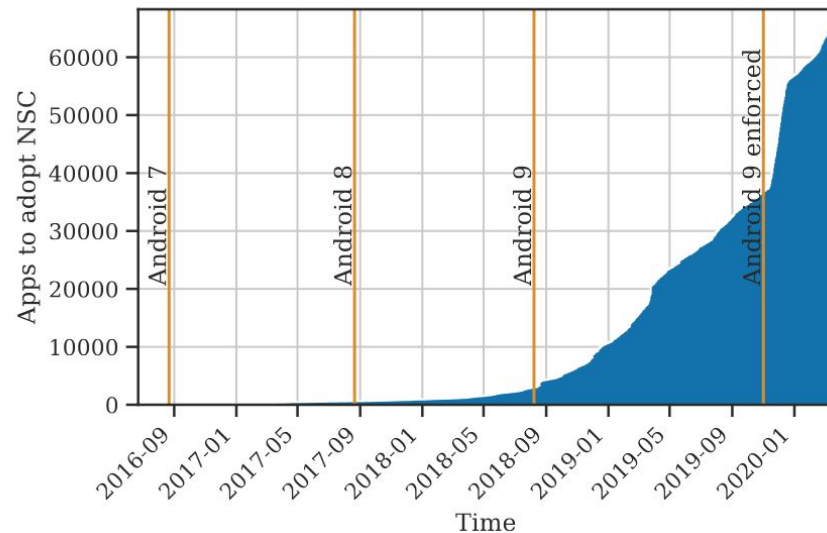
```
<?xml version="1.0" ?>
<network-security-config>
  <domain-config>
    <domain includeSubdomains="true">example.com</domain>
    <trust-anchors>
      <certificates src="system"/>
      <certificates src="user"/>
      <certificates src="@raw/ca"/>
    </trust-anchors>
  </domain-config>
</network-security-config>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

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

```
<?xml version="1.0" ?>
<network-security-config>
  <domain-config>
    <domain includeSubdomains="true">example.com</domain>
    <pin-set expiration="...">
      <pin digest="SHA-256">7HlpactkIAq2Y49orFOOQKurWxmmSFZhBCoQYcRhJ3Y=</pin>
      <pin digest="SHA-256">fwza0LRMXouZHRC8Ei+4PyuldPDcf3UKgO/04cDM1oE=</pin>
    </pin-set>
  </domain-config>
</network-security-config>
```

# Network security configuration - Security Analysis

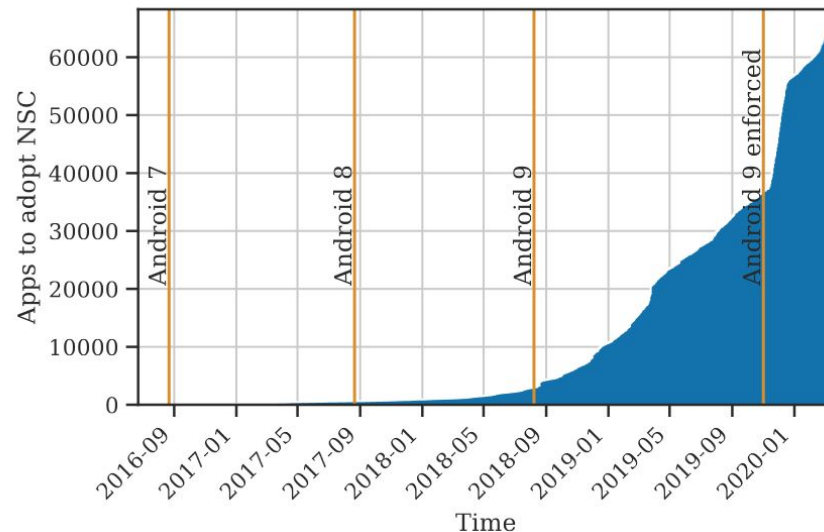
- Collected 1,335,322 Apps (between 2016/08/22 and 2020/03/18)
  - Detected NSC in 99,212 Apps
  - Could analyze 96,400 custom NSC files
- 
- Adoption correlates with enforcement of Android 9 which deactivates cleartext traffic by default



	Total Apps	Apps w. NSC
<b>Target SDK</b>		
< Android 7*	236,843	68
>= Android 7	1,098,479	96,332
>= Android 8	963,750	95,826
>= Android 9	565,910	88,854
<b>Total</b>	<b>1,335,322</b>	<b>96,400</b>

\* 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



Target	true	false
<b>&gt;= Android 9</b>	○	●
Global	57,123	1,252
Domain Specific	34,246	2,712
Total	84,060	3,908
<b>&lt; Android 9</b>	●	●
Global	4,002	36
Domain Specific	826	151
Total	4,709	185
<b>All Android Versions</b>	88,769	4,093

○ Negative impact on security; ● No impact on security; ● Positive impact on security



- 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>

```
<trust-anchors>
  <certificates src="system"/>
  <certificates src="user"/>
  <certificates src="@raw/ca"/>
</trust-anchors>
```

- 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">7HlpactkIAq2Y49orFOOQKurWxmmSFZhBCoQYcRhJ3Y=</pin>  
    <pin digest="SHA-256">fwza0LRMXouZHRC8Ei+4PyuldPDcf3UKgO/04cDM1oE=</pin>  
  </pin-set>  
</domain-config>
```

- 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

```
<domain-config>
  <domain cleartextTrafficPermitted="true">https://example.com</domain>
  <pin-set expiration="...">
    <pin digest="SHA-256">BBBBBBBBBBBBBBBBBBBBBBB...BBBBBBBBBBBBBBBBBBBBBB</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

```
@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();  
}
```

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

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

```
@Override  
public void onReceivedSslError(  
    X509CertificateChain chain, String authType)  
    throws IOException {  
    // ...  
}
```

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

# Efficacy of Google Play Safeguards

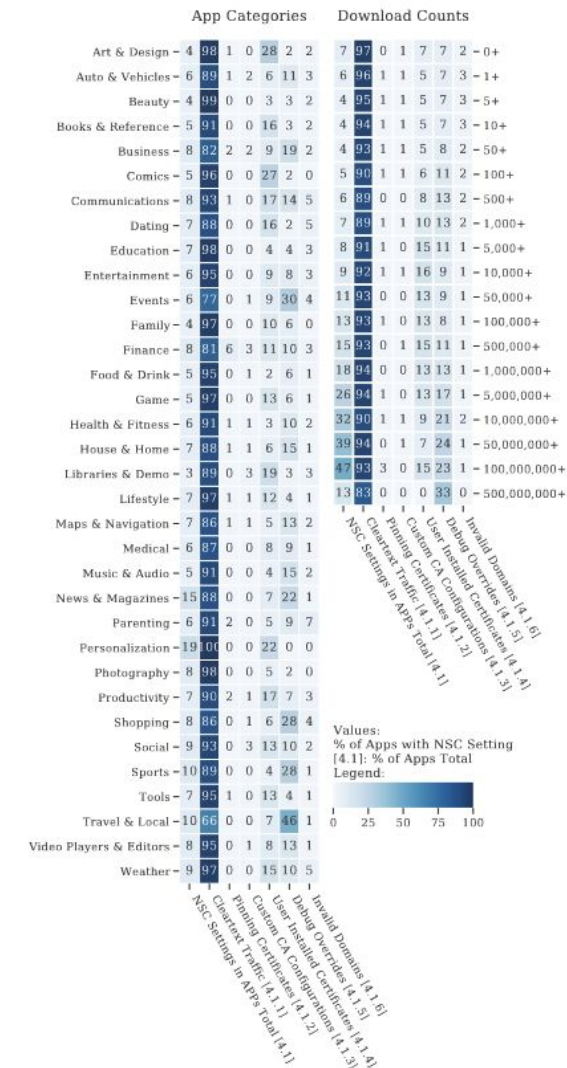
- Evaluating the efficacy of Google Play Safeguards:
  - Built various insecure custom TLS samples
  - Published on Play
  - Violating policies
  
- Result
  - Only in two cases app was rejected
    - WebViewClient only calling proceed() on TLS error
    - Insecure version of Acra library → likely fingerprinted as parts are not blocked
  - Limited capabilities for detecting dangerous implementations
  - Only trivial cases caught

Experiment	Reachability Passed	Validation Logic
<b>TrustManager</b>		
TM-U	○✓	No Validation at All
TM-R	●✓	No Validation at All
TM-D	●✓	No Validation at All
TM-R-renamed	●✓	No Validation at All, Renamed
TM-R-expired	●✓	Cert Is Not Expired
TM-R-selfsigned	●✓	Cert Is selfsigned and Not Expired
TM-R-chain	●✓	Cert Has a Chain
TM-R-chainexpired	●✓	Cert Has a Chain or Is Not Expired
<b>HostnameVerifier</b>		
HV-R	●✓	No Validation at All
HV-D	●✓	No Validation at All, Debug switch
HV-R-global	●✓	No Validation at All, Used by Default
HV-R-contains	●✓	Verify Hostname Using "string.contains"
<b>WebViewClient</b>		
WV-R	●X	always proceed
WV-D	●✓	always proceed, Debug switch
WV-wrapped	●✓	always proceed, Depend on invariant condition
<b>Library</b>		
LB-U-acra	○X	Acra with Insecure TM
LB-U-jsoup	○✓	JSoup with Insecure TM and HV
LB-U-asynhttp	○✓	async-http with insecure TM

● Always (R)eachable; ● Hidden Using a Debug Flag; ○ (U)nreachable  
 ✓ App was accepted by Google Play; X App was blocked by Google Play

# 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



## 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



Thank you

Questions?

## Why Eve and Mallory Still Love Android: Revisiting TLS (In)Security in Android Applications

Marten Oltrogge\*

*marten.oltrogge@cispa.saarland*

Nicolas Huaman<sup>†</sup>

*huaman@sec.uni-hannover.de*

Sabrina Klivan<sup>†</sup>

*amft@sec.uni-hannover.de*

Yasemin Acar<sup>†</sup>

*acar@sec.uni-hannover.de*

Michael Backes\*

*backes@cispa.saarland*

Sascha Fahl<sup>†</sup>

*fahl@sec.uni-hannover.de*

*\*CISPA Helmholtz Center for Information Security*

*<sup>†</sup>Leibniz University Hannover*

