Towards HTTPS Everywhere on Android:

We Are Not There Yet

Andrea Possemato, Yanick Fratantonio

USENIX Security ‘20

IDEMIA
augmented identity

EURECOM

NETWORK SECURITY POLICY

NETWORK SECURITY POLICY

- Declarative XML configuration file to customize the network security settings without modifying app code

NETWORK SECURITY POLICY

- Declarative XML configuration file to customize the network security settings without modifying app code

    - **Cleartext traffic opt-out**
    - **Certificate pinning**
    - **Custom trust anchors**

NETWORK SECURITY POLICY

- Declarative XML configuration file to customize the network security settings without modifying app code

    - **Cleartext traffic opt-out**
    - **Certificate pinning**
    - **Custom trust anchors**

- First version, introduced in Android 6.0 (API Level 23)
    - Only cleartext traffic opt-out

NETWORK SECURITY POLICY

- Declarative XML configuration file to customize the network security settings without modifying app code

    - **Cleartext traffic opt-out**
    - **Certificate pinning**
    - **Custom trust anchors**

- First version, introduced in Android 6.0 (API Level 23)
    - Only cleartext traffic opt-out

- Significantly extended in Android 7.0 (API Level 24)

EXAMPLES

```
<network-security-config>
  <base-config cleartextTrafficPermitted="false" />
</network-security-config>
```

EXAMPLES

```
<network-security-config>
  <base-config cleartextTrafficPermitted="true" />
  <domain-config cleartextTrafficPermitted="false">
        <domain includeSubdomains="false">android.com</domain>
        <pin-set expiration="2020-12-12">
          <pin digest="SHA-256">YZPgTZ+woNCCCIW3LH2CxQeLzB/1m42QcCTBSdgayjs=</pin>
        </pin-set>
  </domain-config>
</network-security-config>
```

EXAMPLES

```
<network-security-config>
  <base-config cleartextTrafficPermitted="true" />
  <domain-config cleartextTrafficPermitted="false">
        <domain includeSubdomains="false">android.com</domain>
        <pin-set expiration="2020-12-12">
          <pin digest="SHA-256">YZPgTZ+woNCCCIW3LH2CxQeLzB/1m42QcCTBSdgayjs=</pin>
        </pin-set>
  </domain-config>
</network-security-config>
```

EXAMPLES

```
<network-security-config>
  <base-config cleartextTrafficPermitted="true" />
  <domain-config cleartextTrafficPermitted="false">
        <domain includeSubdomains="false">android.com</domain>
        <pin-set expiration="2020-12-12">
          <pin digest="SHA-256">YZPgTZ+woNCCCIW3LH2CxQeLzB/1m42QcCTBSdgayjs=</pin>
        </pin-set>
  </domain-config>
</network-security-config>
```

EXAMPLES

```
<network-security-config>
  <base-config cleartextTrafficPermitted="true" />
  <domain-config cleartextTrafficPermitted="false">
        <domain includeSubdomains="false">android.com</domain>
        <pin-set expiration="2020-12-12">
          <pin digest="SHA-256">YZPgTZ+woNCCCIW3LH2CxQeLzB/1m42QcCTBSdgayjs=</pin>
        </pin-set>
  </domain-config>
</network-security-config>
```

EXAMPLES

```
<network-security-config>
  <debug-overrides>
    <trust-anchors>
      <certificates src="@raw/custom_cert"/>
    </trust-anchors>
  </debug-overrides>
</network-security-config>
```

DEFAULTS

- If no policy has been defined, the system applies a default one
  - Defaults change according to the API Level

DEFAULTS

- If no policy has been defined, the system applies a default one
  - Defaults change according to the API Level

- Default for API Level 28+ (Android 9)

```
<base-config cleartextTrafficPermitted="false">
    <trust -anchors>
        <certificates src="system" />
    </trust-anchors>
</base-config>
```

DEFAULTS

- If no policy has been defined, the system applies a default one
  - Defaults change according to the API Level

- Default for API Level 28+ (Android 9)

```
<base-config cleartextTrafficPermitted="false">
    <trust -anchors>
        <certificates src="system" />
    </trust-anchors>
</base-config>
```

DEFAULTS

- If no policy has been defined, the system applies a default one
  - Defaults change according to the API Level

- Default for API Level 28+ (Android 9)

```
<base-config cleartextTrafficPermitted="false">
    <trust -anchors>
        <certificates src="system" />
    </trust-anchors>
</base-config>
```

Cleartext protocols are not permitted!

DEFAULTS

- If no policy has been defined, the system applies a default one
  - Defaults change according to the API Level

- Default for API Level 28+ (Android 9)

```
<base-config cleartextTrafficPermitted="false">
    <trust -anchors>
        <certificates src="system" />
    </trust-anchors>
</base-config>
```

Cleartext protocols are not permitted!

- From Nov 2019, new Play Store policy: all apps/updates MUST target API Level 28+

POLICY WEAKNESSES

ALLOW CLEARTEXT

```
<network-security-config>
  <base-config cleartextTrafficPermitted=”true” />
</network-security-config>
```

ALLOW CLEARTEXT

```
<network-security-config>
  <base-config cleartextTrafficPermitted=”true” />
</network-security-config>
```

ALLOW CLEARTEXT

```
<network-security-config>
  <base-config cleartextTrafficPermitted=”true” />
</network-security-config>
```

An attacker on the same WiFi network of the victim can inspect and modify apps’ unencrypted connections and data <u>at will</u>

CERTIFICATE PINNING OVERRIDE

```
<network-security-config>
  <domain-config cleartextTrafficPermitted="false">
    <domain>android.com</domain>
      <pin-set expiration="2020-12-12">
        <pin digest="SHA-256">YZPgTZ+woNCCCIW3LH2CxQeLzB/1m42QcCTBSdgayjs=</pin>
      </pin-set>
  </domain-config>
  <trust-anchors>
    <certificates src="system" overridePins=”true” />
  </trust-anchors>
</network-security-config>
```

CERTIFICATE PINNING OVERRIDE

```
<network-security-config>
  <domain-config cleartextTrafficPermitted="false">
    <domain>android.com</domain>
      <pin-set expiration="2020-12-12">
        <pin digest="SHA-256">YZPgTZ+woNCCCIW3LH2CxQeLzB/1m42QcCTBSdgayjs=</pin>
      </pin-set>
  </domain-config>
   <trust-anchors>
    <certificates src="system" overridePins=”true” />
   </trust-anchors>
</network-security-config>
```

CERTIFICATE PINNING OVERRIDE

```
<network-security-config>
  <domain-config cleartextTrafficPermitted="false">
    <domain>android.com</domain>
      <pin-set expiration="2020-12-12">
        <pin digest="SHA-256">YZPgTZ+woNCCCIW3LH2CxQeLzB/1m42QcCTBSdgayjs=</pin>
      </pin-set>
  </domain-config>
  <trust-anchors>
    <certificates src="system" overridePins=”true” />
  </trust-anchors>
</network-security-config>
```

This says: implement certificate pinning, so ‘***trust connections only if they are signed by this certificate***’

CERTIFICATE PINNING OVERRIDE

```
<network-security-config>
  <domain-config cleartextTrafficPermitted="false">
    <domain>android.com</domain>
      <pin-set expiration="2020-12-12">
        <pin digest="SHA-256">YZPgTZ+woNCCCIW3LH2CxQeLzB/1m42QcCTBSdgayjs=</pin>
      </pin-set>
  </domain-config>
  <trust-anchors>
    <certificates src="system" overridePins=”true” />
  </trust-anchors>
</network-security-config>
```

This says: implement certificate pinning, so ‘***trust connections only if they are signed by this certificate***’

CERTIFICATE PINNING OVERRIDE

```
<network-security-config>
  <domain-config cleartextTrafficPermitted="false">
    <domain>android.com</domain>
      <pin-set expiration="2020-12-12">
        <pin digest="SHA-256">YZPgTZ+woNCCCIW3LH2CxQeLzB/1m42QcCTBSdgayjs=</pin>
      </pin-set>
  </domain-config>
  <trust-anchors>
    <certificates src="system" overridePins=”true” />
  </trust-anchors>
</network-security-config>
```

This says: implement certificate pinning, so ‘***trust connections only if they are signed by this certificate’***

This says: '***for standard CAs, ignore pinning specifications’***

SILENT MAN-IN-THE-MIDDLE

```
<network-security-config>
  <base-config cleartextTrafficPermitted="false" />
   <trust-anchors>
      <certificates src="system"/>
      <certificates src="user"/>
  </trust-anchors>
</network-security-config>
```

SILENT MAN-IN-THE-MIDDLE

```
<network-security-config>
  <base-config cleartextTrafficPermitted="false" />
   <trust-anchors>
      <certificates src="system"/>
      <certificates src="user"/>
   </trust-anchors>
</network-security-config>
```

SILENT MAN-IN-THE-MIDDLE

```
<network-security-config>
  <base-config cleartextTrafficPermitted="false" />
   <trust-anchors>
      <certificates src="system"/>
      <certificates src="user"/>
   </trust-anchors>
</network-security-config>
```

This says: **‘for SSL/TLS, trust any CAs in the User and System KeyStore’**

SILENT MAN-IN-THE-MIDDLE

```
<network-security-config>
  <base-config cleartextTrafficPermitted="false" />
   <trust-anchors>
      <certificates src="system"/>
      <certificates src="user"/>
   </trust-anchors>
</network-security-config>
```

This says: **'for SSL/TLS, trust any CAs in the User and System KeyStore'**

Attacker with physical access can install (silently) a custom CA in the User KeyStores

SILENT MAN-IN-THE-MIDDLE

```
<network-security-config>
  <base-config cleartextTrafficPermitted="false" />
   <trust-anchors>
      <certificates src="system"/>
      <certificates src="user"/>
   </trust-anchors>
</network-security-config>
```

This says: **'for SSL/TLS, trust any CAs in the User and System KeyStore'**

Relevant for *reverse engineering or stalkware/spouseware threat models!*

Attacker with physical access can install (silently) a custom CA in the User KeyStores

LARGE SCALE ANALYSIS

ANALYSIS

- Dataset of 125,419 apps
  - First crawl: from June-July 2019
  - Second crawl: from April-June 2020 (after Google forces all the apps to target API Level >= 28+ and changes the defaults of the policy)

ANALYSIS

- Dataset of 125,419 apps
  - First crawl: from June-July 2019
  - Second crawl: from April-June 2020 (after Google forces all the apps to target API Level >= 28+ and changes the defaults of the policy)

- Analysis of the adoption of the Network Security Policy amongst apps

ANALYSIS

- Dataset of 125,419 apps
  - First crawl: from June-July 2019
  - Second crawl: from April-June 2020 (after Google forces all the apps to target API Level >= 28+ and changes the defaults of the policy)

- Analysis of the adoption of the Network Security Policy amongst apps

- Analysis of the policy to identify potential weaknesses

ANALYSIS

- Dataset of 125,419 apps
  - First crawl: from June-July 2019
  - Second crawl: from April-June 2020 (after Google forces all the apps to target API Level >= 28+ and changes the defaults of the policy)
- Analysis of the adoption of the Network Security Policy amongst apps
- Analysis of the policy to identify potential weaknesses
- Clustering on the different policies to highlight common patterns
  - Two policies belong to the same cluster if they contain the same nodes, attributes, and values, in any order

POLICY ADOPTION

- Dataset of 125,419 apps (June-July 2019)

POLICY ADOPTION

- Dataset of 125,419 apps (June-July 2019)

- 16,332 apps (13% of the dataset) that do implement a policy
  - 7,605 of them (6% of the total) adopt the first version of the policy (Android 6.0)
  - 8,727 (6.95%) adopt the new version (Android 7.0)

POLICY ADOPTION - EVOLUTION

- Dataset of 108,542 apps (April-June 2020)
  - 16,877 apps were not available anymore on the Google Play Store

POLICY ADOPTION - EVOLUTION

- Dataset of 108,542 apps (April-June 2020)
  - 16,877 apps were not available anymore on the Google Play Store

- 36,165 apps (33.3% of the dataset) that do implement a policy
  - 23,718 of the apps adopt the first version of the policy (Android 6.0)
  - 15,492 apps adopt the new version (Android 7.0)
  - 3,045 combined both the versions

POLICY WEAKNESSES

- Cleartext:
  - 67% of the apps still configure a Network Security Policy that permits cleartext traffic

POLICY WEAKNESSES

- Cleartext:
  - 67% of the apps still configure a Network Security Policy that permits cleartext traffic

- Certificate Pinning Override:
  - Only 102 apps define a Certificate Pinning strategy through the Network Security Policy
  - 9 apps specify one or more **pin-set**, but set the **overridePins** attribute to true, making the various pin-set useless

POLICY WEAKNESSES

- Cleartext:
  - 67% of the apps still configure a Network Security Policy that permits cleartext traffic

- Certificate Pinning Override:
  - Only 102 apps define a Certificate Pinning strategy through the Network Security Policy
  - 9 apps specify one or more **pin-set**, but set the **overridePins** attribute to true, making the various pin-set useless

- Silent Man-In-The-Middle:
  - 1,159 apps specify a **trust-anchors**
  - 1,038 of them allow their SSL/TLS traffic to be potentially intercepted, since they trust the certificates in the User KeyStore

POLICY CLUSTERING

- We perform clustering to explore the dataset, and we found some 'interesting' policies

POLICY CLUSTERING

- We perform clustering to explore the dataset, and we found some 'interesting' policies

- A couple of interesting examples

POLICY CLUSTERING

- We perform clustering to explore the dataset, and we found some 'interesting' policies

- A couple of interesting examples

```
<domain>example.com</domain>
<pin-set>
        <pin digest="SHA-256">BB..(44 times)..BB</pin>
</pin-set>
```

POLICY CLUSTERING

- We perform clustering to explore the dataset, and we found some 'interesting' policies

- A couple of interesting examples

```
<domain>example.com</domain>
<pin-set>
        <pin digest="SHA-256">BB..(44 times)..BB</pin>
</pin-set>
```

41 apps share this Policy

POLICY CLUSTERING

- We perform clustering to explore the dataset, and we found some 'interesting' policies

- A couple of interesting examples

```
<domain>example.com</domain>
<pin-set>
        <pin digest="SHA-256">BB..(44 times)..BB</pin>
</pin-set>
```

41 apps share this Policy
copied/pasted from StackOverflow

POLICY CLUSTERING

- We perform clustering to explore the dataset, and we found some 'interesting' policies

- A couple of interesting examples

```
<domain>example.com</domain>
<pin-set>
        <pin digest="SHA-256">BB..(44 times)..BB</pin>
</pin-set>
```

41 apps share this Policy
copied/pasted from StackOverflow

```
<domain-config cleartextTrafficPermitted="true">
        <domain includeSubdomains="true">127.0.0.1</domain>
</domain-config>
```

POLICY CLUSTERING

- We perform clustering to explore the dataset, and we found some 'interesting' policies

- A couple of interesting examples

```
<domain>example.com</domain>
<pin-set>
        <pin digest="SHA-256">BB..(44 times)..BB</pin>
</pin-set>
```

41 apps share this Policy
copied/pasted from StackOverflow

```
<domain-config cleartextTrafficPermitted="true">
        <domain includeSubdomains="true">127.0.0.1</domain>
</domain-config>
```

1,113 apps share this Policy

POLICY CLUSTERING

- We perform clustering to explore the dataset, and we found some 'interesting' policies

- A couple of interesting examples

```
<domain>example.com</domain>
<pin-set>
        <pin digest="SHA-256">BB..(44 times)..BB</pin>
</pin-set>
```

41 apps share this Policy
copied/pasted from StackOverflow

```
<domain-config cleartextTrafficPermitted="true">
        <domain includeSubdomains="true">127.0.0.1</domain>
</domain-config>
```

1,113 apps share this Policy
from Facebook ADS SDK

ADS ECOSYSTEM

ANALYSIS - ADS ECOSYSTEM

- Dataset of 29 advertisement libraries

ANALYSIS - ADS ECOSYSTEM

- Dataset of 29 advertisement libraries

- Analysis of the developer documentation:
  - 12 libraries out of 29 asks the developer to enforce a Network Security Policy

ANALYSIS - ADS ECOSYSTEM

- Dataset of 29 advertisement libraries

- Analysis of the developer documentation:
  - 12 libraries out of 29 asks the developer to enforce a Network Security Policy

- *11 libraries ask the developer to allow cleartext on the application*

ANALYSIS - ADS ECOSYSTEM

- Impact of ad libraries on apps with cleartext configuration

ANALYSIS - ADS ECOSYSTEM

- Impact of ad libraries on apps with cleartext configuration

- Extended analysis of both the datasets:
  - Policy analysis
  - LibScout to detect third-party libraries in the apps
  - We extended LibScout to support the profile for the 11 ad libraries

ANALYSIS - ADS ECOSYSTEM

- Impact of ad libraries on apps with cleartext configuration

- Extended analysis of both the datasets:
  - Policy analysis
  - LibScout to detect third-party libraries in the apps
  - We extended LibScout to support the profile for the 11 ad libraries

- We used both detection systems provided by LibScout
  - Package name
  - Code similarity

ADS ANALYSIS - RESULTS

- 3,230 applications with Network Security Policy have an ad library
  - Disclaimer: most of the libraries matches with LibScout were with “Package Name” similarity: the number of apps with ad libraries might be bigger

ADS ANALYSIS - RESULTS

- 3,230 applications with Network Security Policy have an ad library
  - Disclaimer: most of the libraries matches with LibScout were with “Package Name” similarity: the number of apps with ad libraries might be bigger

- Of these apps, 89% allow cleartext traffic

ADS ANALYSIS - RESULTS

- 3,230 applications with Network Security Policy have an ad library
  - Disclaimer: most of the libraries matches with LibScout were with “Package Name” similarity: the number of apps with ad libraries might be bigger

- Of these apps, 89% allow cleartext traffic

- The remaining 11% do not allow cleartext connections:
  - They will not receive HTTP based advertisements (they might not work correctly)

NETWORK SECURITY POLICY LIMITATIONS

NETWORK SECURITY POLICY LIMITATIONS

- In some scenarios like ads, the developer doesn't know which domains will be contacted
  - No domain name
  - No protocol

NETWORK SECURITY POLICY LIMITATIONS

- In some scenarios like ads, the developer doesn't know which domains will be contacted
  - No domain name
  - No protocol

- A very open and potentially weak policy is the only available option
  - Forbidding HTTP by default may not be possible

NETWORK SECURITY POLICY EXTENSION

EXTENSION

- The granularity of “domain name” is not the best abstraction layer for several use cases

EXTENSION

- The granularity of “domain name” is not the best abstraction layer for several use cases

- We propose an extension to the Network Security Policy, which allows a developer to bind a specific policy to a specific package name(s)

EXTENSION

- The granularity of “domain name” is not the best abstraction layer for several use cases

- We propose an extension to the Network Security Policy, which allows a developer to bind a specific policy to a specific package name(s)

- Fully backward compatible: act as a drop-in replacement of the old version

EXTENSION

```
<network-security-config>
  <base-config cleartextTrafficPermitted="false" />
    <!-- the developer can define and use all the
          nodes of the original policy -->

  <package-config>
    <!-- introduced by our extension -->
      <package name="com.adlib.unsafe" cleartextTrafficPermitted="true" />
  </package-config>


</network-security-config>
```

EXTENSION

```
<network-security-config>
  <base-config cleartextTrafficPermitted="false" />
    <!-- the developer can define and use all the
         nodes of the original policy -->

  <package-config>
    <!-- introduced by our extension -->
      <package name="com.adlib.unsafe" cleartextTrafficPermitted="true" />
  </package-config>

</network-security-config>
```

EXTENSION

```
<network-security-config>
  <base-config cleartextTrafficPermitted="false" />
    <!-- the developer can define and use all the
         nodes of the original policy -->
  <package-config>
    <!-- introduced by our extension -->
      <package name="com.adlib.unsafe" cleartextTrafficPermitted="true" />
  </package-config>
</network-security-config>
```

Cleartext connections are allowed only if started from “com.adlib.unsafe” (Stack Introspection)

TAKEAWAYS

- We perform the first comprehensive study on the newly introduced Android Network Security Policy, identifying strengths and common pitfalls

TAKEAWAYS

- We perform the first comprehensive study on the newly introduced Android Network Security Policy, identifying strengths and common pitfalls

- We investigate the root causes leading to weak policies, and we found that third-party libraries might encourage unsafe practices

TAKEAWAYS

- We perform the first comprehensive study on the newly introduced Android Network Security Policy, identifying strengths and common pitfalls.

- We investigate the root causes leading to weak policies, and we found that third-party libraries might encourage unsafe practices.

- We designed and implemented a drop-in extension on the actual Network Security Policy, that limits the impact of third-party libraries over the security of the policy

QUESTIONS?

Twitter: @_pox_
Email: possemat@eurecom.fr
Code & Dataset: https://github.com/eurecom-s3/android_nsp