Automated analysis of Android Network Security Configurations

Anže Jenšterle, Mobile Security 2022

Goals

- Building a simple tool for extracting the NSC file from compiled APK files
- Analysing the NSC file for any misconfigurations
- Executing this tool on a corpus of applications from Google Play
- Short discussion of results

The Data Set

- Top 3000 apps from Google Play in Austria
- Some couldn’t be downloaded, so final count is 2697
- 60 GB compressed, 72 GB uncompressed, 117 GB extracted

Tools

- Use apktool to extract the APK files
  - Bash scripting helps
  - Tried Androguard, but xml resource type not fully supported yet
- Extract the xml files from the apktool dump
  - Python script that parses the manifest and gets the resource identifier
  - 1 app (Starbucks) couldn’t be fully decoded with apktool, had to manually handle the AXML, which the script now supports
- Analyze them for common errors and interesting statistics and data for security research
  - Python script with counters and parser
  - Also generates CSV file for all apps that are analyzed

CSV Contents

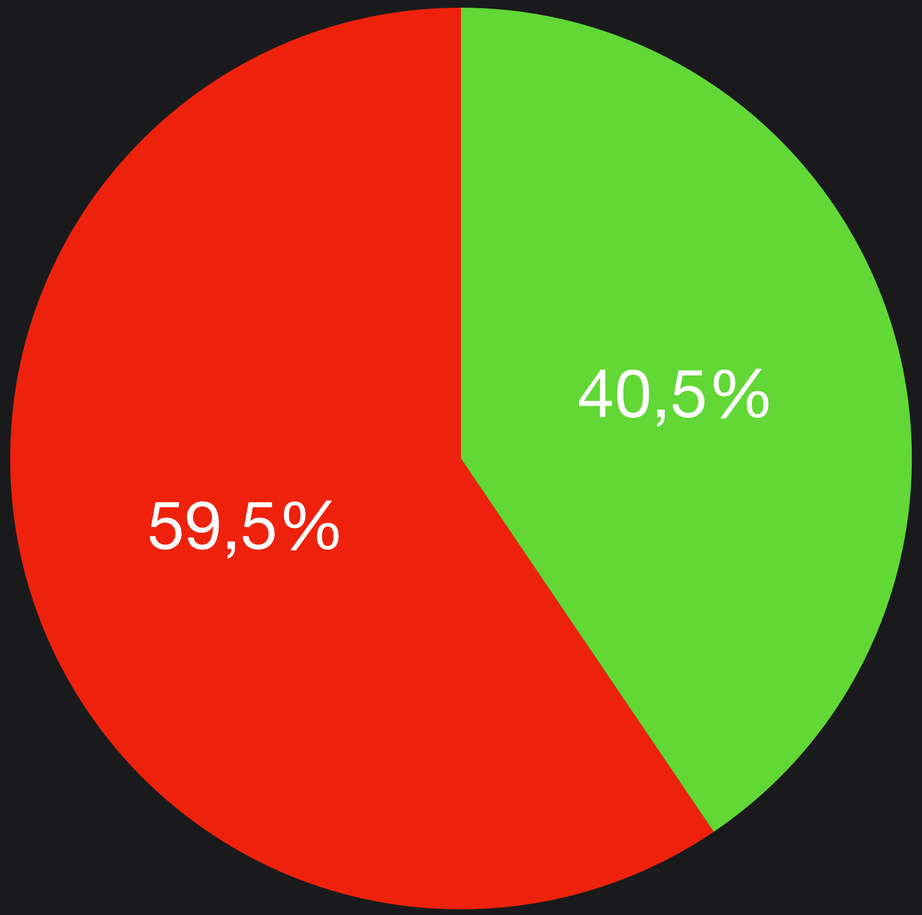
- Does app have a base config? What does it set?
- Does app allow clear text?
- Does app have debug overrides? What do they do?
- Does app use custom CAs? How many?
- Does app use certificate pinning? How much?
- Does app use special domain rules?

NSC Deployment Stats

19 apps target so low SDKs that their default NSC still allows for user certificates

91% use resource name “@xml/network_security_config”

Found some funny typos:
- @xml/netwrok_security_config
- @xml/netowrk_security
- @xml/network_sequrity_config

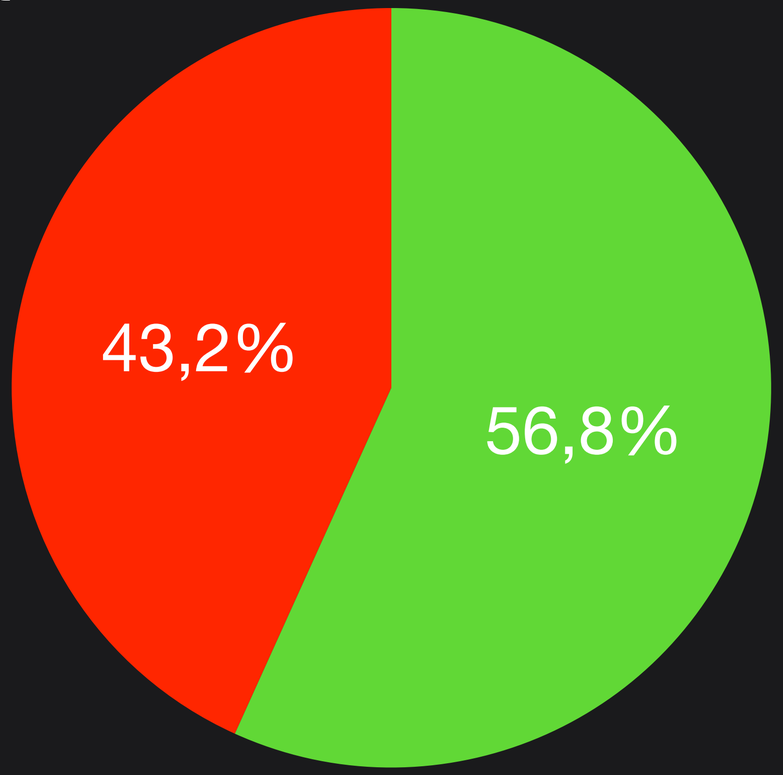


1.534 apps without NSC target high enough SDK that they use HTTPS and system store only

22 apps had NSCs that were completely empty

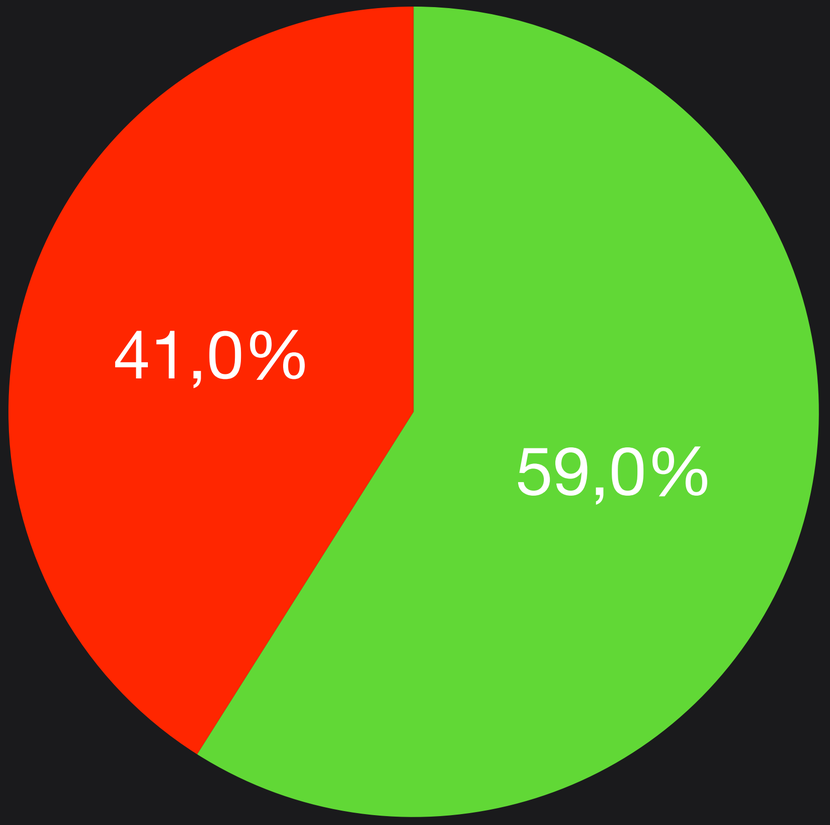
Base Config

- Applies to all undefined app traffic
- Allows users to specify trusted CAs and allow plaintext
- 15 apps had custom CAs added to the main trust
- 79% of apps with a base config allow plaintext
- 62% of apps set trust anchors



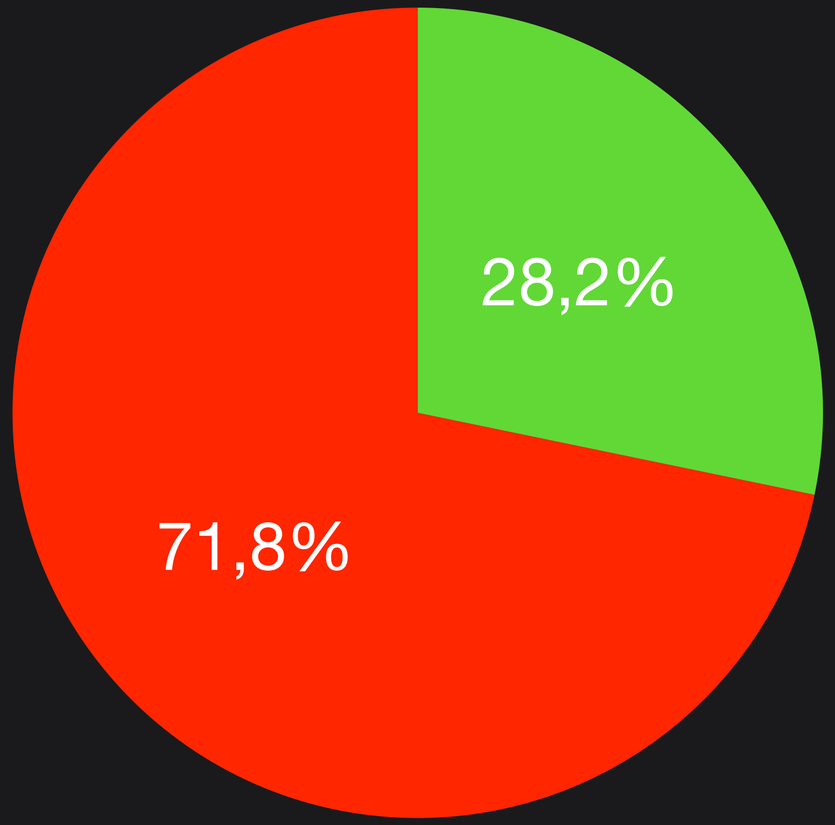
Domain Config

- Allows developers to specify domain specific rules
- 88% of domain rules observed allowed plaintext



Debug Overrides

- Allow for added trust when android:debuggable is set in manifest
- 98% add user cert store
- 24% also use system store
- 8 apps have custom CAs in debug



Pin Sets

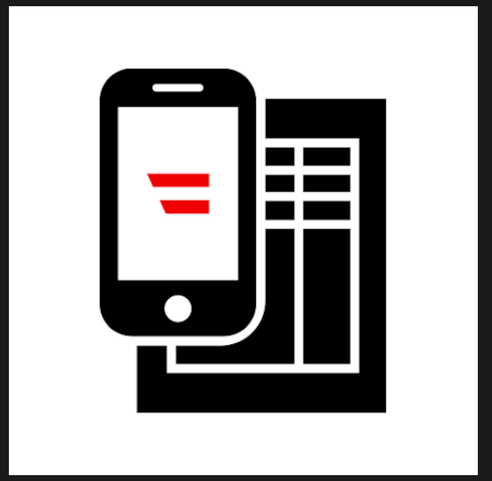
- Allow developers to pin public key hash to domain
- Used by 26 apps
- Pin sets can expire
  - After expiration, pinning no longer done
  - 4 apps had all pin sets expire at some point in the future

Interesting Finds

```
<network-security-config>
    <debug-overrides>
        <trust-anchors>
            <certificates src="user" />
        </trust-anchors>
    </debug-overrides>
    <domain-config cleartextTrafficPermitted="true">
        <domain includeSubdomains="true">192.168.8.1</domain>
        <domain includeSubdomains="true">192.168.0.1</domain>
        <domain includeSubdomains="true">192.168.1.1</domain>
        <domain includeSubdomains="true">10.0.0.138</domain>
        <domain includeSubdomains="true">fritz.box</domain>
        <domain includeSubdomains="true">net-perform.com</domain>
    </domain-config>
</network-security-config>
```

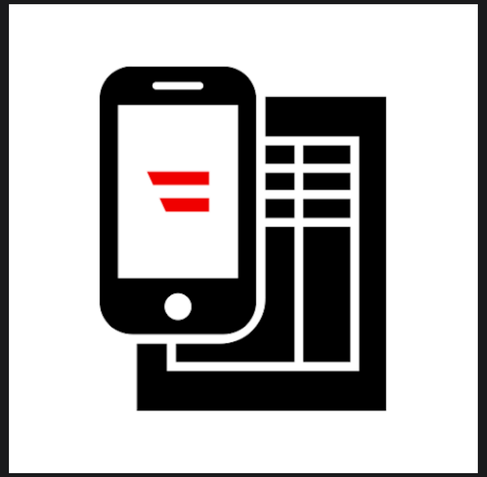
Looks like app can configure your router as well 🤔

Interesting Finds



```
<network-security-config>
    <domain-config>
        <domain includeSubdomains="true">finanzonline.bmf.gv.at</domain>
        <pin-set>
            <pin digest="SHA-256">IwaeZs0ADCb4LzAfL0plSLcdfa704rnsADUzMAyvfak=</pin>
            <pin digest="SHA-256">/HXdEOHh2yADtI1rDI+W+W49REA0BbaA08dtzYVW+Sw=</pin>
        </pin-set>
        <trustkit-config disableDefaultReportUri="true" enforcePinning="true" />
    </domain-config>
      . . .
    <base-config cleartextTrafficPermitted="true">
        <trust-anchors>
            <certificates src="system" />
            <certificates src="user" />
        </trust-anchors>
    </base-config>
</network-security-config>
```

Interesting Finds



```
<network-security-config>
    <domain-config>
        <domain includeSubdomains="true">finanzonline.bmf.gv.at</domain>
        <pin-set>
            <pin digest="SHA-256">IwaeZs0ADCb4LzAfL0pISLcdfa704rnsADUzMAyvfak=</pin>
            <pin digest="SHA-256">/HXdEOHh2yADtI1rDI+W+W49REA0BbaA08dtzYVW+Sw=</pin>
        </pin-set>
        <trustkit-config disableDefaultReportUri="true" enforcePinning="true" />
    </domain-config>
      . . .
    <base-config cleartextTrafficPermitted="true">
        <trust-anchors>
            <certificates src="system" />
            <certificates src="user" />
        </trust-anchors>
    </base-config>
</network-security-config>
```

Interesting Finds

fitbit®

```
<network-security-config>
    <domain-config cleartextTrafficPermitted="true">
        <domain includeSubdomains="true">192.168.1.0</domain>
        <domain includeSubdomains="true">192.168.0.0</domain>
        <domain includeSubdomains="true">127.0.0.1</domain>
        <domain includeSubdomains="true">10.0.0.1/</domain>
        <domain includeSubdomains="true">192.168.0.1</domain>
         . . .
        <domain includeSubdomains="true">192.168.1.255</domain>
        <trust-anchors>
            <certificates src="system" />
            <certificates src="user" />
        </trust-anchors>
    </domain-config>
</network-security-config>
```

Covers whole 192.168.0.0/23
Why do we need local net?

Interesting Finds


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

asos

Interesting Finds

```
<network-security-config>
    <base-config cleartextTrafficPermitted="true">
        <trust-anchors>
            <certificates src="system" />
        </trust-anchors>
    </base-config>
    <domain-config cleartextTrafficPermitted="false">
        <domain includeSubdomains="true">example.com</domain>
        <domain includeSubdomains="true">cdn.example2.com</domain>
    </domain-config>
    <domain-config cleartextTrafficPermitted="true">
        <domain includeSubdomains="true">127.0.0.1</domain>
    </domain-config>
    <domain-config>
        <domain includeSubdomains="true">facebook.com</domain>
        <trust-anchors>
            <certificates src="system" />
            <certificates src="user" />
        </trust-anchors>
    </domain-config>
</network-security-config>
```

Does not even support HTTPS 🤦

Interesting Finds

```
<network-security-config>
    <base-config cleartextTrafficPermitted="true">
        <trust-anchors>
            <certificates src="system" />
        </trust-anchors>
    </base-config>
    <domain-config cleartextTrafficPermitted="false">
        <domain includeSubdomains="true">example.com</domain>
        <domain includeSubdomains="true">cdn.example2.com</domain>
    </domain-config>
    <domain-config cleartextTrafficPermitted="true">
        <domain includeSubdomains="true">127.0.0.1</domain>
    </domain-config>
    <domain-config>
        <domain includeSubdomains="true">facebook.com</domain>
        <trust-anchors>
            <certificates src="system" />
            <certificates src="user" />
        </trust-anchors>
    </domain-config>
</network-security-config>
```

Why user CAs for Facebook? 🤷

Interesting Finds

Corporate login portals may use internal CAs, but they still force HTTPS

```
<network-security-config>
    <base-config cleartextTrafficPermitted="false">
        <trust-anchors>
            <certificates src="system" />
            <certificates src="user" />
        </trust-anchors>
    </base-config>
    <domain-config cleartextTrafficPermitted="false">
        <domain includeSubdomains="true">pim.microsoft.com</domain>
        <pin-set>
            <pin digest="SHA-256">grbH6X9f+atOLWXmHQTzQsx9QDo6F9ep7yaiPZUp6s0=</pin>
            <pin digest="SHA-256">UgpUVparimk8QCjtWQaUQ7EGrtrykc/L8N66EhFY3VE=</pin>
        </pin-set>
    </domain-config>
    . . .
    <domain-config>
        <domain includeSubdomains="true">phonefactor.net</domain>
        <domain includeSubdomains="true">login.live.com</domain>
        <domain includeSubdomains="true">storage.live.com</domain>
        . . .
        <domain includeSubdomains="true">sts.windows.net</domain>
        <trust-anchors>
            <certificates src="system" />
        </trust-anchors>
    </domain-config>
</network-security-config>
```

Interesting Finds

- A lot of apps allow plaintext to localhost, 192.168.0.1 and 10.0.2.2
- Targeted SDK does not correlate with adoption of NSC
- A few apps only trust Let’s Encrypt CA, but no actual pinning

Possible Improvements

- Use Androguard after XML resource support is added so we are not dependant on apktool and bash anymore
  - This will also allow adding apps with no NSC to the CSV
- Output domains per app
- Look up certificate hashes in Certificate Transparency database

Conclusions

- NCS is still not being deployed in more than 50% of top Android apps
- Developers still don’t fully understand NCS
  - Some NCS for example have a base config allowing clear-text and then extra domain configs doing the same
- Certificate pinning is basically unheard of
- A lot of certificate pinning that is done does not expire
- A lot of apps still use plaintext communications

GitHub

https://github.com/craftbyte/laqueus