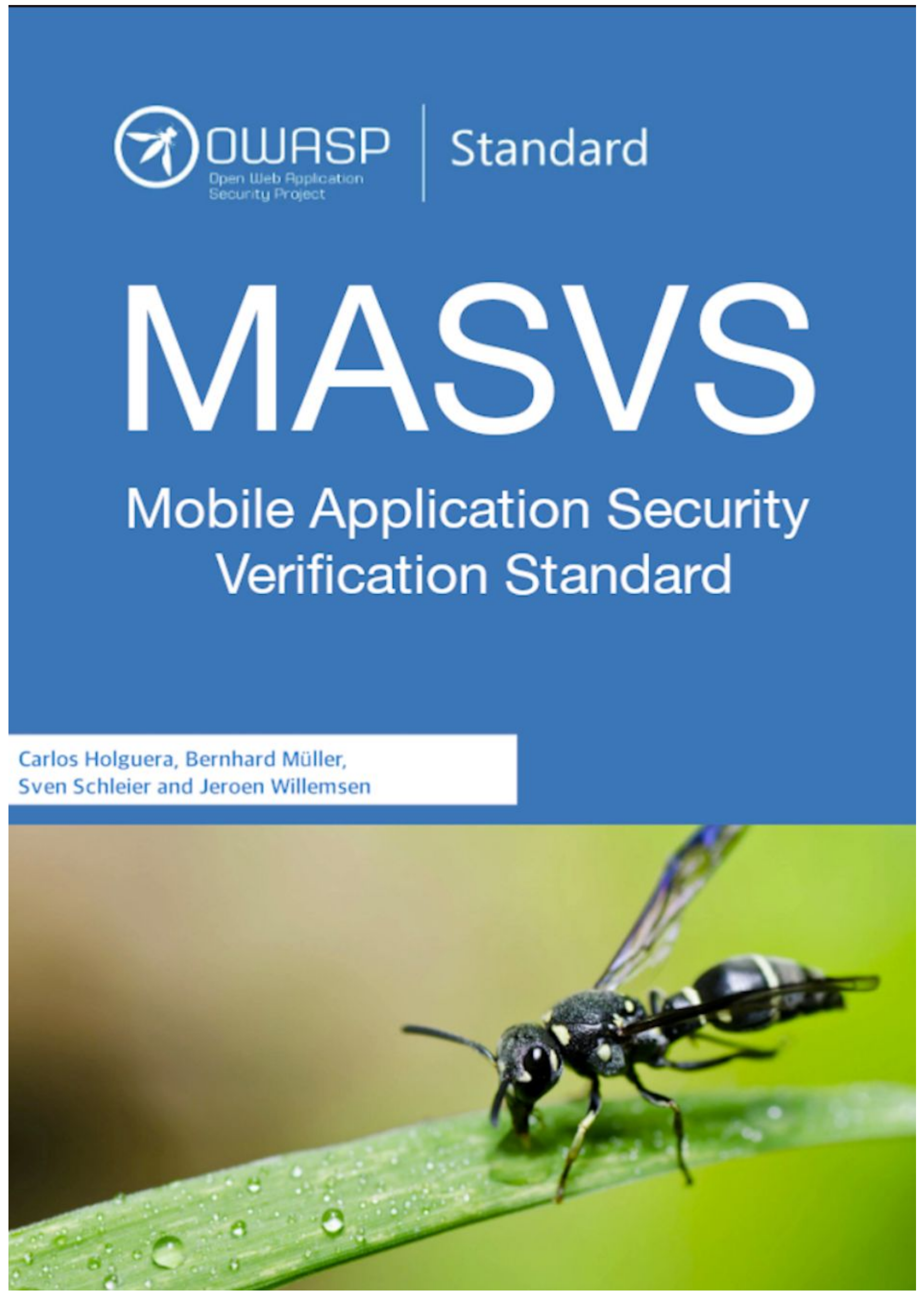
OWASP MSTG

Curious security issues and **how to protect mobile apps against them**

By Andre Carvalho, Software Engineer and Daniel Schwendner, Software Developer

GUARDSQUARE
Mobile application protection

MASVS/MSTG

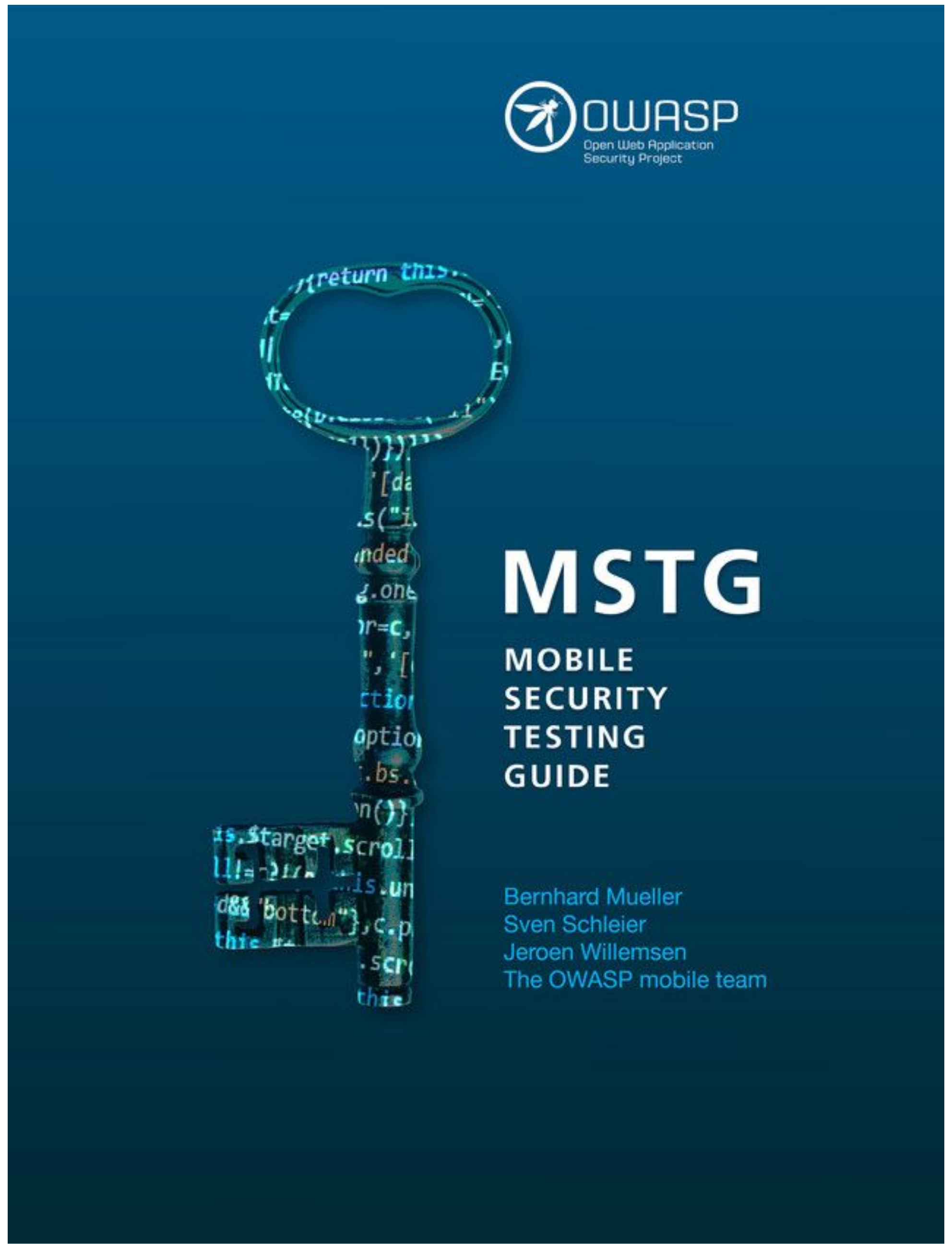


OWASP Open Web Application Security Project | Standard

MASVS

Mobile Application Security Verification Standard

Carlos Holguera, Bernhard Müller, Sven Schleier and Jeroen Willemsen



OWASP Open Web Application Security Project

MSTG

MOBILE SECURITY TESTING GUIDE

Bernhard Mueller
Sven Schleier
Jeroen Willemsen
The OWASP mobile team

MSTG Categories

MSTG-ARCH
This category lists requirements pertaining to architecture and design of the app.

MSTG-STORAGE
The protection of sensitive data, such as user credentials and private information

MSTG-CRYPTO
Assuring the cryptography used by the application follows the industry best practices

MSTG-AUTH
Requirements regarding how user accounts and sessions are to be managed

MSTG **Categories**

MSTG-NETWORK
This category ensures the confidentiality and integrity of information exchange between the mobile app and remote service endpoints

MSTG-PLATFORM
Ensuring the APP uses platform APIs and standard components in a secure manner

MSTG-CODE
Ensure that basic security coding practices are followed during development and that “free” security features offered by the compiler are activated

MSTG-RESILIENCE
Defense-in-depth measures recommended for apps that process, or give access to, sensitive data or functionality

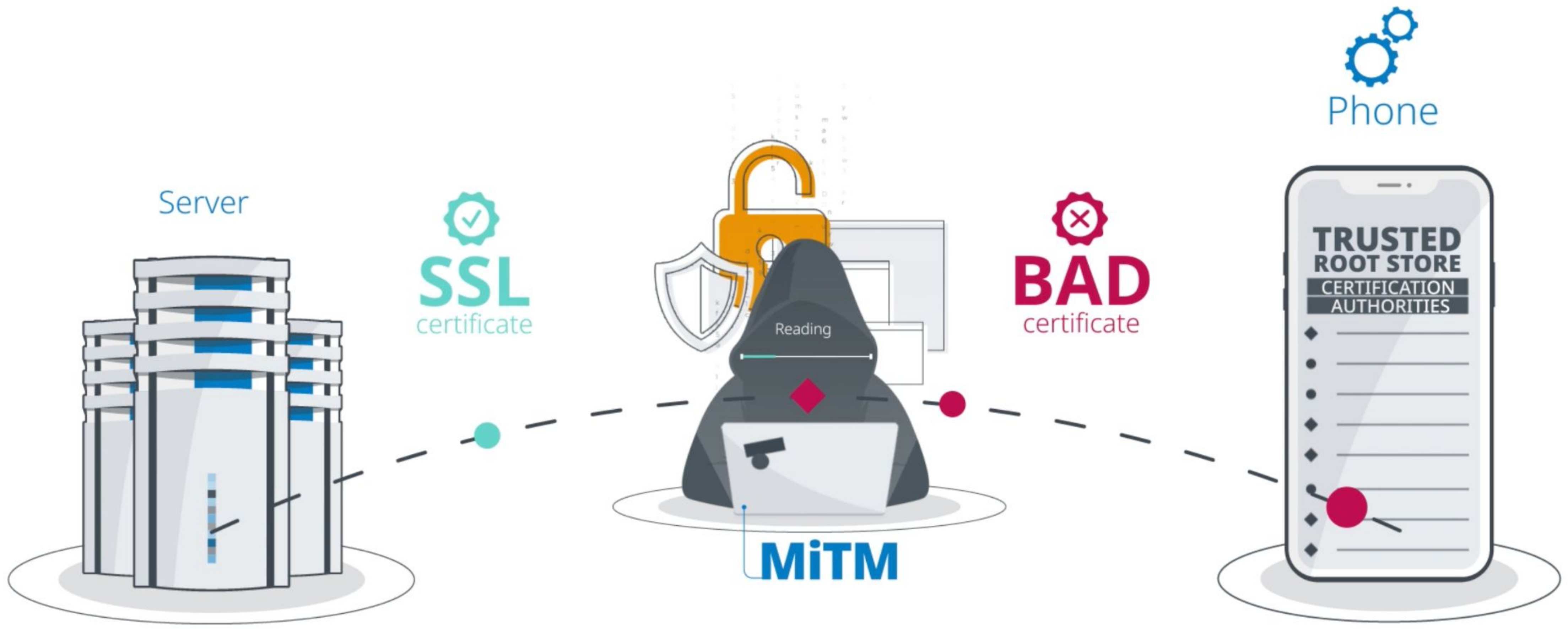
Which Security Issues **We Will Analyze**

1. **Android Networking**
   1.1. Cleartext communications
   1.2. Missing backend attestation
2. **Android Platform**
   2.1. Input interception
3. **Android Data Storage**
   3.1. Sensitive Data Disclosure
      a. Logging
      b. Send data to web
      c. Writing data to disk

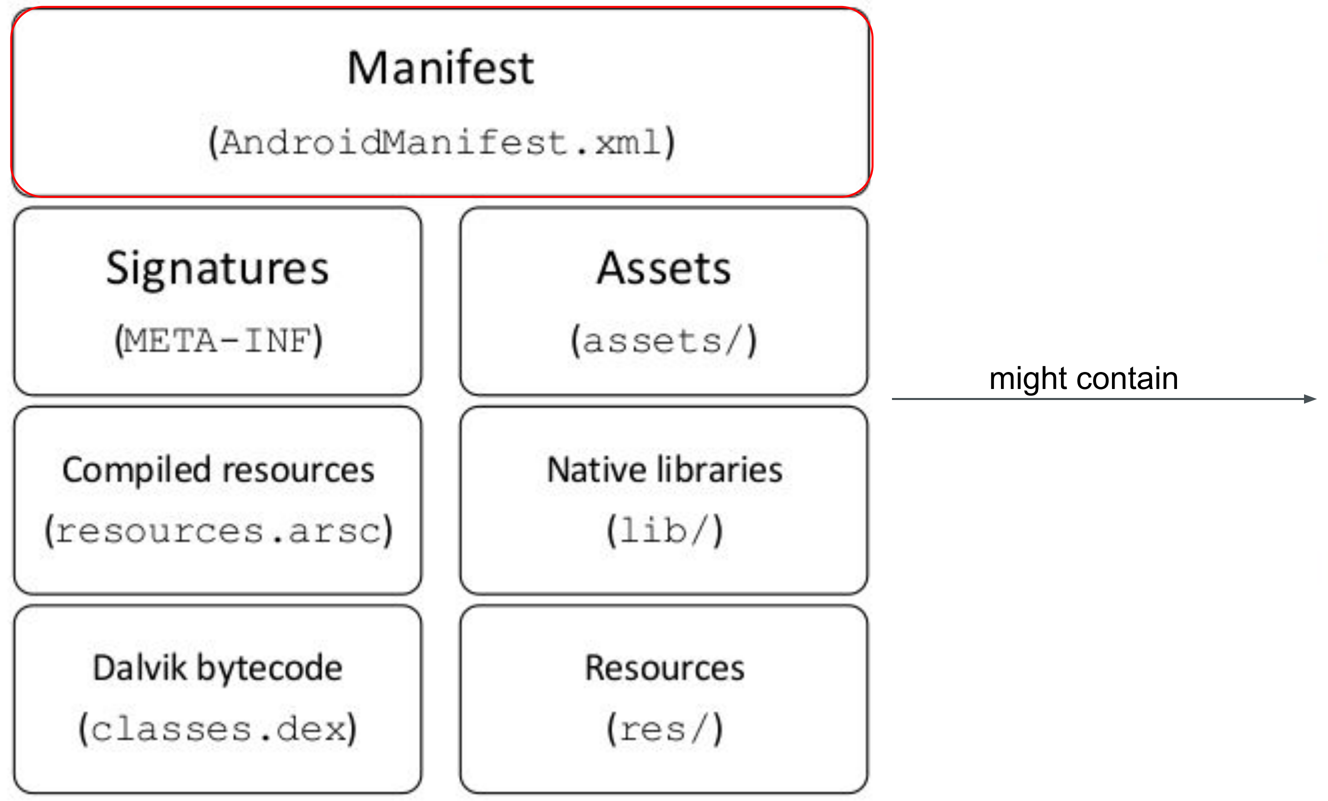
1. Android Networking

Cleartext Communications: **What is it?** 🤔

- In cryptography and computer security, a man-in-the-middle attack is a cyberattack where the attacker secretly relays and possibly alters the communications between two parties who believe that they are directly communicating with each other, as the attacker has inserted themselves between the two parties.
- Cleartext is any transmitted or stored information that is not encrypted or meant to be encrypted. When an app communicates with the server using a cleartext network traffic, such as HTTP, it could raise the risk of eavesdropping and tampering of content.

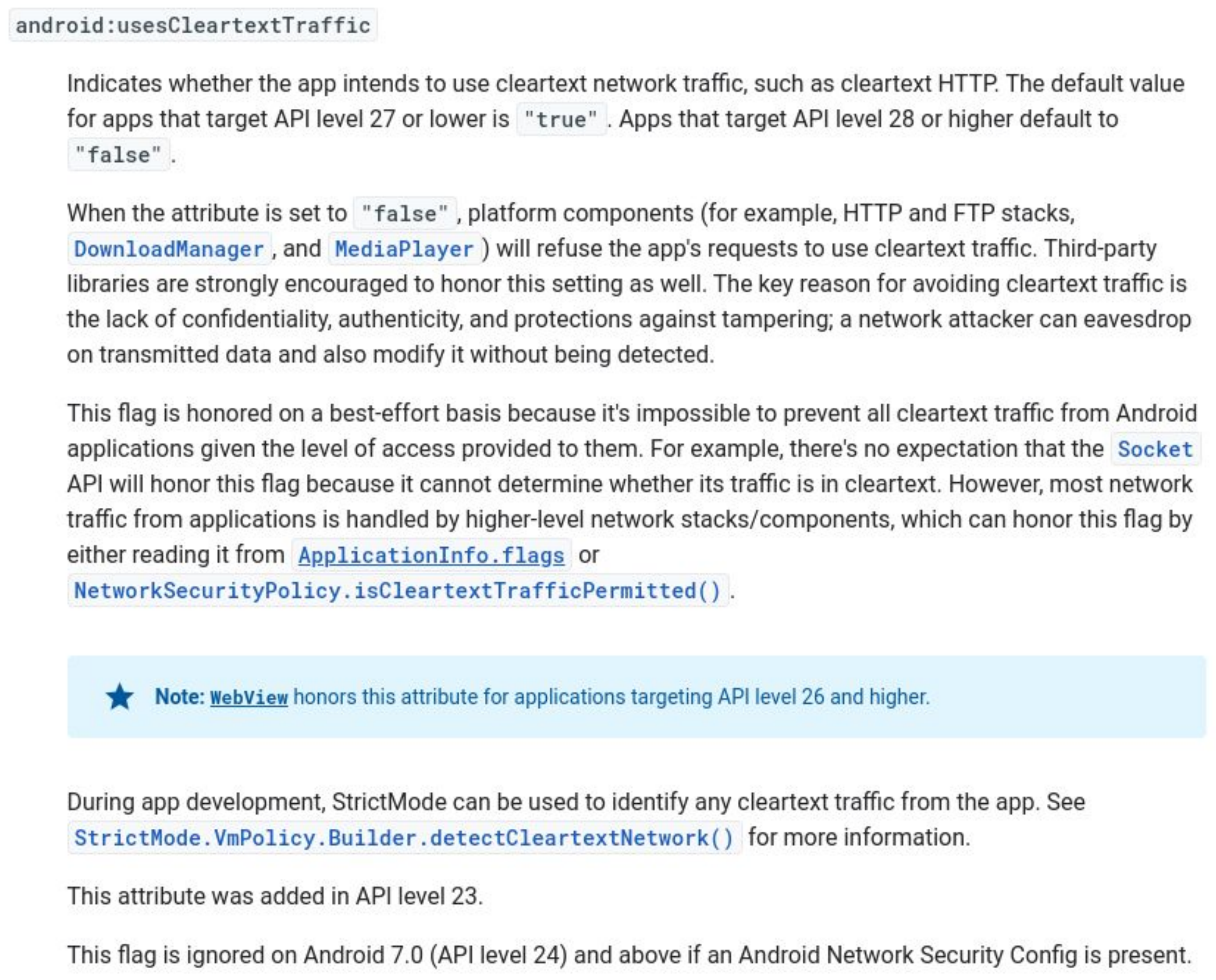


Cleartext Communications – **How to protect** 🙅‍♂️



```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
  <domain-config>
    <domain includeSubdomains="true">currency-converter.demo.approov.io</domain>
    <pin-set expiration="2021-11-23">
      <pin digest="SHA-256">Nny5/17/K1xsiGjNRppl7Px3OyXoi71RJ30kQ7ceV0I=</pin>
      <pin digest="SHA-256">1mNK3usLI0cgMXDFsRcvi5n8fXHSRTe/Xh0M1btY54g=</pin>
    </pin-set>
  </domain-config>
</network-security-config>
```

Cleartext Communications - **How to protect** 🙅‍♂️



`android:usesCleartextTraffic`

Indicates whether the app intends to use cleartext network traffic, such as cleartext HTTP. The default value for apps that target API level 27 or lower is `"true"`. Apps that target API level 28 or higher default to `"false"`.

When the attribute is set to `"false"`, platform components (for example, HTTP and FTP stacks, `DownloadManager`, and `MediaPlayer`) will refuse the app's requests to use cleartext traffic. Third-party libraries are strongly encouraged to honor this setting as well. The key reason for avoiding cleartext traffic is the lack of confidentiality, authenticity, and protections against tampering; a network attacker can eavesdrop on transmitted data and also modify it without being detected.

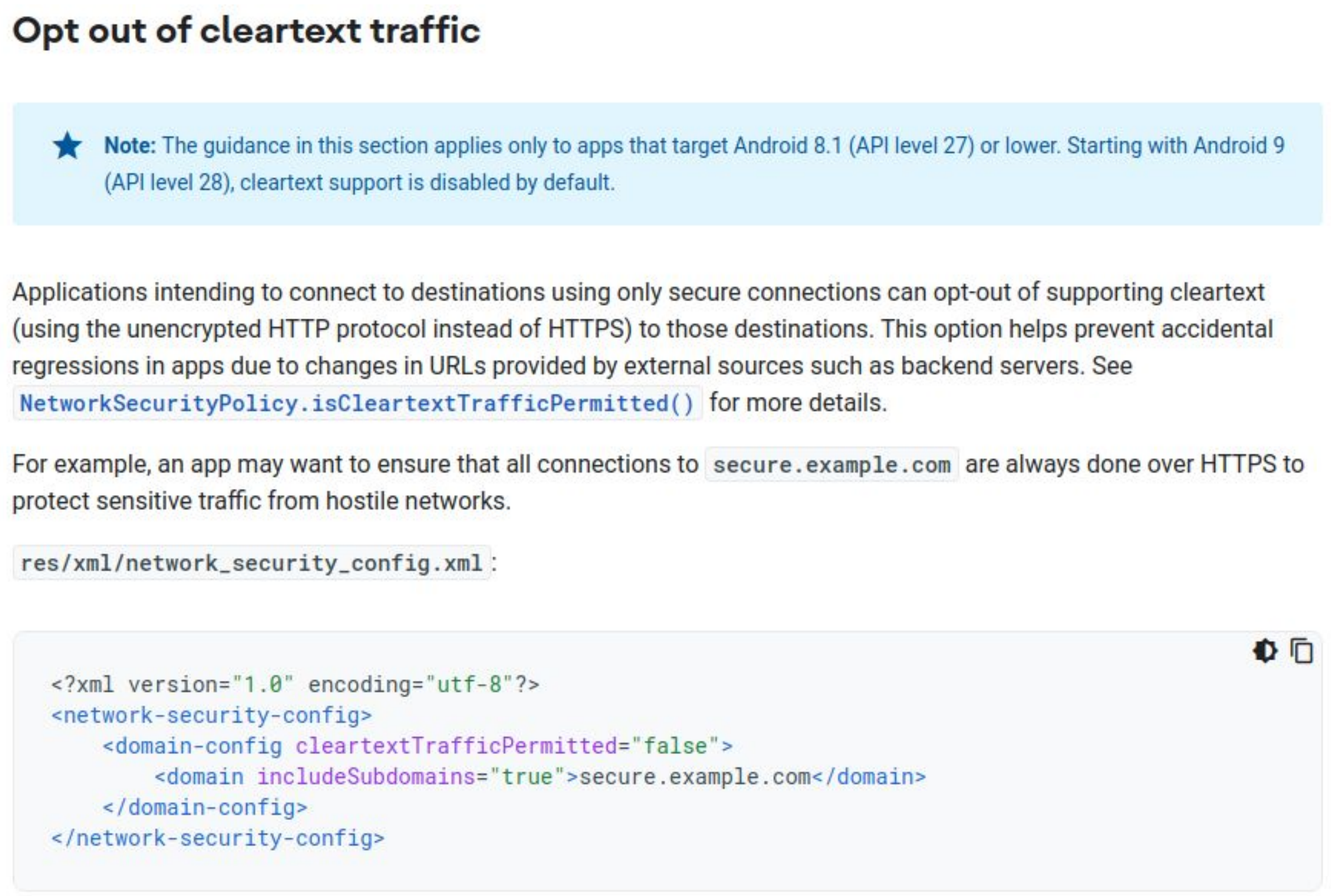
This flag is honored on a best-effort basis because it's impossible to prevent all cleartext traffic from Android applications given the level of access provided to them. For example, there's no expectation that the `Socket` API will honor this flag because it cannot determine whether its traffic is in cleartext. However, most network traffic from applications is handled by higher-level network stacks/components, which can honor this flag by either reading it from `ApplicationInfo.flags` or `NetworkSecurityPolicy.isCleartextTrafficPermitted()`.

★ **Note:** `WebView` honors this attribute for applications targeting API level 26 and higher.

During app development, StrictMode can be used to identify any cleartext traffic from the app. See `StrictMode.VmPolicy.Builder.detectCleartextNetwork()` for more information.

This attribute was added in API level 23.

This flag is ignored on Android 7.0 (API level 24) and above if an Android Network Security Config is present.



Opt out of cleartext traffic

★ **Note:** The guidance in this section applies only to apps that target Android 8.1 (API level 27) or lower. Starting with Android 9 (API level 28), cleartext support is disabled by default.

Applications intending to connect to destinations using only secure connections can opt-out of supporting cleartext (using the unencrypted HTTP protocol instead of HTTPS) to those destinations. This option helps prevent accidental regressions in apps due to changes in URLs provided by external sources such as backend servers. See `NetworkSecurityPolicy.isCleartextTrafficPermitted()` for more details.

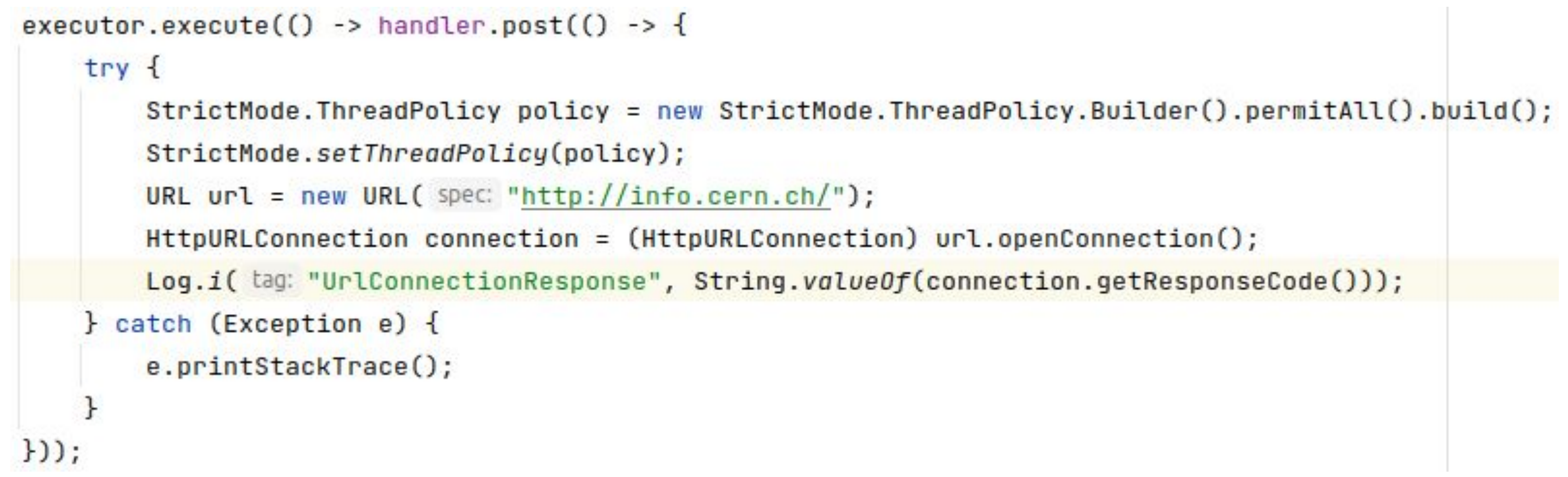
For example, an app may want to ensure that all connections to `secure.example.com` are always done over HTTPS to protect sensitive traffic from hostile networks.

`res/xml/network_security_config.xml`:

```xml
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config cleartextTrafficPermitted="false">
        <domain includeSubdomains="true">secure.example.com</domain>
    </domain-config>
</network-security-config>
```

Cleartext Communications – **How to protect** 🙅‍♂️

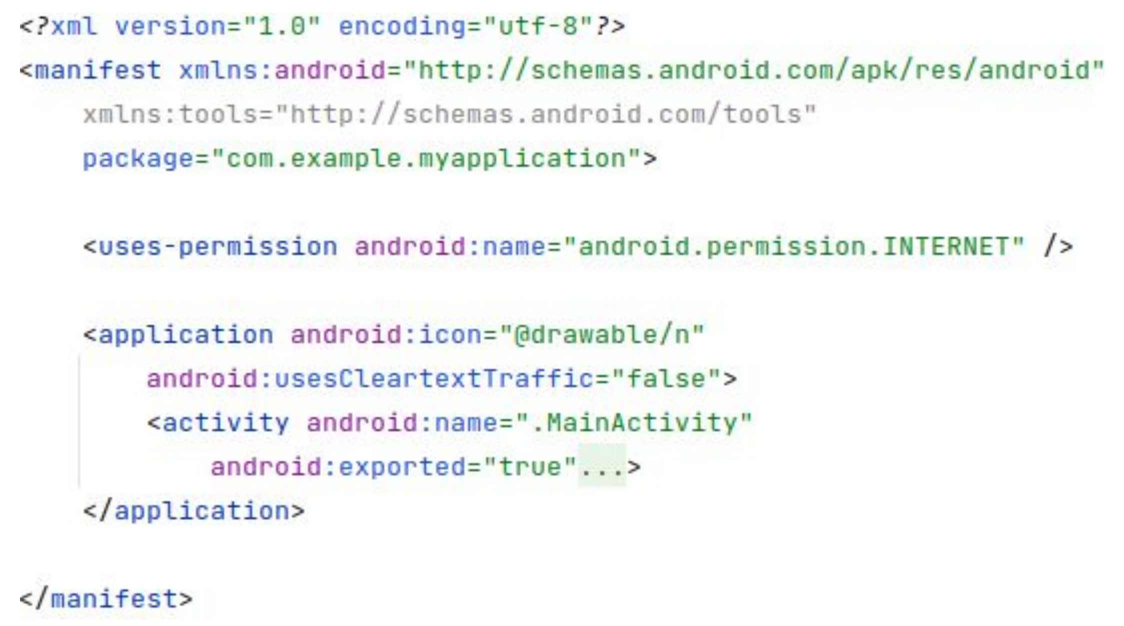
API: 23



```
executor.execute(() -> handler.post(() -> {
    try {
        StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
        StrictMode.setThreadPolicy(policy);
        URL url = new URL( spec: "http://info.cern.ch/");
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        Log.i( tag: "UrlConnectionResponse", String.valueOf(connection.getResponseCode()));
    } catch (Exception e) {
        e.printStackTrace();
    }
}));
```

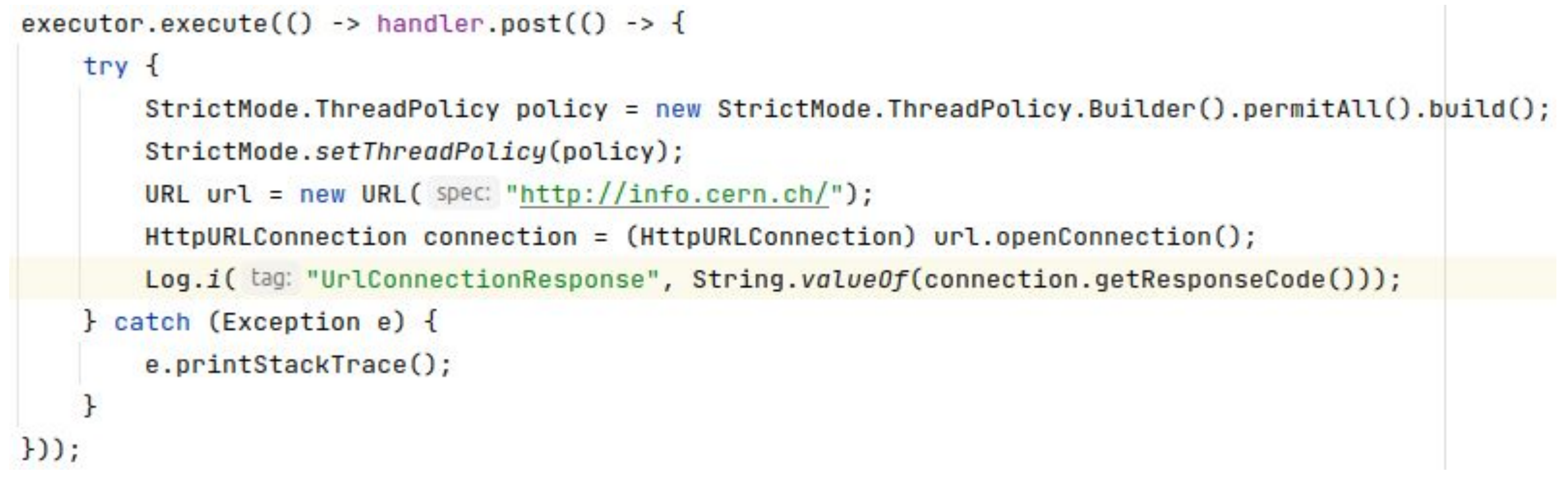
```
2022-06-03 10:31:51.038 13757-13757/? I/UrlConnectionStatus: 200
```

Cleartext Communications – **How to protect**



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.myapplication">

    <uses-permission android:name="android.permission.INTERNET" />

    <application android:icon="@drawable/n"
        android:usesCleartextTraffic="false">
        <activity android:name=".MainActivity"
            android:exported="true"...>
    </application>

</manifest>
```

API: 23

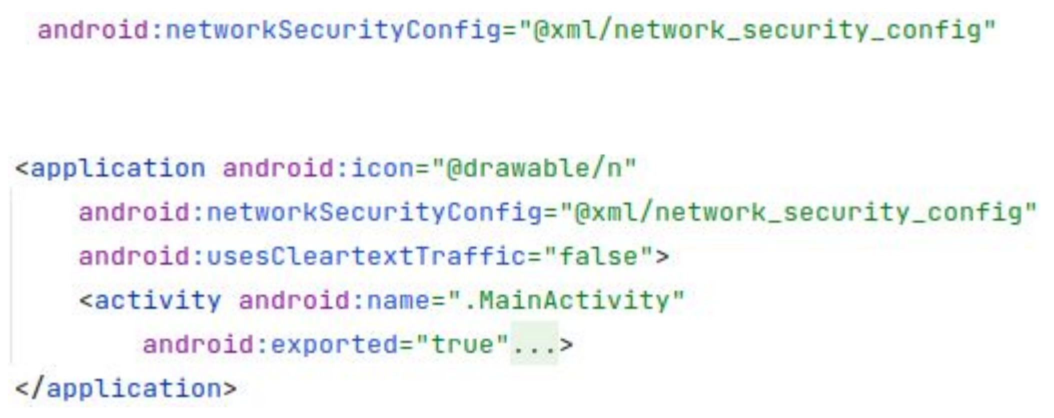


```
executor.execute(() -> handler.post(() -> {
    try {
        StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
        StrictMode.setThreadPolicy(policy);
        URL url = new URL( spec: "http://info.cern.ch/");
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        Log.i( tag: "UrlConnectionResponse", String.valueOf(connection.getResponseCode()));
    } catch (Exception e) {
        e.printStackTrace();
    }
}));
```

```
2022-06-03 10:39:14.310 14482-14482/? I/UrlConnectionResponse: FAILED
```

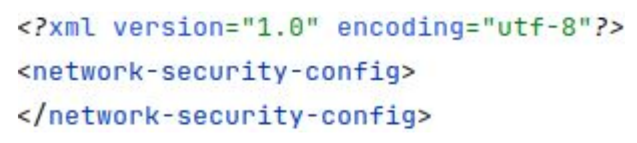
Cleartext Communications – **How to protect** 🙅‍♂️

API: 23

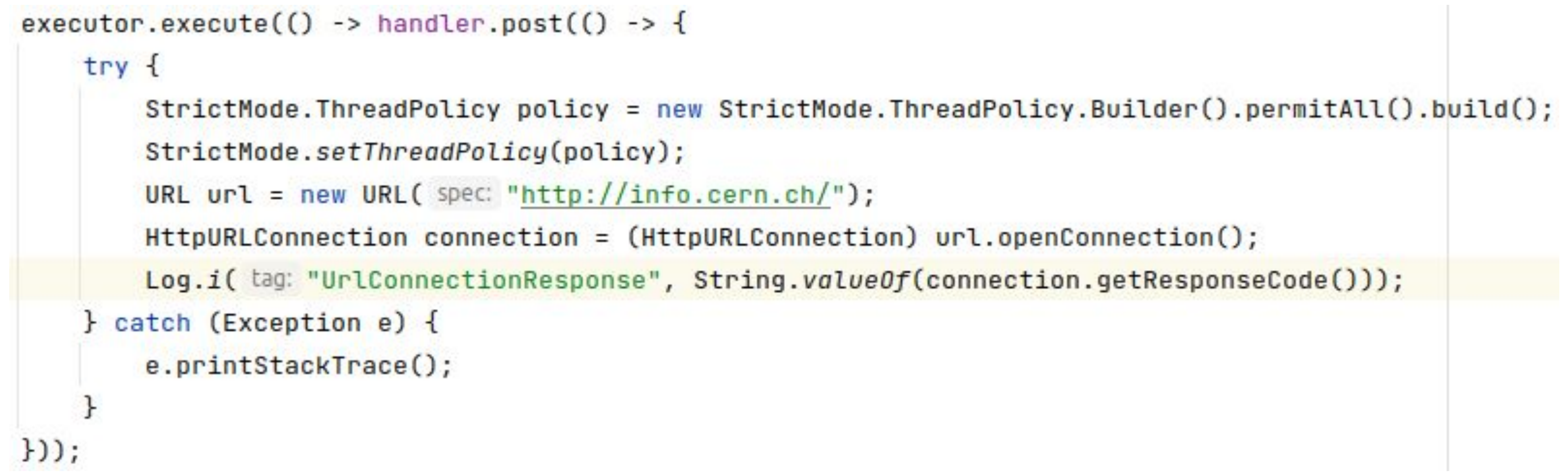


```
android:networkSecurityConfig="@xml/network_security_config"
```

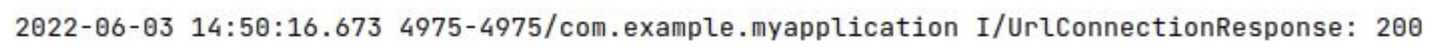
```
<application android:icon="@drawable/n"
    android:networkSecurityConfig="@xml/network_security_config"
    android:usesCleartextTraffic="false">
    <activity android:name=".MainActivity"
        android:exported="true"...>
</application>
```



```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
</network-security-config>
```



```
executor.execute(() -> handler.post(() -> {
    try {
        StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
        StrictMode.setThreadPolicy(policy);
        URL url = new URL( spec: "http://info.cern.ch/");
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        Log.i( tag: "UrlConnectionResponse", String.valueOf(connection.getResponseCode()));
    } catch (Exception e) {
        e.printStackTrace();
    }
}));
```



```
2022-06-03 14:50:16.673 4975-4975/com.example.myapplication I/UrlConnectionResponse: 200
```

Cleartext Communications - **How to protect** 🙅‍♂️

API: 23

```
android:networkSecurityConfig="@xml/network_security_config"
```

```
<application android:icon="@drawable/n"
    android:networkSecurityConfig="@xml/network_security_config"
    android:usesCleartextTraffic="true"
    tools:targetApi="n">
    <activity android:name=".MainActivity"
        android:exported="true"...>
</application>
```

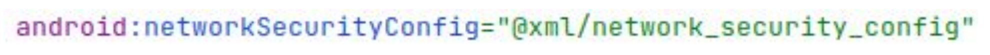
```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <base-config cleartextTrafficPermitted="false" />
</network-security-config>
```

```
executor.execute(() -> handler.post(() -> {
    try {
        StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
        StrictMode.setThreadPolicy(policy);
        URL url = new URL( spec: "http://info.cern.ch/");
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        Log.i( tag: "UrlConnectionResponse", String.valueOf(connection.getResponseCode()));
    } catch (Exception e) {
        e.printStackTrace();
    }
}));
```

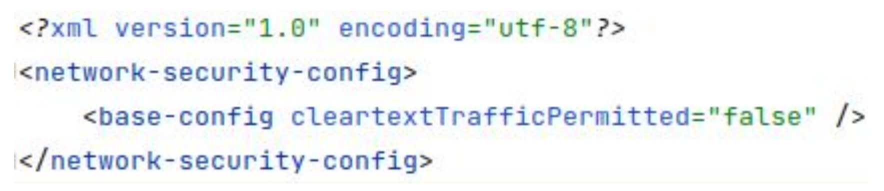
```
2022-06-03 14:24:25.317 16422-16422/? I/UrlConnectionResponse: FAILED
```

Cleartext Communications – **How to protect** 🙅‍♂️

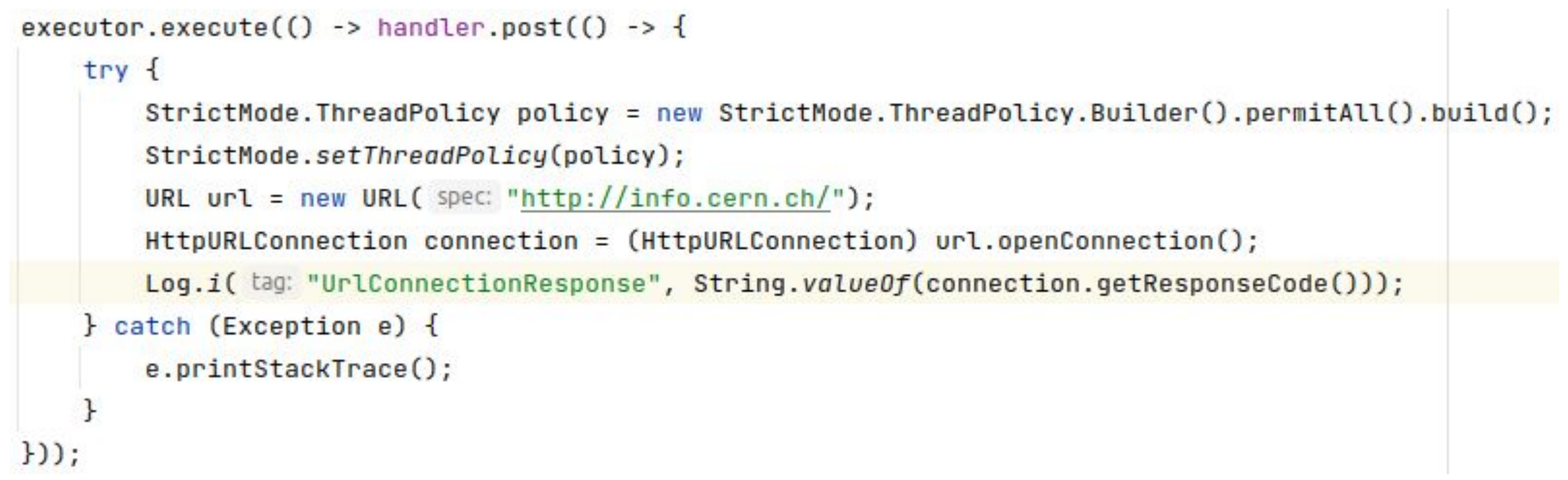
API: 23



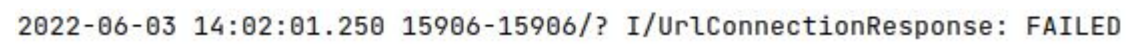
```
android:networkSecurityConfig="@xml/network_security_config"
```



```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <base-config cleartextTrafficPermitted="false" />
</network-security-config>
```

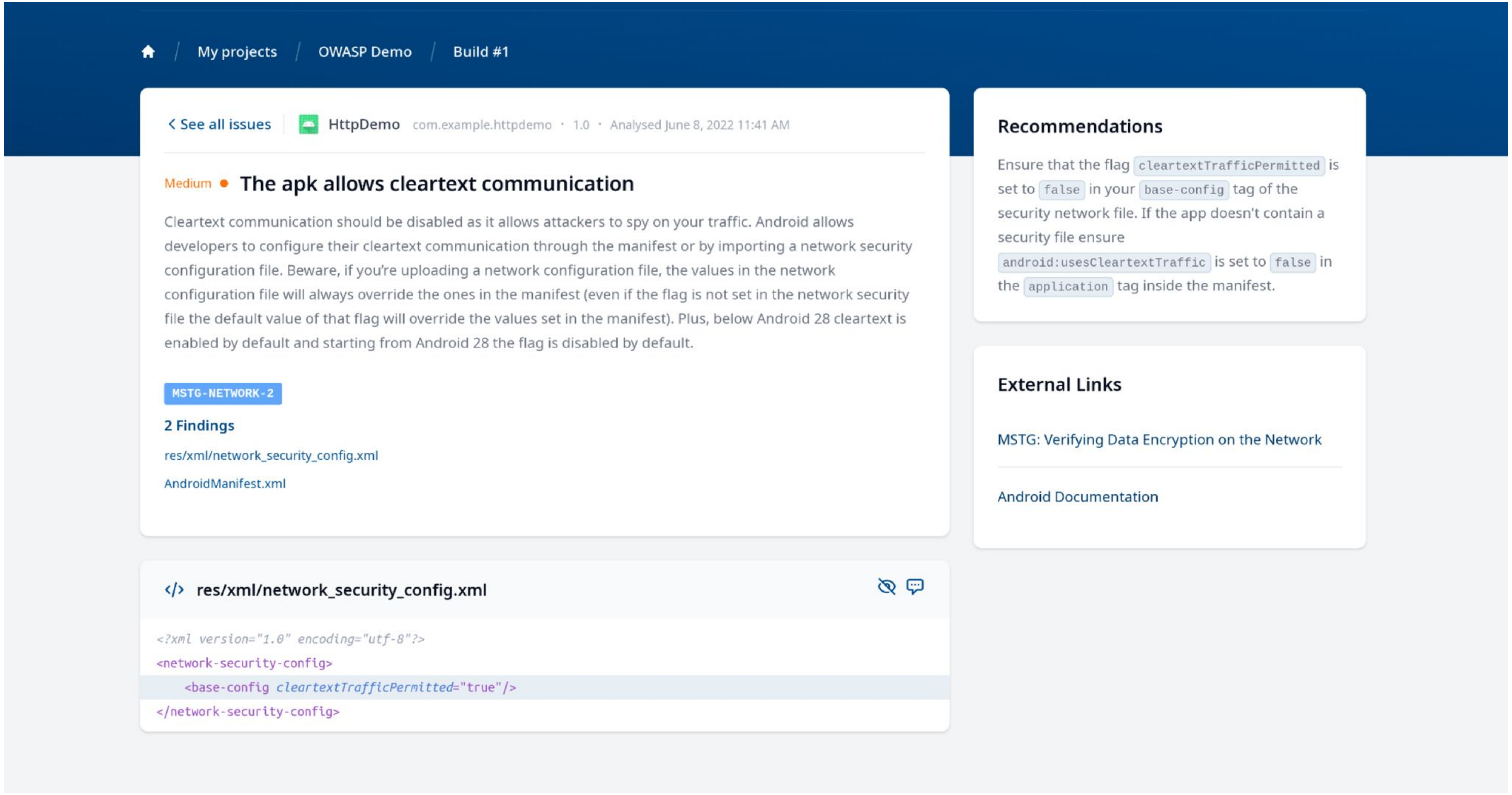


```
executor.execute(() -> handler.post(() -> {
    try {
        StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
        StrictMode.setThreadPolicy(policy);
        URL url = new URL( spec: "http://info.cern.ch/");
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        Log.i( tag: "UrlConnectionResponse", String.valueOf(connection.getResponseCode()));
    } catch (Exception e) {
        e.printStackTrace();
    }
}));
```



```
2022-06-03 14:02:01.250 15906-15906/? I/UrlConnectionResponse: FAILED
```

Cleartext Communications – **How to protect** 🙅‍♂️



< See all issues | HttpDemo com.example.httpdemo · 1.0 · Analysed June 8, 2022 11:41 AM

Medium ● **The apk allows cleartext communication**

Cleartext communication should be disabled as it allows attackers to spy on your traffic. Android allows developers to configure their cleartext communication through the manifest or by importing a network security configuration file. Beware, if you're uploading a network configuration file, the values in the network configuration file will always override the ones in the manifest (even if the flag is not set in the network security file the default value of that flag will override the values set in the manifest). Plus, below Android 28 cleartext is enabled by default and starting from Android 28 the flag is disabled by default.

MSTG-NETWORK-2

2 Findings

res/xml/network_security_config.xml

AndroidManifest.xml

res/xml/network_security_config.xml

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <base-config cleartextTrafficPermitted="true"/>
</network-security-config>
```

Recommendations

Ensure that the flag `cleartextTrafficPermitted` is set to `false` in your `base-config` tag of the security network file. If the app doesn't contain a security file ensure `android:usesCleartextTraffic` is set to `false` in the `application` tag inside the manifest.

External Links

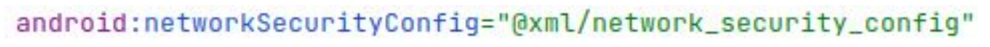
MSTG: Verifying Data Encryption on the Network

Android Documentation

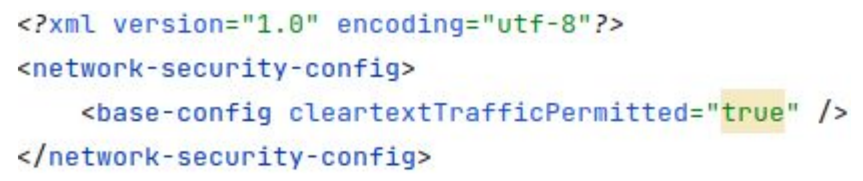
(Source: AppSweep)

Cleartext Communications – **How to protect** 🙅‍♂️

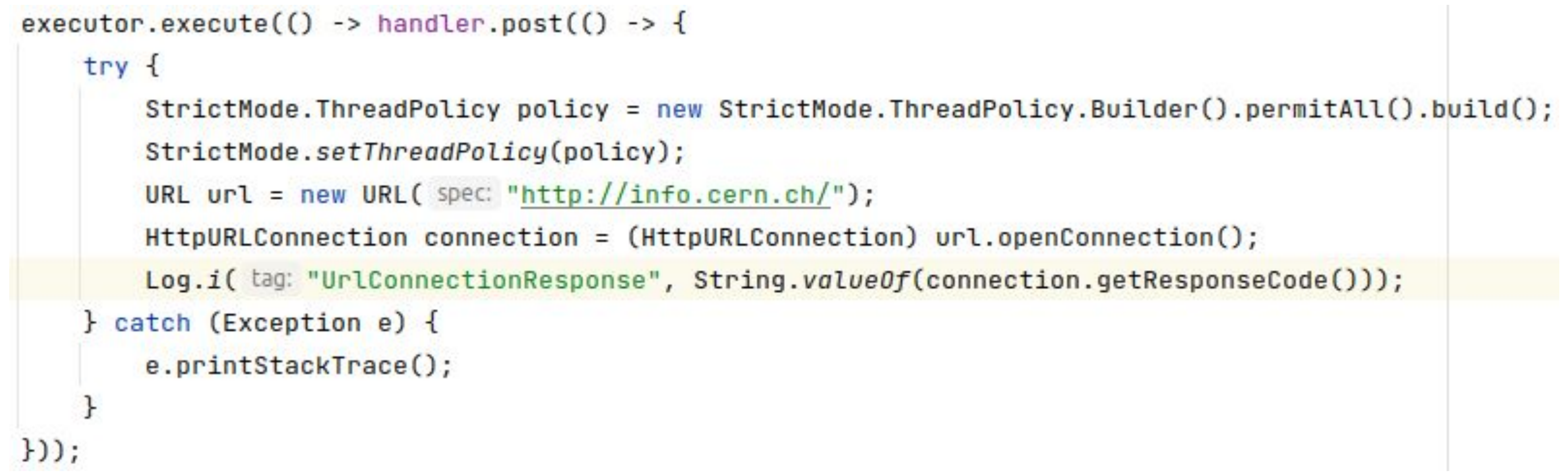
API: 28



```
android:networkSecurityConfig="@xml/network_security_config"
```



```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <base-config cleartextTrafficPermitted="true" />
</network-security-config>
```

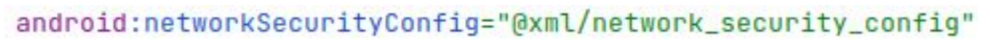


```
executor.execute(() -> handler.post(() -> {
    try {
        StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
        StrictMode.setThreadPolicy(policy);
        URL url = new URL( spec: "http://info.cern.ch/");
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        Log.i( tag: "UrlConnectionResponse", String.valueOf(connection.getResponseCode()));
    } catch (Exception e) {
        e.printStackTrace();
    }
}));
```

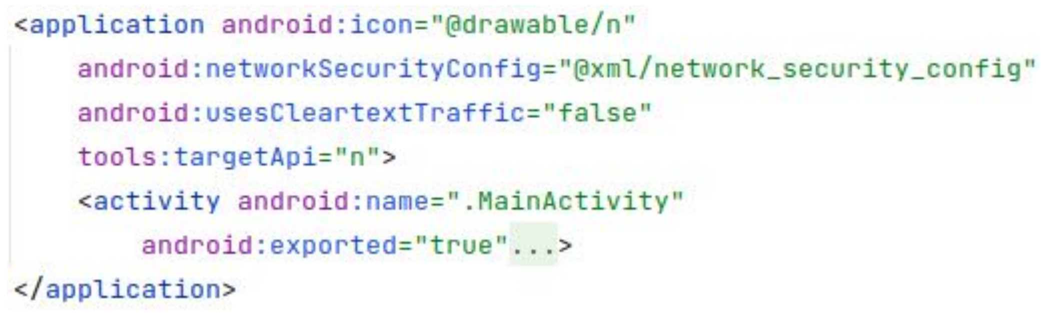
```
2022-06-03 14:14:48.041 16081-16081/? I/UrlConnectionResponse: 200
```

Cleartext Communications – **How to protect** 🙅‍♂️

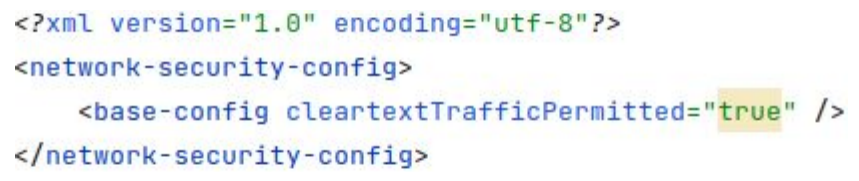
API: 28



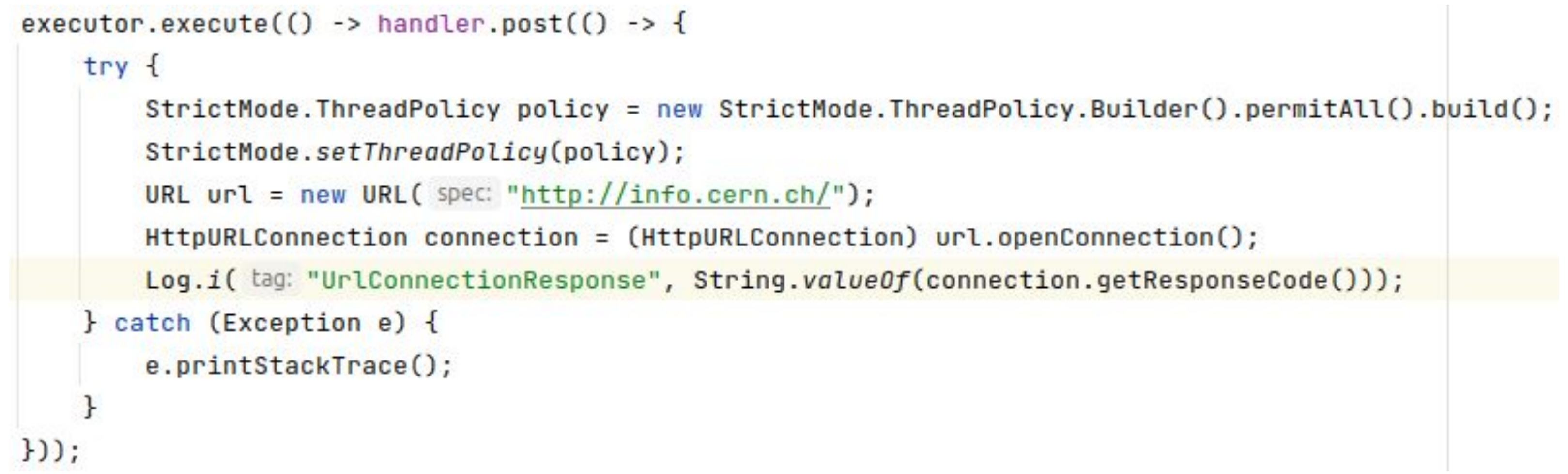
```
android:networkSecurityConfig="@xml/network_security_config"
```



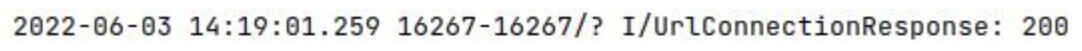
```
<application android:icon="@drawable/n"
    android:networkSecurityConfig="@xml/network_security_config"
    android:usesCleartextTraffic="false"
    tools:targetApi="n">
    <activity android:name=".MainActivity"
        android:exported="true"...>
</application>
```



```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <base-config cleartextTrafficPermitted="true" />
</network-security-config>
```

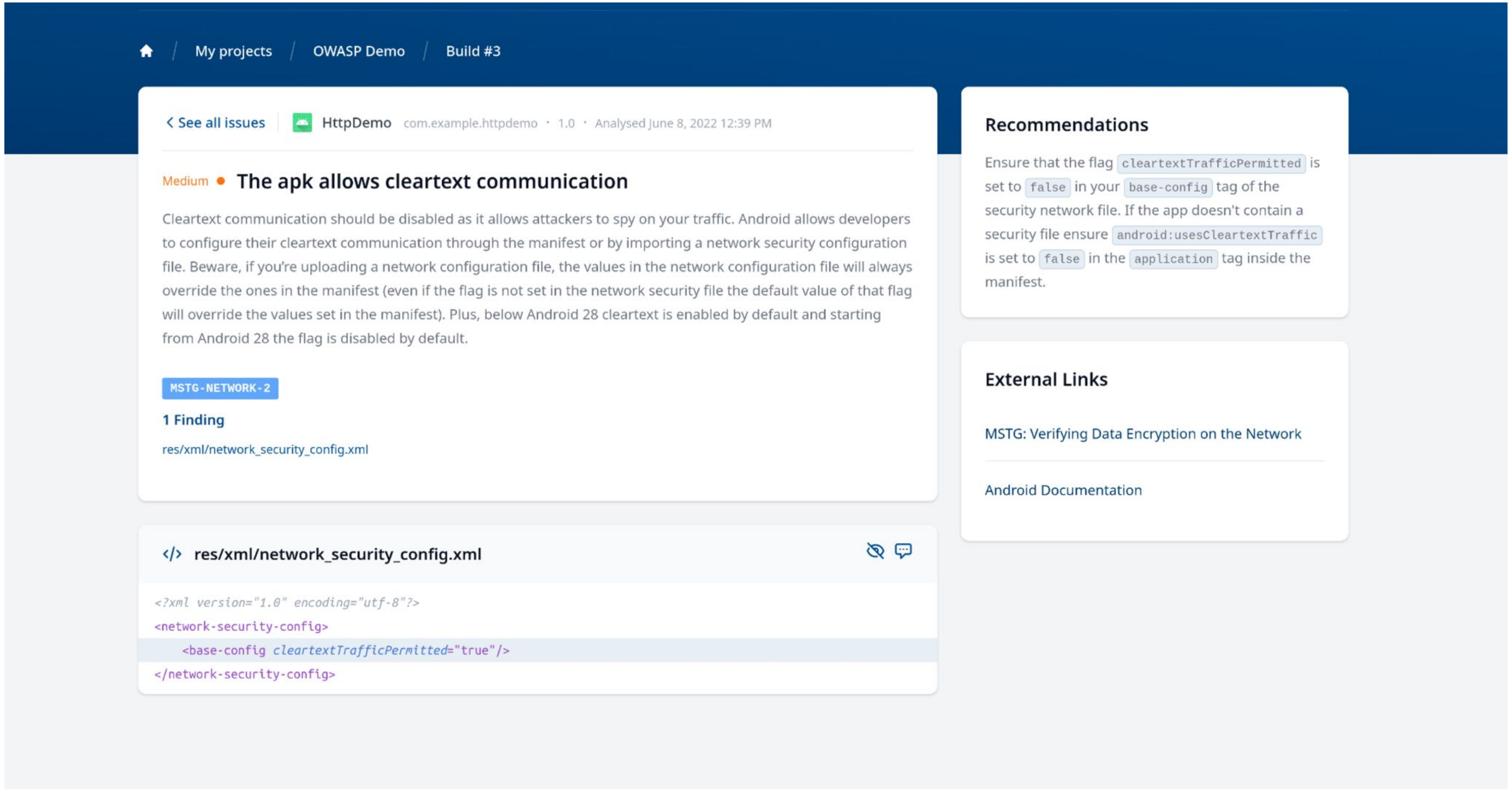


```
executor.execute(() -> handler.post(() -> {
    try {
        StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
        StrictMode.setThreadPolicy(policy);
        URL url = new URL( spec: "http://info.cern.ch/");
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        Log.i( tag: "UrlConnectionResponse", String.valueOf(connection.getResponseCode()));
    } catch (Exception e) {
        e.printStackTrace();
    }
}));
```



```
2022-06-03 14:19:01.259 16267-16267/? I/UrlConnectionResponse: 200
```

Cleartext Communications – **How to protect** 🙅‍♂️



My projects / OWASP Demo / Build #3

HttpDemo com.example.httpdemo · 1.0 · Analysed June 8, 2022 12:39 PM

Medium ● **The apk allows cleartext communication**

Cleartext communication should be disabled as it allows attackers to spy on your traffic. Android allows developers to configure their cleartext communication through the manifest or by importing a network security configuration file. Beware, if you're uploading a network configuration file, the values in the network configuration file will always override the ones in the manifest (even if the flag is not set in the network security file the default value of that flag will override the values set in the manifest). Plus, below Android 28 cleartext is enabled by default and starting from Android 28 the flag is disabled by default.

MSTG-NETWORK-2

1 Finding

res/xml/network_security_config.xml

Recommendations

Ensure that the flag `cleartextTrafficPermitted` is set to `false` in your `base-config` tag of the security network file. If the app doesn't contain a security file ensure `android:usesCleartextTraffic` is set to `false` in the `application` tag inside the manifest.

External Links

MSTG: Verifying Data Encryption on the Network

Android Documentation

</> res/xml/network_security_config.xml

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <base-config cleartextTrafficPermitted="true"/>
</network-security-config>
```

(Source: AppSweep)

Cleartext Communications - **How to protect** 🙅‍♂️

API: 28

```
android:networkSecurityConfig="@xml/network_security_config"
```

```
<application android:icon="@drawable/n"
    android:networkSecurityConfig="@xml/network_security_config"
    android:usesCleartextTraffic="false">
    <activity android:name=".MainActivity"
        android:exported="true"...>
</application>
```

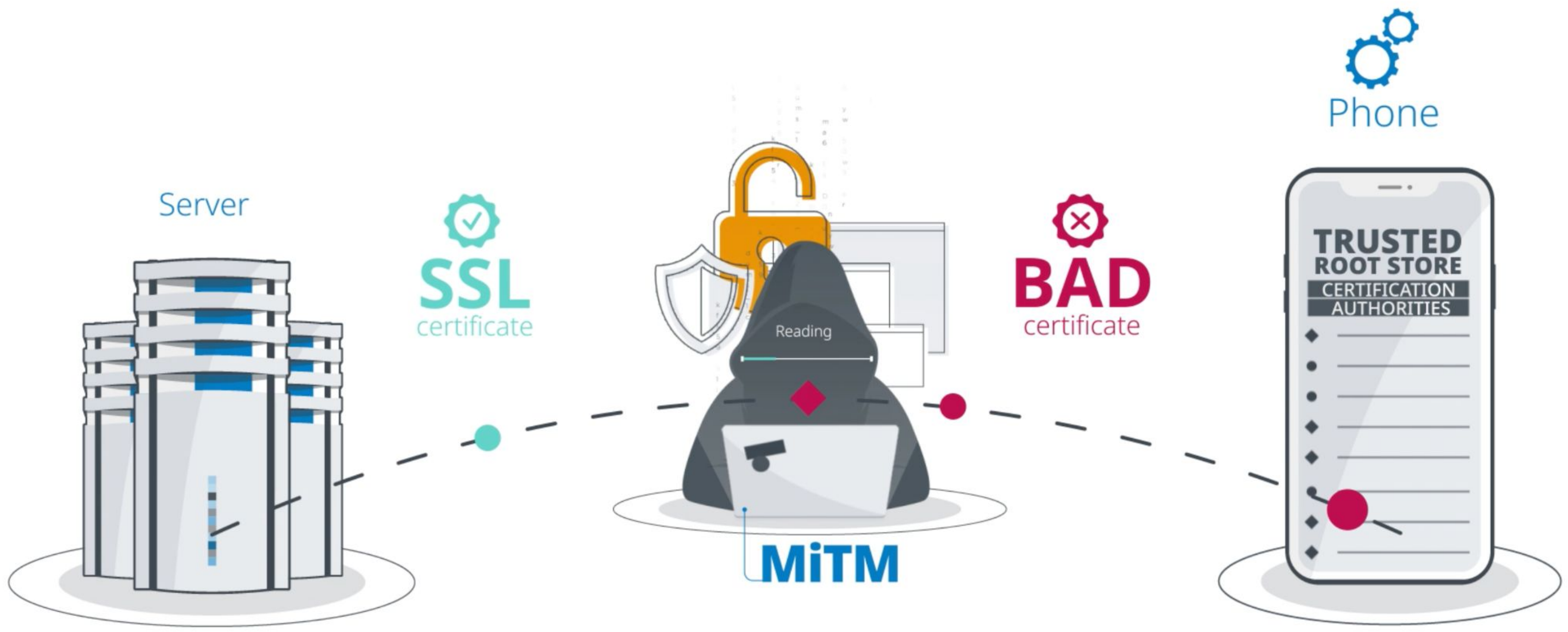
```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
</network-security-config>
```

```
executor.execute(() -> handler.post(() -> {
    try {
        StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
        StrictMode.setThreadPolicy(policy);
        URL url = new URL( spec: "http://info.cern.ch/");
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        Log.i( tag: "UrlConnectionResponse", String.valueOf(connection.getResponseCode()));
    } catch (Exception e) {
        e.printStackTrace();
    }
}));
```

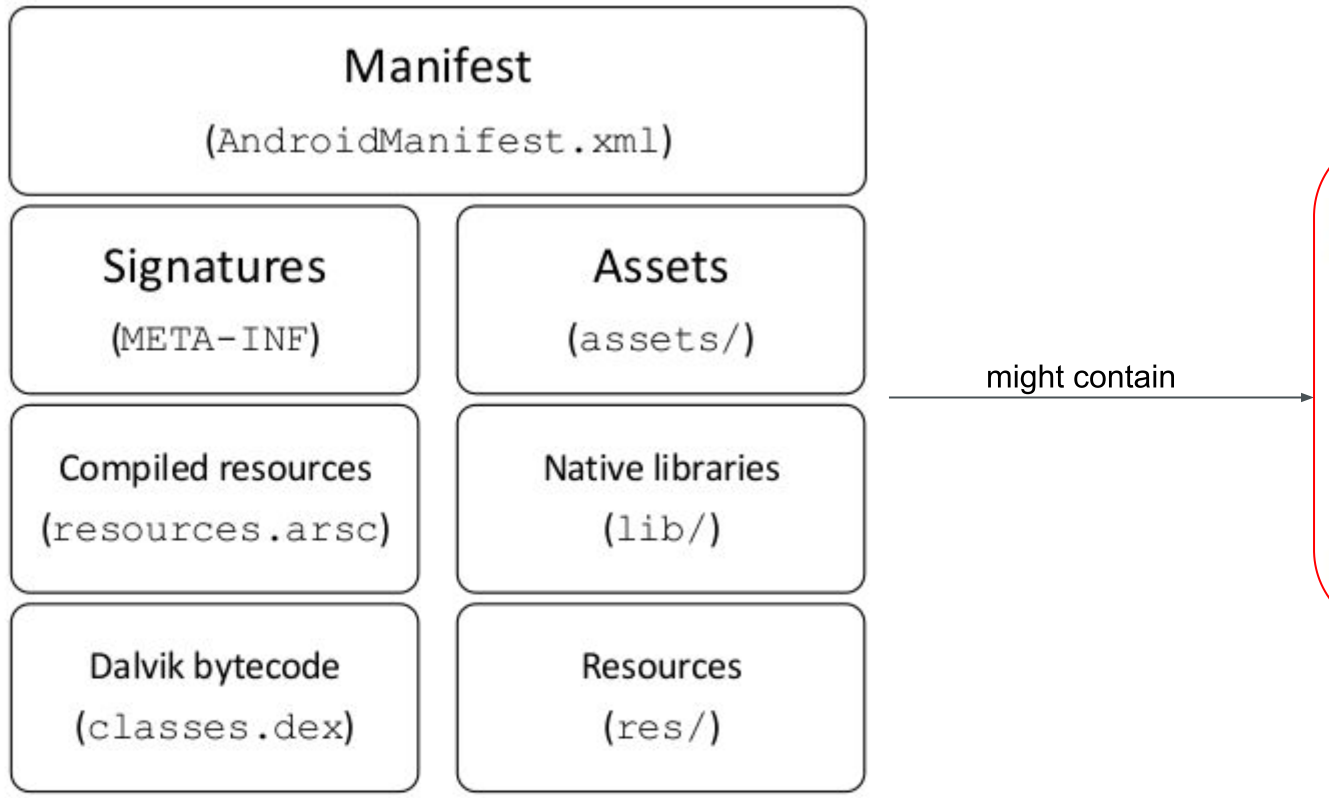
```
2022-06-03 16:13:34.462 6981-6981/? I/UrlConnectionResponse: FAILED
```

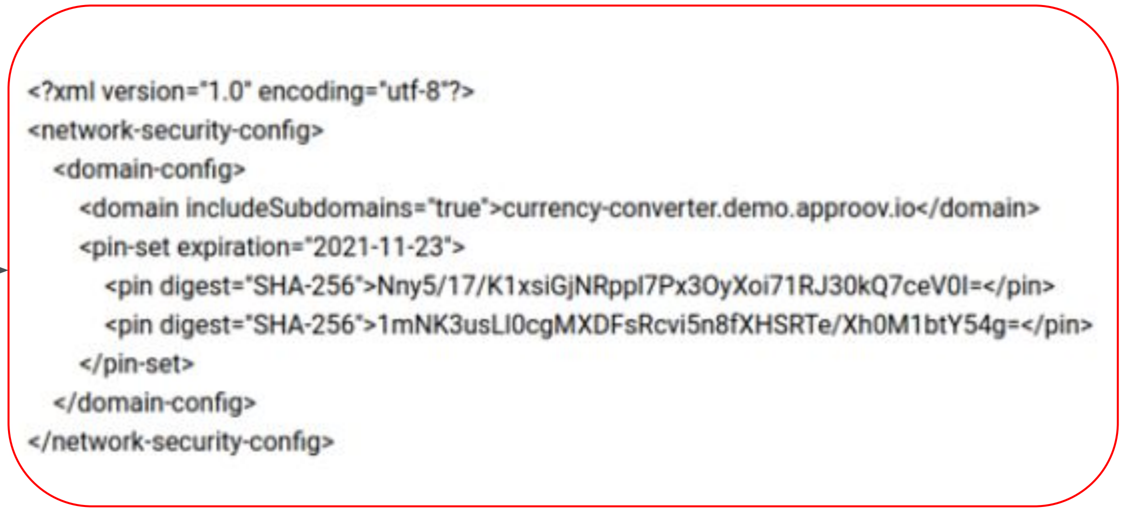
Missing backend attestation: **What is it?** 🤔

- Certificate pinning restricts which certificates are considered valid for a particular website, limiting risk. Instead of allowing any trusted certificate to be used, operators "pin" the certificate authority (CA) issuer(s), public keys or even end-entity certificates of their choice.



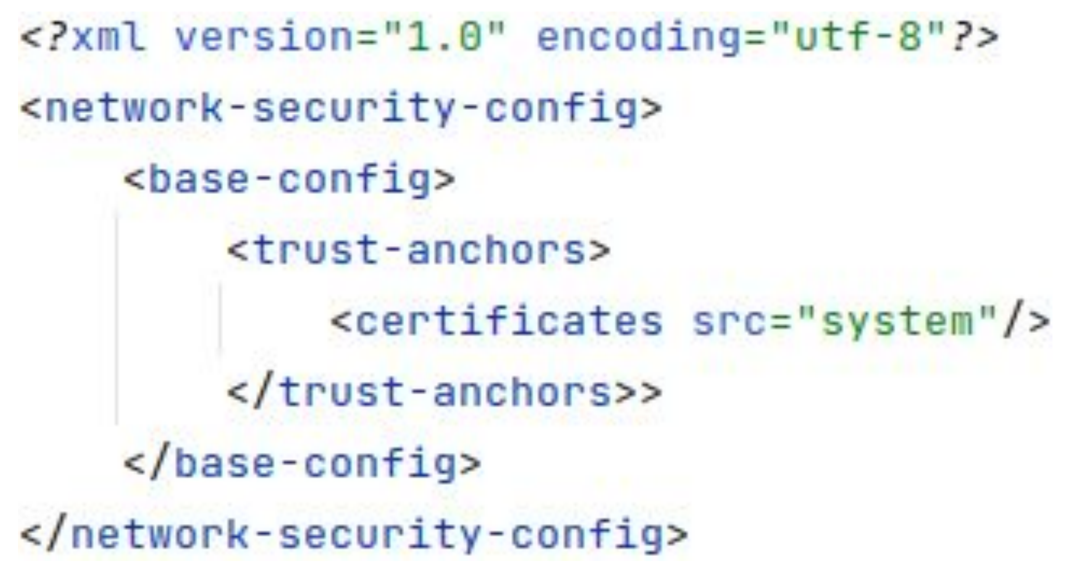
Missing backend attestation - **How to protect** 🙅‍♂️





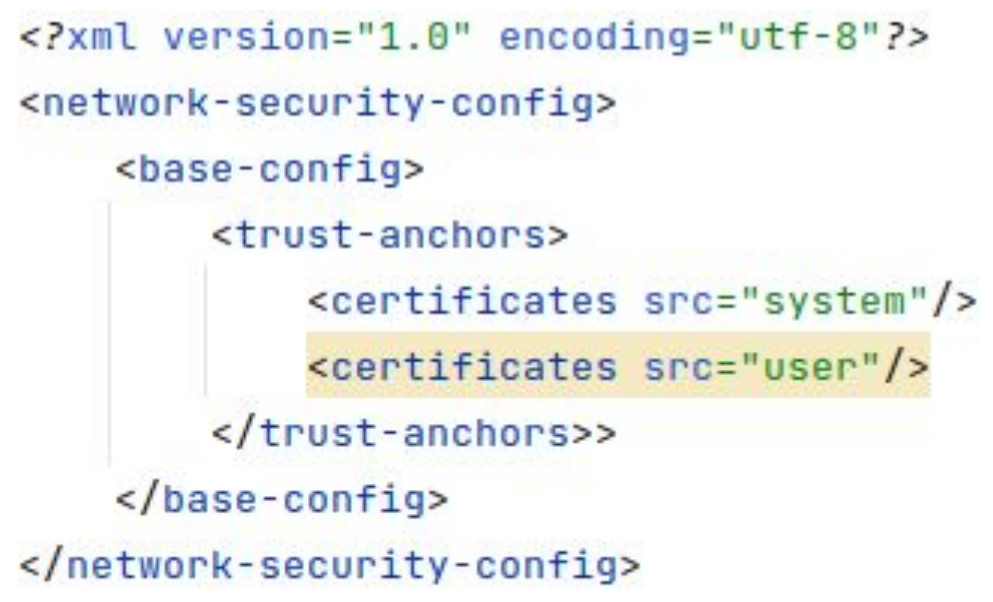
```xml
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
  <domain-config>
    <domain includeSubdomains="true">currency-converter.demo.approov.io</domain>
    <pin-set expiration="2021-11-23">
      <pin digest="SHA-256">Nny5/17/K1xsiGjNRppl7Px3OyXoi71RJ30kQ7ceV0I=</pin>
      <pin digest="SHA-256">1mNK3usLI0cgMXDFsRcvi5n8fXHSRTe/Xh0M1btY54g=</pin>
    </pin-set>
  </domain-config>
</network-security-config>
```

Missing backend attestation – **How to protect** 🙅‍♂️



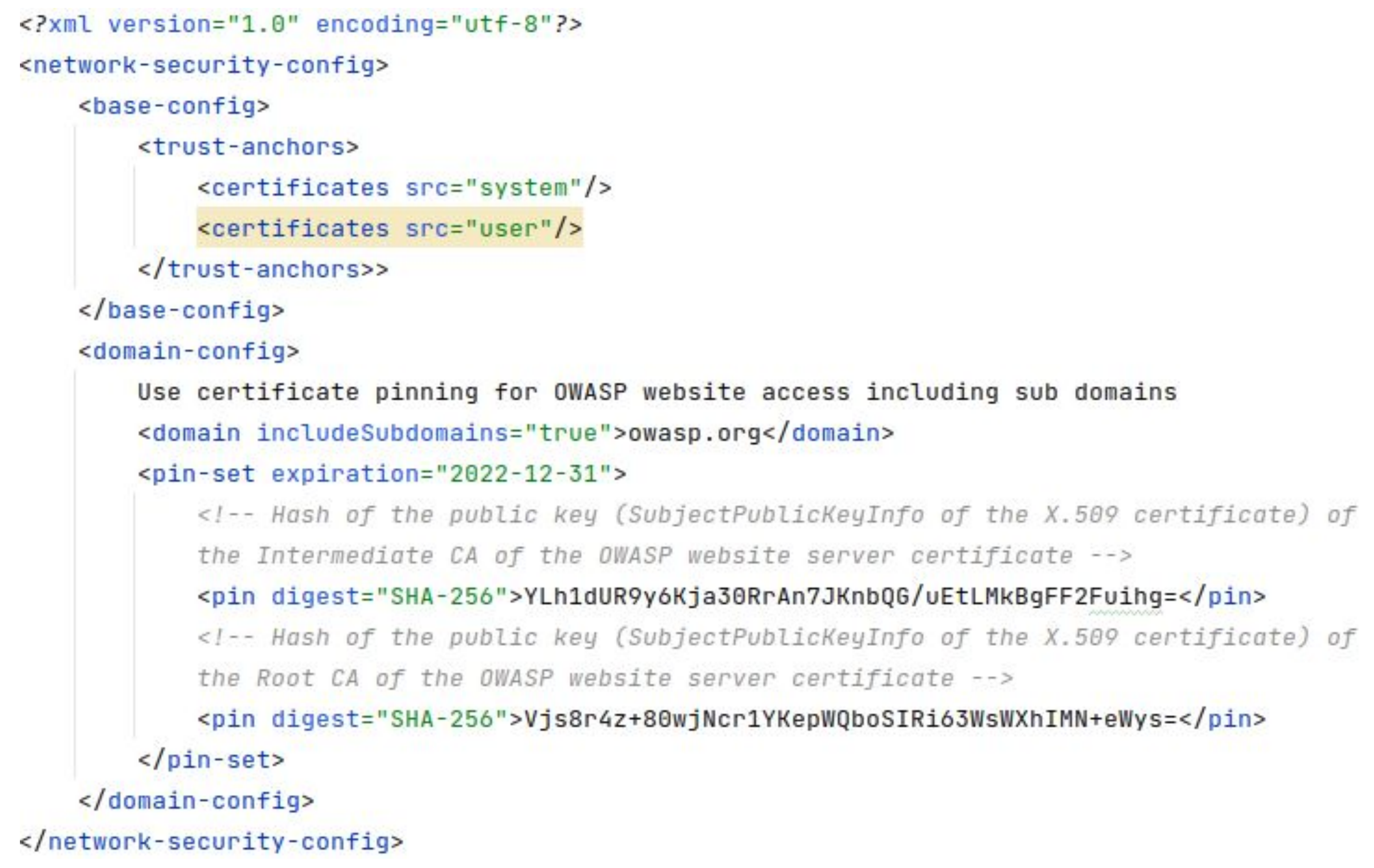
```xml
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <base-config>
        <trust-anchors>
            <certificates src="system"/>
        </trust-anchors>>
    </base-config>
</network-security-config>
```

Missing backend attestation - **How to protect** 🙅‍♂️



```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <base-config>
        <trust-anchors>
            <certificates src="system"/>
            <certificates src="user"/>
        </trust-anchors>>
    </base-config>
</network-security-config>
```

Missing backend attestation – **How to protect** 🙅‍♂️



```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <base-config>
        <trust-anchors>
            <certificates src="system"/>
            <certificates src="user"/>
        </trust-anchors>>
    </base-config>
    <domain-config>
        Use certificate pinning for OWASP website access including sub domains
        <domain includeSubdomains="true">owasp.org</domain>
        <pin-set expiration="2022-12-31">
            <!-- Hash of the public key (SubjectPublicKeyInfo of the X.509 certificate) of
            the Intermediate CA of the OWASP website server certificate -->
            <pin digest="SHA-256">YLh1dUR9y6Kja30RrAn7JKnbQG/uEtLMkBgFF2Fuihg=</pin>
            <!-- Hash of the public key (SubjectPublicKeyInfo of the X.509 certificate) of
            the Root CA of the OWASP website server certificate -->
            <pin digest="SHA-256">Vjs8r4z+80wjNcr1YKepWQboSIRi63WsWXhIMN+eWys=</pin>
        </pin-set>
    </domain-config>
</network-security-config>
```

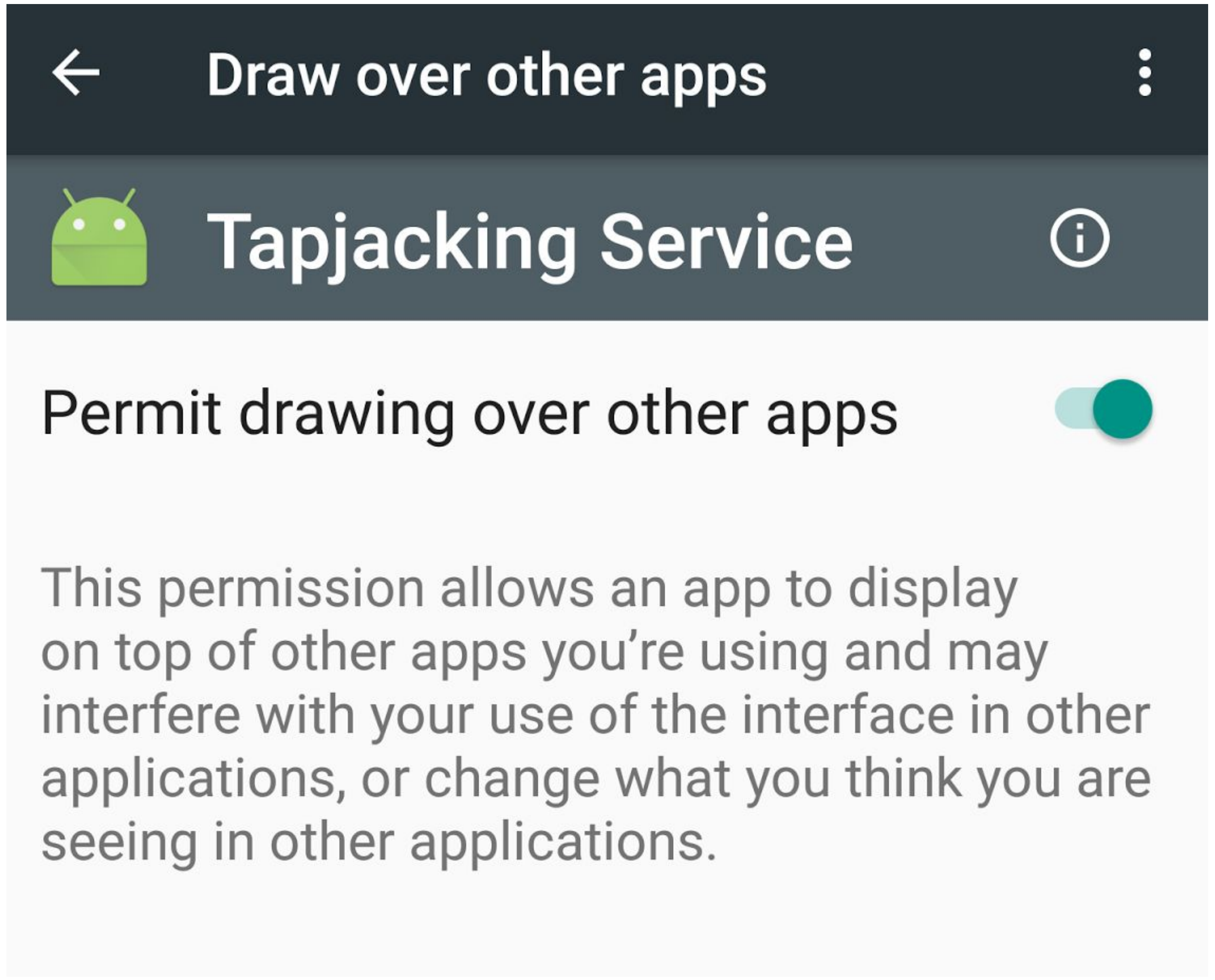
2. Android Platform

Tapjacking: **What is it?** 🤔

OWASP MSTG definition:

- Tapjacking abuses the screen overlay feature of Android listening for taps and intercepting any information being passed to the underlying activity.

Tapjacking: **What is it?** 🤔



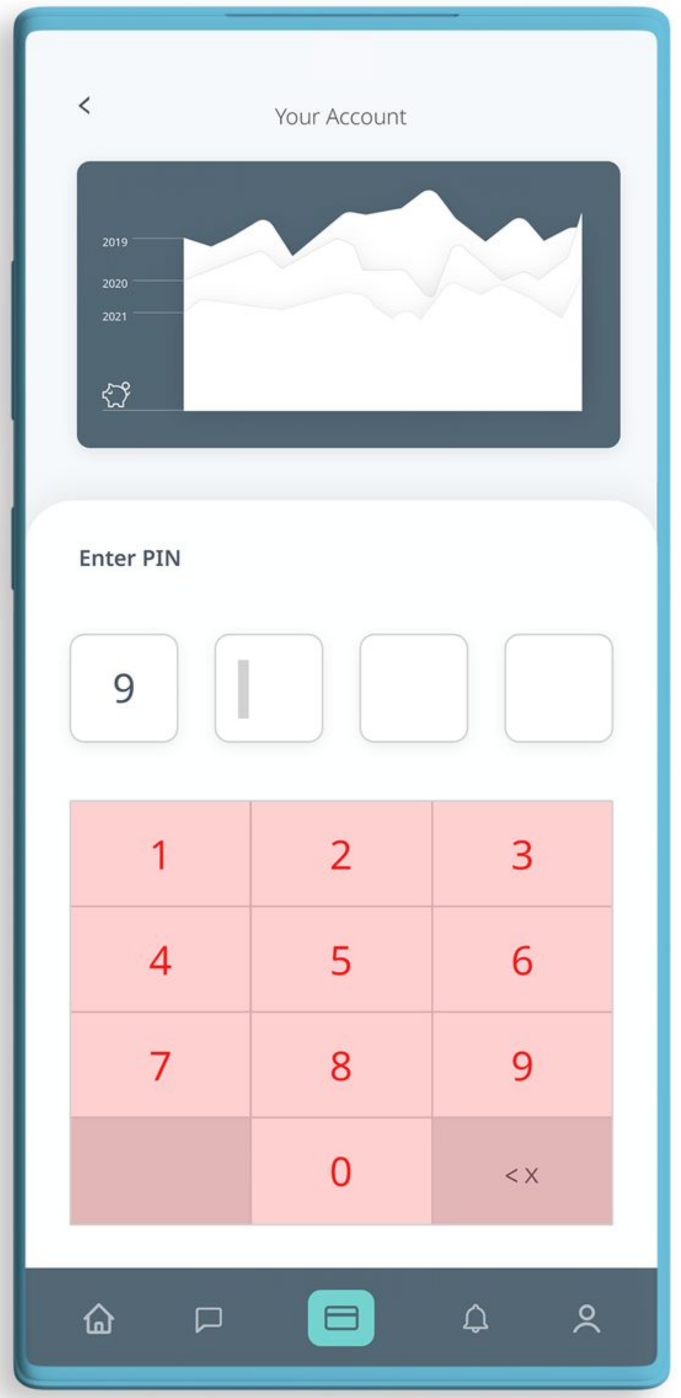
Draw over other apps

Tapjacking Service

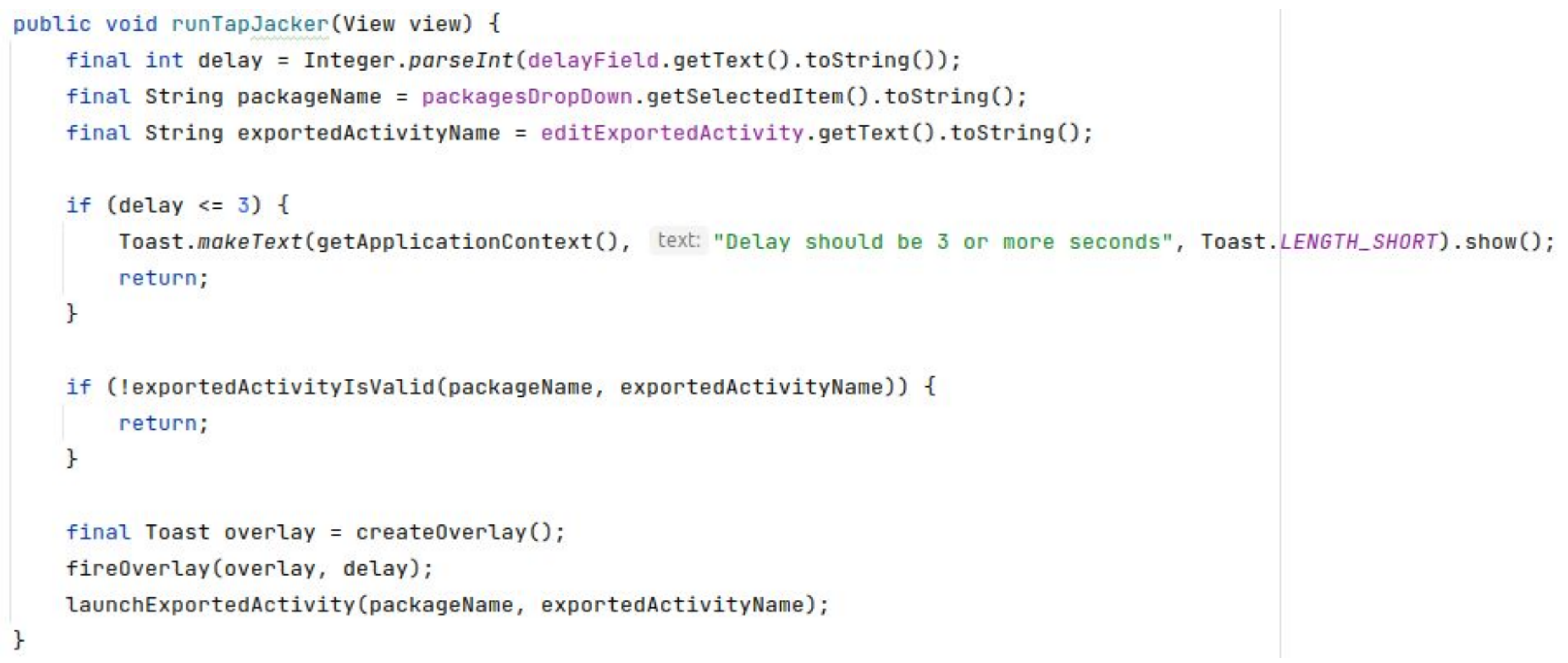
Permit drawing over other apps

This permission allows an app to display on top of other apps you're using and may interfere with your use of the interface in other applications, or change what you think you are seeing in other applications.

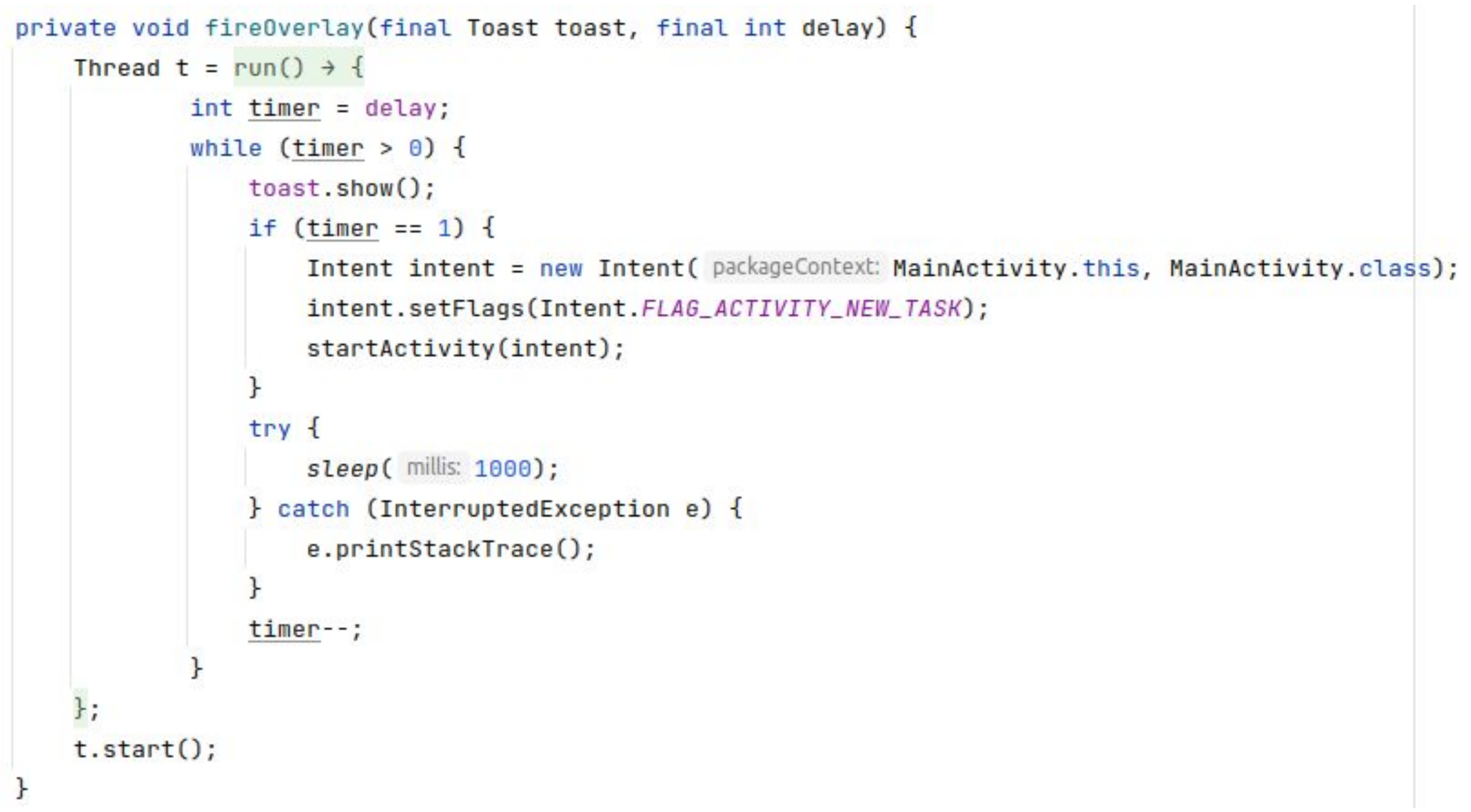
Tapjacking: **Visualization**



Tapjacking: **What is it?** 🤔

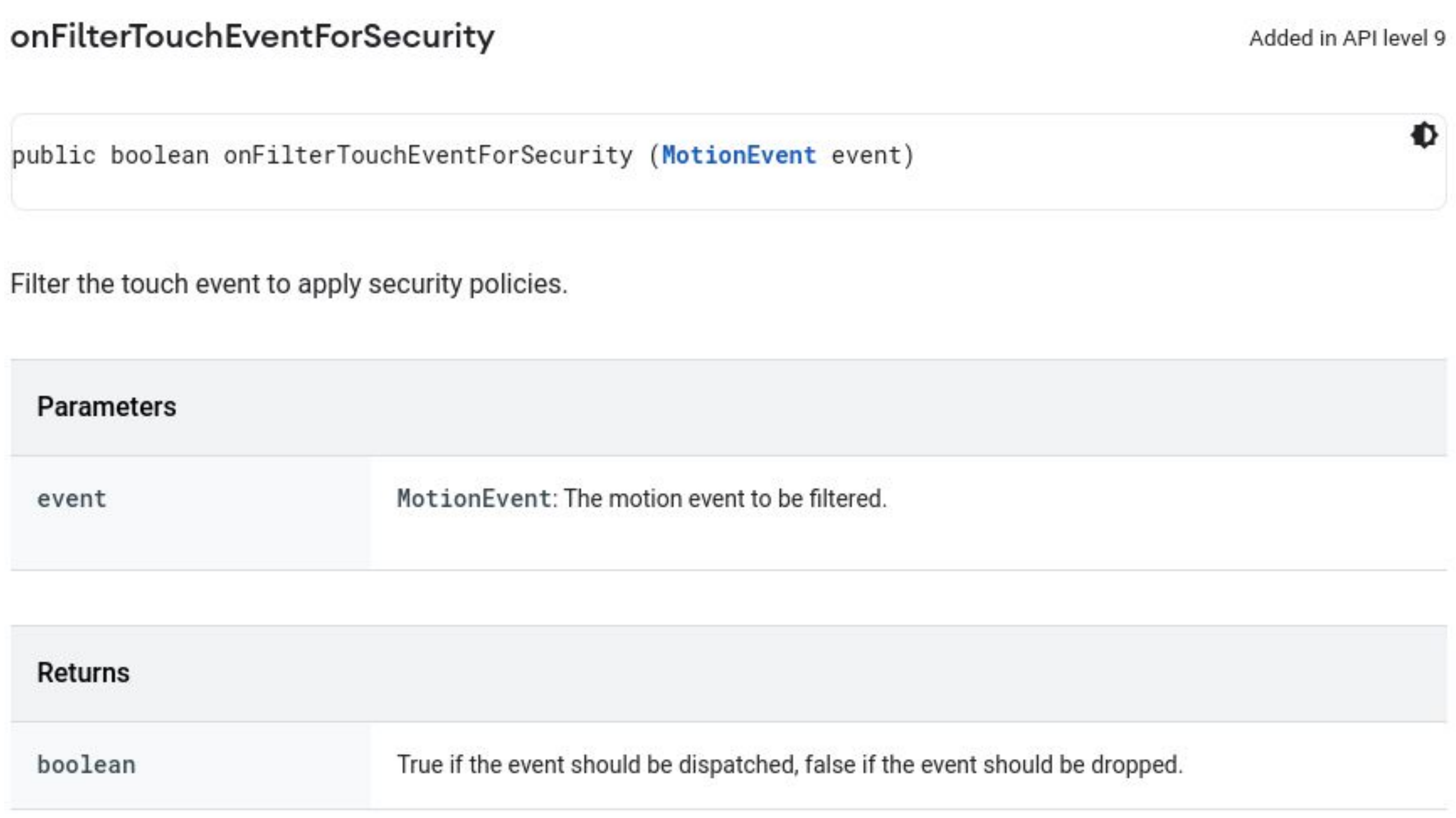


```java
public void runTapJacker(View view) {
    final int delay = Integer.parseInt(delayField.getText().toString());
    final String packageName = packagesDropDown.getSelectedItem().toString();
    final String exportedActivityName = editExportedActivity.getText().toString();

    if (delay <= 3) {
        Toast.makeText(getApplicationContext(),  text: "Delay should be 3 or more seconds", Toast.LENGTH_SHORT).show();
        return;
    }

    if (!exportedActivityIsValid(packageName, exportedActivityName)) {
        return;
    }

    final Toast overlay = createOverlay();
    fireOverlay(overlay, delay);
    launchExportedActivity(packageName, exportedActivityName);
}
```



```java
private void fireOverlay(final Toast toast, final int delay) {
    Thread t = run() → {
            int timer = delay;
            while (timer > 0) {
                toast.show();
                if (timer == 1) {
                    Intent intent = new Intent( packageContext: MainActivity.this, MainActivity.class);
                    intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                    startActivity(intent);
                }
                try {
                    sleep( millis: 1000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                timer--;
            }
    };
    t.start();
}
```

Tapjacking - How to protect 🙅‍♂️



onFilterTouchEventForSecurity

Added in API level 9

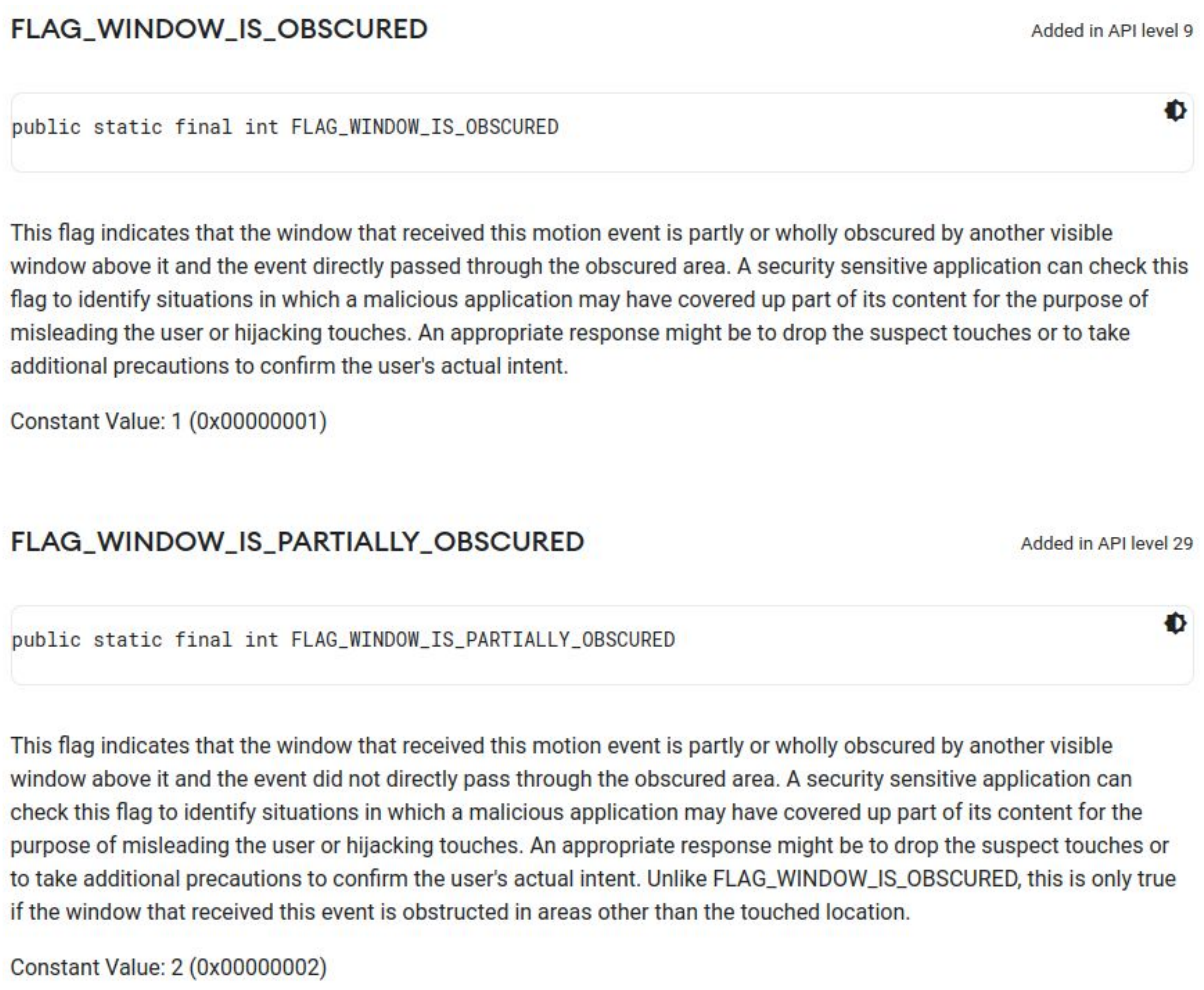
```
public boolean onFilterTouchEventForSecurity (MotionEvent event)
```

Filter the touch event to apply security policies.

Parameters	
`event`	`MotionEvent`: The motion event to be filtered.

Returns	
`boolean`	True if the event should be dispatched, false if the event should be dropped.

Tapjacking - How to protect 🙅‍♂️



FLAG_WINDOW_IS_OBSCURED

Added in API level 9

```
public static final int FLAG_WINDOW_IS_OBSCURED
```

This flag indicates that the window that received this motion event is partly or wholly obscured by another visible window above it and the event directly passed through the obscured area. A security sensitive application can check this flag to identify situations in which a malicious application may have covered up part of its content for the purpose of misleading the user or hijacking touches. An appropriate response might be to drop the suspect touches or to take additional precautions to confirm the user's actual intent.

Constant Value: 1 (0x00000001)

FLAG_WINDOW_IS_PARTIALLY_OBSCURED

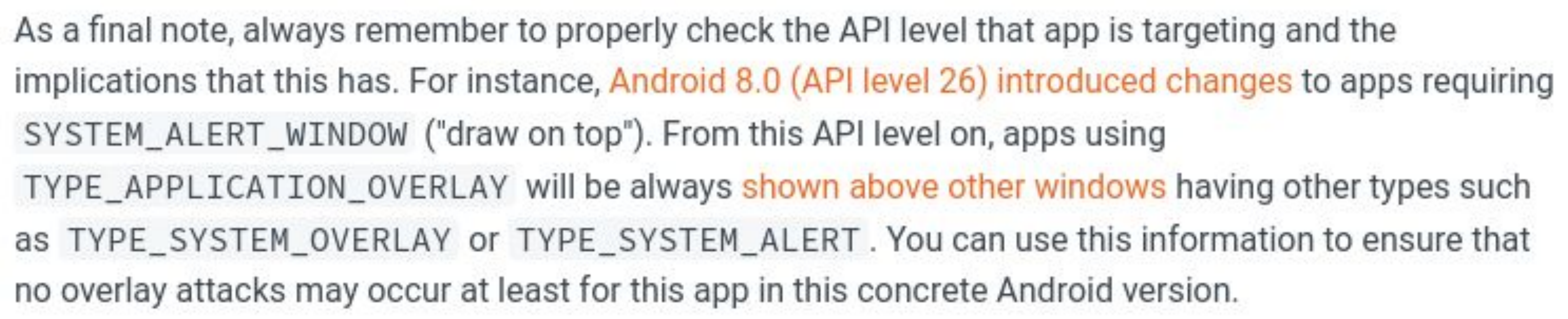
Added in API level 29

```
public static final int FLAG_WINDOW_IS_PARTIALLY_OBSCURED
```

This flag indicates that the window that received this motion event is partly or wholly obscured by another visible window above it and the event did not directly pass through the obscured area. A security sensitive application can check this flag to identify situations in which a malicious application may have covered up part of its content for the purpose of misleading the user or hijacking touches. An appropriate response might be to drop the suspect touches or to take additional precautions to confirm the user's actual intent. Unlike FLAG_WINDOW_IS_OBSCURED, this is only true if the window that received this event is obstructed in areas other than the touched location.

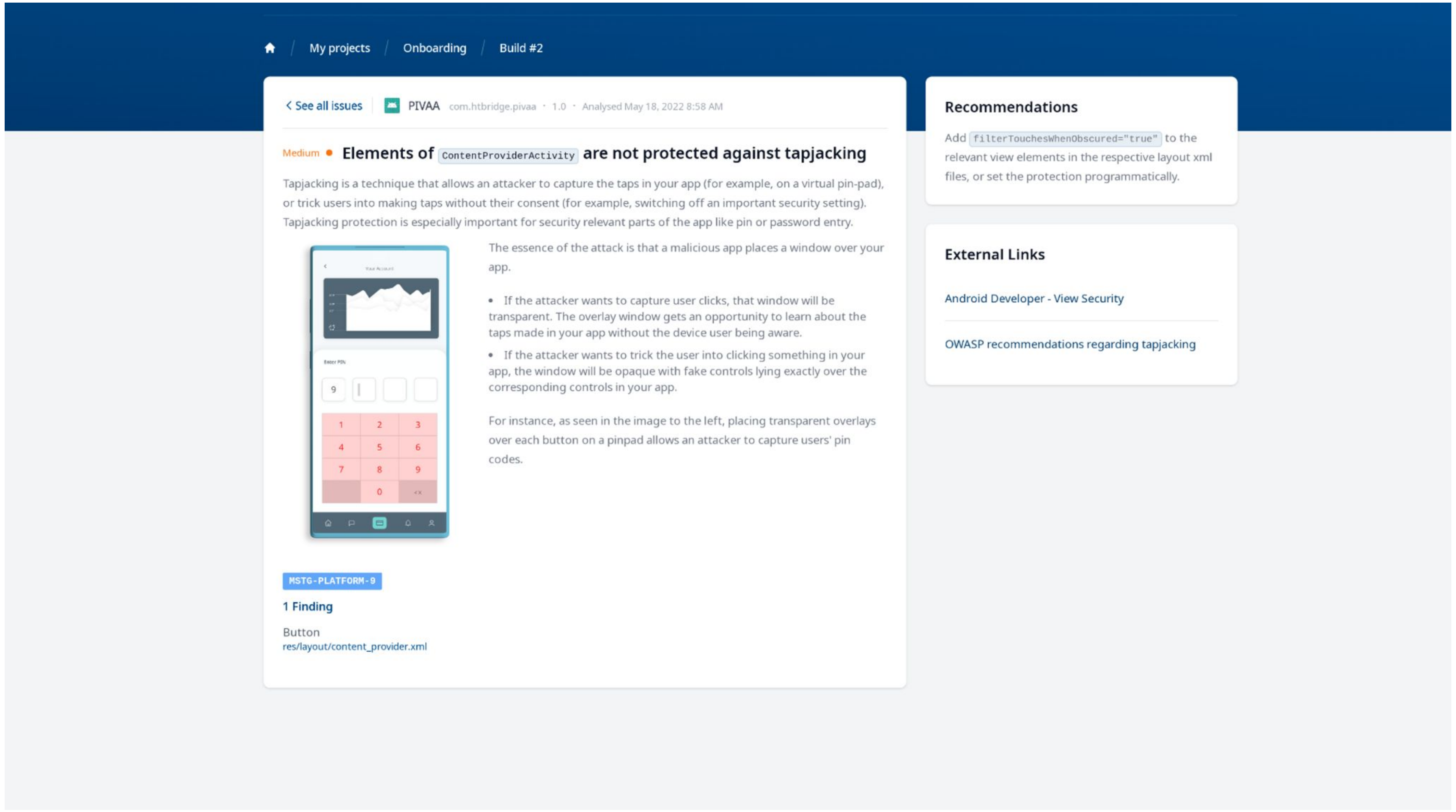
Constant Value: 2 (0x00000002)

Tapjacking - **How to protect** 🙅‍♂️



As a final note, always remember to properly check the API level that app is targeting and the implications that this has. For instance, Android 8.0 (API level 26) introduced changes to apps requiring `SYSTEM_ALERT_WINDOW` ("draw on top"). From this API level on, apps using `TYPE_APPLICATION_OVERLAY` will be always shown above other windows having other types such as `TYPE_SYSTEM_OVERLAY` or `TYPE_SYSTEM_ALERT`. You can use this information to ensure that no overlay attacks may occur at least for this app in this concrete Android version.

Tapjacking – How to protect 🙅‍♂️



My projects / Onboarding / Build #2

< See all issues | PIVAA com.htbridge.pivaa · 1.0 · Analysed May 18, 2022 8:58 AM

Medium ● **Elements of `ContentProviderActivity` are not protected against tapjacking**

Tapjacking is a technique that allows an attacker to capture the taps in your app (for example, on a virtual pin-pad), or trick users into making taps without their consent (for example, switching off an important security setting). Tapjacking protection is especially important for security relevant parts of the app like pin or password entry.

The essence of the attack is that a malicious app places a window over your app.

- If the attacker wants to capture user clicks, that window will be transparent. The overlay window gets an opportunity to learn about the taps made in your app without the device user being aware.
- If the attacker wants to trick the user into clicking something in your app, the window will be opaque with fake controls lying exactly over the corresponding controls in your app.

For instance, as seen in the image to the left, placing transparent overlays over each button on a pinpad allows an attacker to capture users' pin codes.

MSTG-PLATFORM-9

1 Finding

Button
res/layout/content_provider.xml

Recommendations

Add `filterTouchesWhenObscured="true"` to the relevant view elements in the respective layout xml files, or set the protection programmatically.

External Links

Android Developer - View Security

OWASP recommendations regarding tapjacking

(Source: AppSweep)

3. Android Data Storage

Sensitive data disclosure: **What is it?** 🤔

- One of the focus points around the MSTG guide is how sensitive data is handled and secured.
- What is sensitive data?
- To which places should we be careful about sending sensitive data?

Sensitive data disclosure: **What is sensitive data?** 🤔

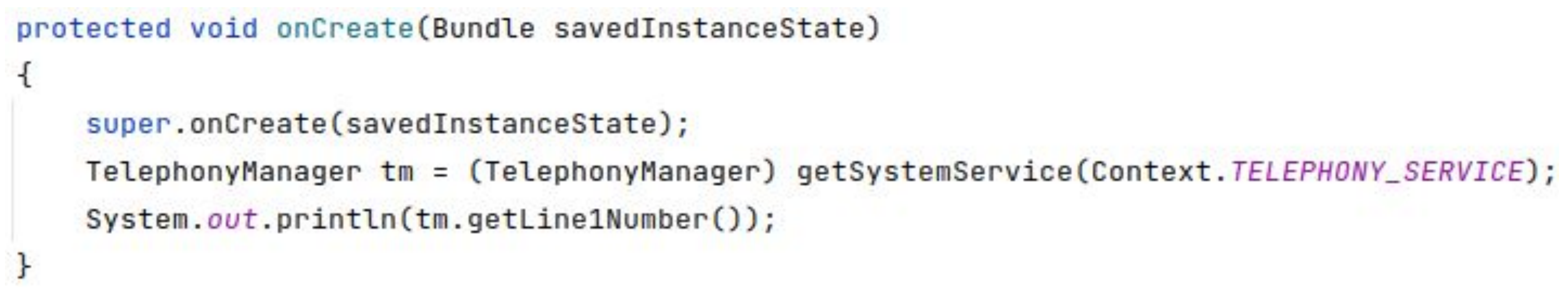
- Usually some data classification policy is in place

- When no policy is in place the following list is generally considered sensitive:
    - user authentication information (credentials, PINs, etc...)
    - personally identifiable information (PII) that can be abused for identity theft: social security numbers, credit card numbers, bank account numbers, health information
    - device identifiers that may identify a person
    - highly sensitive data whose compromise would lead to reputational harm and/or financial costs
    - any data whose protection is a legal obligation
    - any technical data generated by the application (or related systems) and used to protect other data or the system itself (e.g. encryption keys)

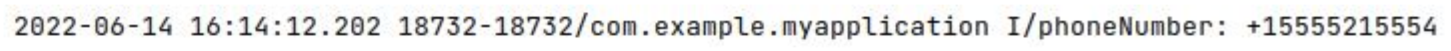
Sensitive data disclosure: **Places to be careful sending data** 🤔

- Internal Storage (SharedPreferences, Databases, Files, etc.)
- External Storage (Amazon S3, Google Cloud, etc.)
- Databases
- Keystore
- Logs
- Backups
- IPC Mechanisms
- External APIs

Sensitive data disclosure: **Logging**



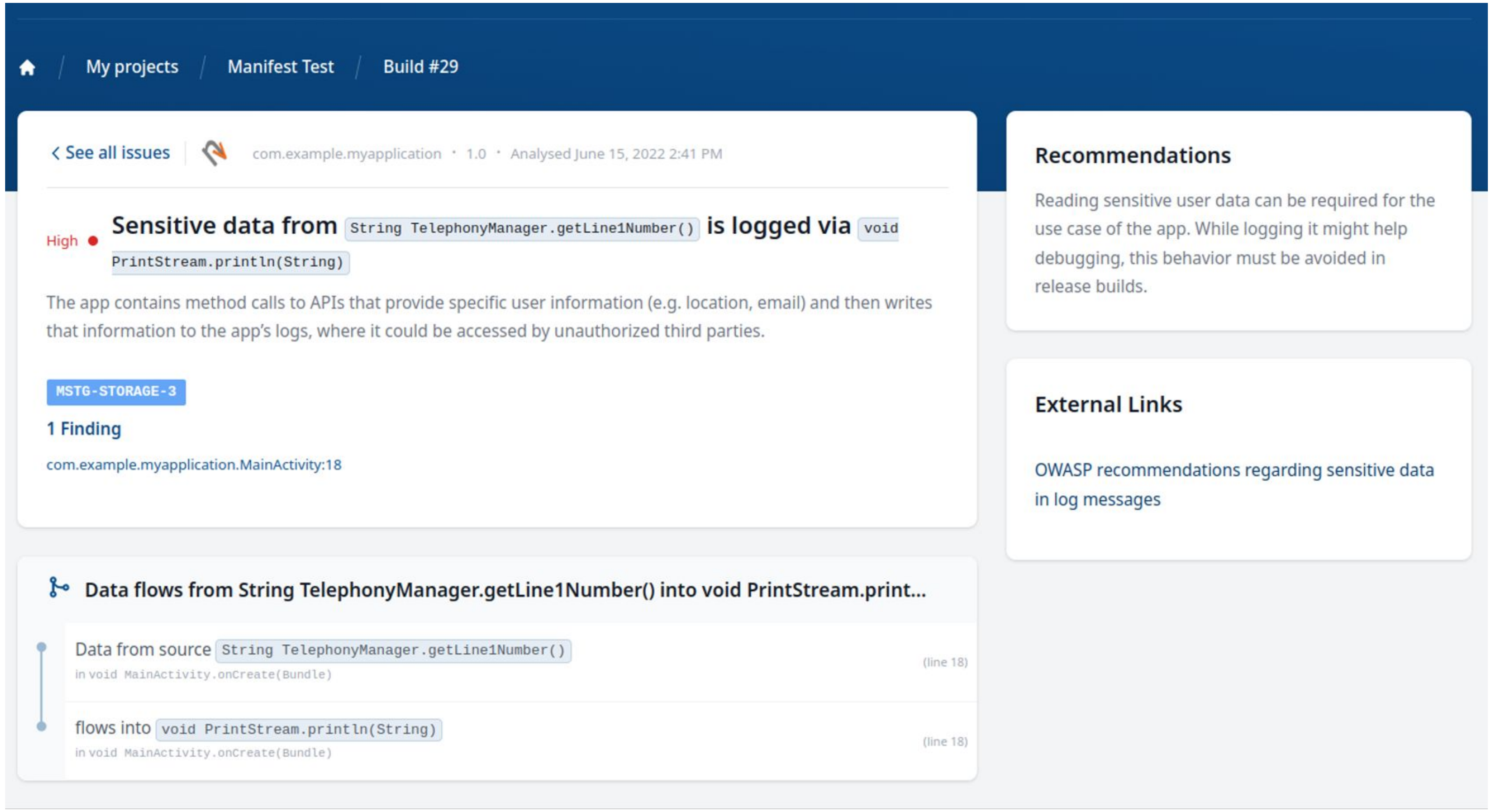
```java
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    TelephonyManager tm = (TelephonyManager) getSystemService(Context.TELEPHONY_SERVICE);
    System.out.println(tm.getLine1Number());
}
```



```
2022-06-14 16:14:12.202 18732-18732/com.example.myapplication I/phoneNumber: +15555215554
```

Sensitive data disclosure: **Logging**

(Source: AppSweep)



My projects / Manifest Test / Build #29

< See all issues | com.example.myapplication · 1.0 · Analysed June 15, 2022 2:41 PM

High ● **Sensitive data from** `String TelephonyManager.getLine1Number()` **is logged via** `void PrintStream.println(String)`

The app contains method calls to APIs that provide specific user information (e.g. location, email) and then writes that information to the app's logs, where it could be accessed by unauthorized third parties.

MSTG-STORAGE-3

1 Finding

com.example.myapplication.MainActivity:18

Recommendations

Reading sensitive user data can be required for the use case of the app. While logging it might help debugging, this behavior must be avoided in release builds.

External Links

OWASP recommendations regarding sensitive data in log messages

Data flows from String TelephonyManager.getLine1Number() into void PrintStream.print...

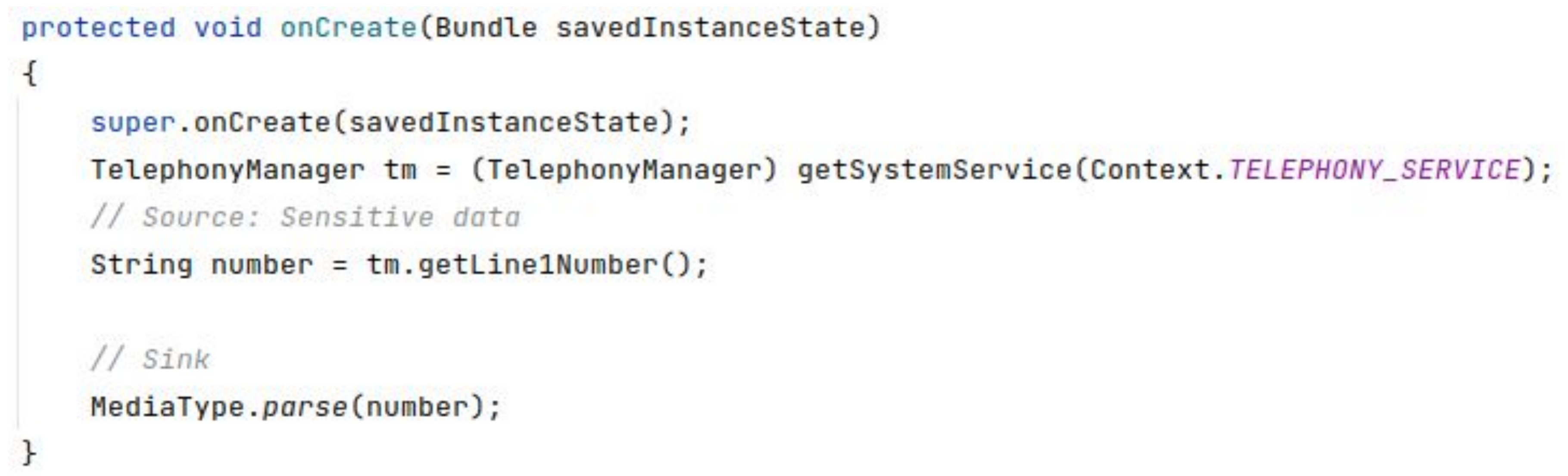
Data from source `String TelephonyManager.getLine1Number()` (line 18)
in `void MainActivity.onCreate(Bundle)`

flows into `void PrintStream.println(String)` (line 18)
in `void MainActivity.onCreate(Bundle)`

Logging - How to protect 🙅‍♂️

- In general if you’re handling sensitive data you shouldn’t pass it to a logging function.
- Apply obfuscation tools that will remove these instructions automatically.

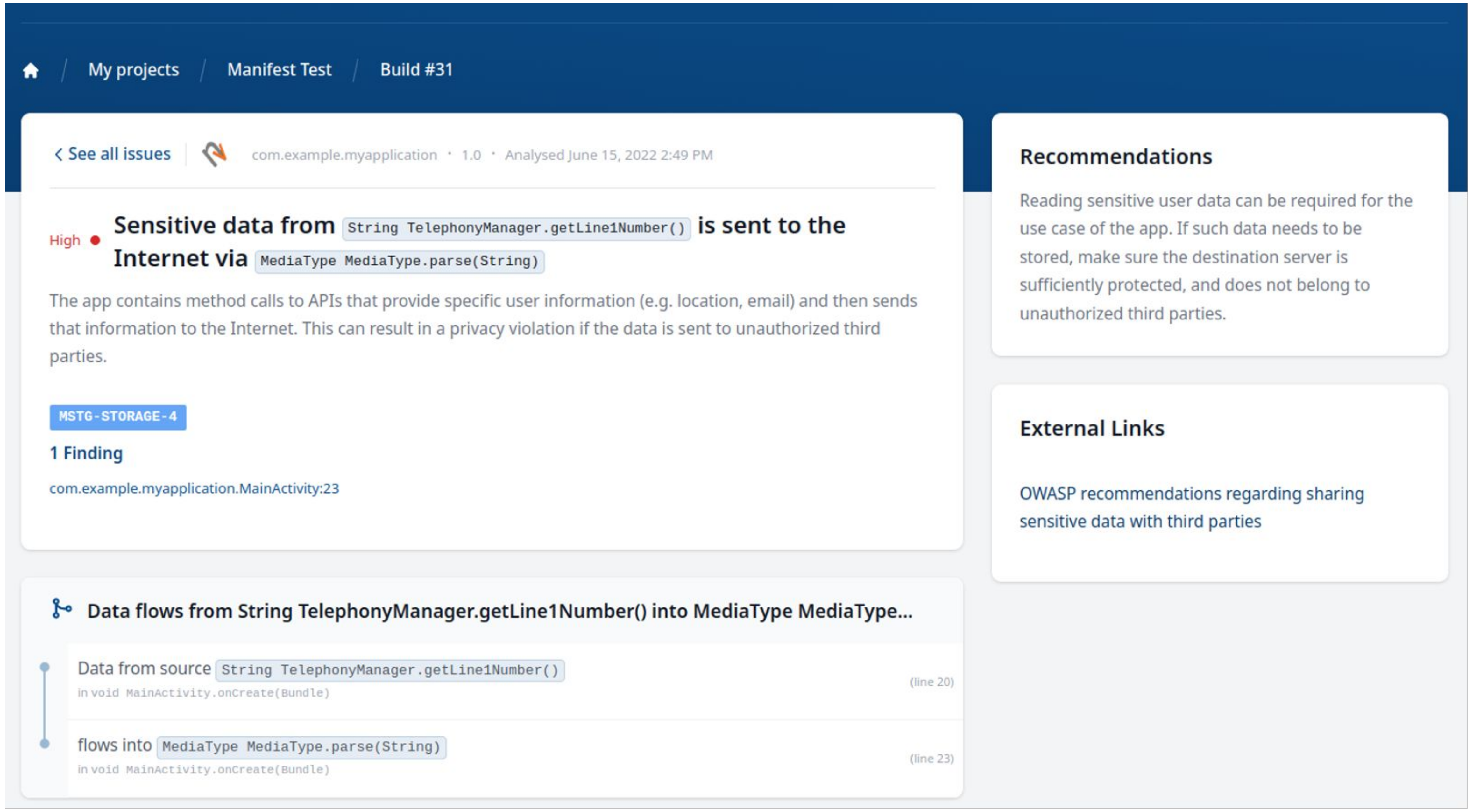
Sensitive data disclosure: **Send data to web**



```
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    TelephonyManager tm = (TelephonyManager) getSystemService(Context.TELEPHONY_SERVICE);
    // Source: Sensitive data
    String number = tm.getLine1Number();

    // Sink
    MediaType.parse(number);
}
```

Sensitive data disclosure: **Send data to web**

(Source: AppSweep)



com.example.myapplication · 1.0 · Analysed June 15, 2022 2:49 PM

High ● **Sensitive data from `String TelephonyManager.getLine1Number()` is sent to the Internet via `MediaType MediaType.parse(String)`**

The app contains method calls to APIs that provide specific user information (e.g. location, email) and then sends that information to the Internet. This can result in a privacy violation if the data is sent to unauthorized third parties.

MSTG-STORAGE-4

1 Finding

com.example.myapplication.MainActivity:23

Recommendations

Reading sensitive user data can be required for the use case of the app. If such data needs to be stored, make sure the destination server is sufficiently protected, and does not belong to unauthorized third parties.

External Links

OWASP recommendations regarding sharing sensitive data with third parties

Data flows from String TelephonyManager.getLine1Number() into MediaType MediaType...

Data from source `String TelephonyManager.getLine1Number()` (line 20)
in void `MainActivity.onCreate(Bundle)`

flows into `MediaType MediaType.parse(String)` (line 23)
in void `MainActivity.onCreate(Bundle)`

Send data to web - **How to protect** 🙅‍♂️

- Never share sensitive information into app notifications

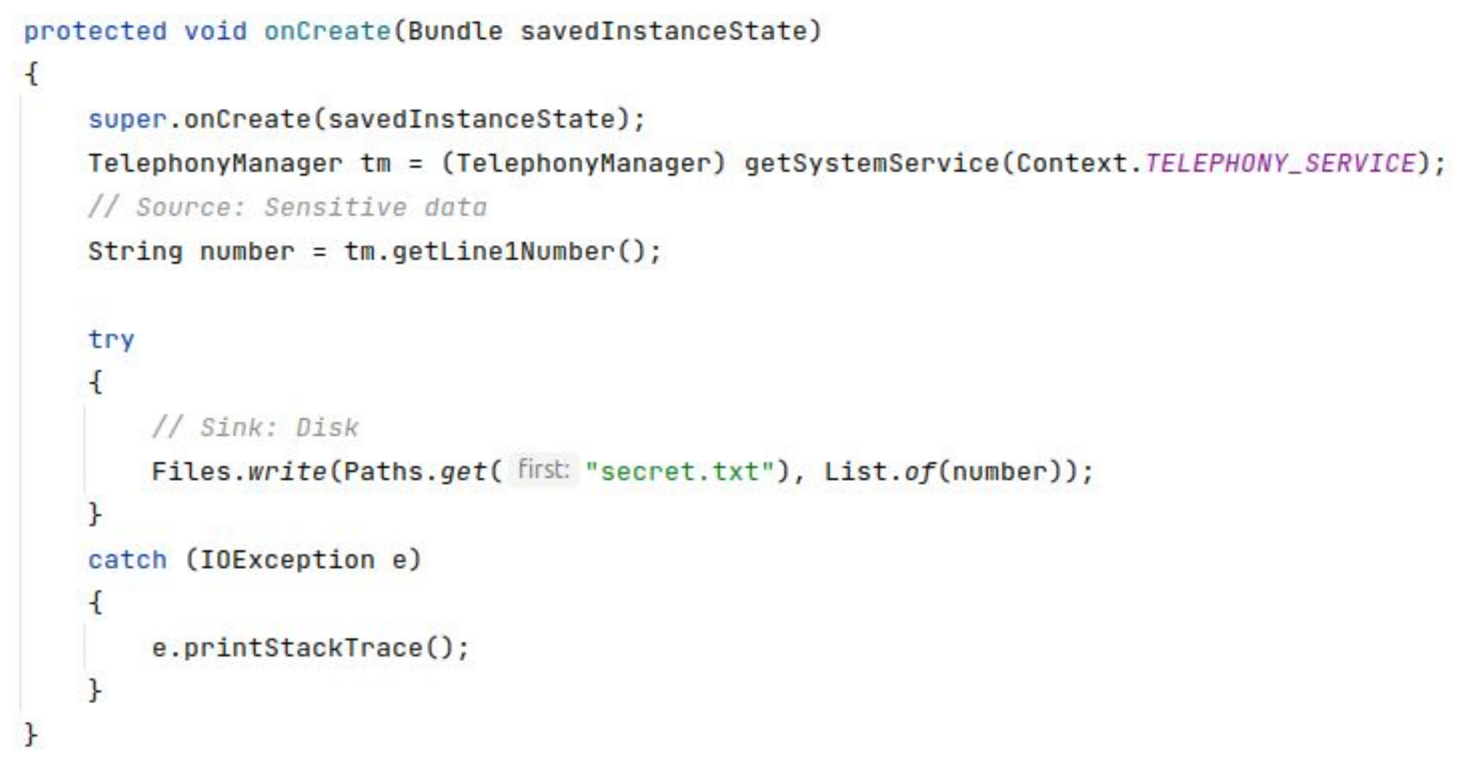
- Analyse third party-libraries being used by the application
  - Check libraries are being used according to best practices
  - If possible, review their source code
  - If not possible, run them through a static analysis tool (like dependency-check-gradle) from OWASP
  - Verify online for known vulnerabilities

- All data that’s sent to third-party services should be anonymized to prevent exposure of PII (Personal Identifiable Information)

- When communicating with 3rd parties ensure that the connection is encrypted

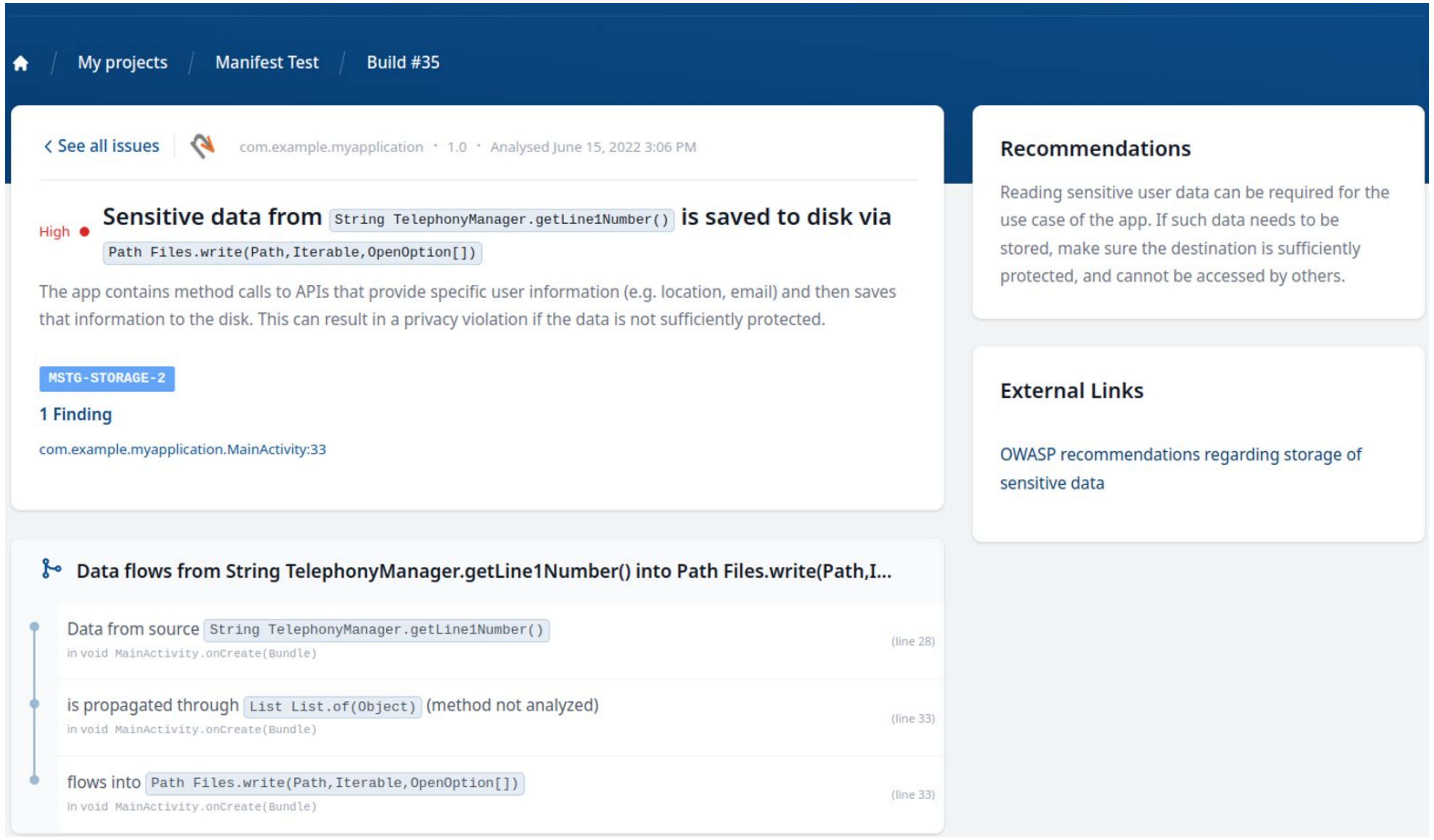
- If possible encrypt the data before sending

Sensitive data disclosure: **Writing data to disk**



```
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    TelephonyManager tm = (TelephonyManager) getSystemService(Context.TELEPHONY_SERVICE);
    // Source: Sensitive data
    String number = tm.getLine1Number();

    try
    {
        // Sink: Disk
        Files.write(Paths.get( first: "secret.txt"), List.of(number));
    }
    catch (IOException e)
    {
        e.printStackTrace();
    }
}
```

Sensitive data disclosure: **Writing data to disk**



My projects / Manifest Test / Build #35

See all issues · com.example.myapplication · 1.0 · Analysed June 15, 2022 3:06 PM

High ● **Sensitive data from `String TelephonyManager.getLine1Number()` is saved to disk via `Path Files.write(Path,Iterable,OpenOption[])`**

The app contains method calls to APIs that provide specific user information (e.g. location, email) and then saves that information to the disk. This can result in a privacy violation if the data is not sufficiently protected.

MSTG-STORAGE-2

1 Finding

com.example.myapplication.MainActivity:33

Data flows from String TelephonyManager.getLine1Number() into Path Files.write(Path,I...

Data from source `String TelephonyManager.getLine1Number()` (line 28)
in `void MainActivity.onCreate(Bundle)`

is propagated through `List List.of(Object)` (method not analyzed) (line 33)
in `void MainActivity.onCreate(Bundle)`

flows into `Path Files.write(Path,Iterable,OpenOption[])` (line 33)
in `void MainActivity.onCreate(Bundle)`

Recommendations

Reading sensitive user data can be required for the use case of the app. If such data needs to be stored, make sure the destination is sufficiently protected, and cannot be accessed by others.

External Links

OWASP recommendations regarding storage of sensitive data

(Source: AppSweep)

Writing data to disk – **How to protect** 🙅‍♂️

- Check AndroidManifest.xml for correct read/write external storage permissions (e.g. uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE")

- Check the source code for keywords and API calls that are used to store data
  - File permissions such as:
    - MODE_WORLD_READABLE or MODE_WORLD_WRITABLE: These flags should be avoided since any app will be able to read from or write to the files even if the files are stored in the app private data directory
  - Classes and functions such as:
    - SharedPreferences class (stores key-values pairs)
    - FileOutputStream class (uses internal or external storage)
    - getExternal* functions (use external storage)
    - getWritableDatabase functions (returns a SQLiteDatabase for writing)
    - getReadableDatabase functions (returns a SQLiteDatabase for reading)
    - getCacheDir and getExternalCacheDirs functions (use cached files)

- Encryption should be implemented using proven SDK functions, however beware of the following bad practices:
  - Locally stored sensitive information encrypted via simple bit operations like XOR or bit flipping.
  - Keys used or created without Android onboard feature, such as the Android KeyStore
  - Keys disclosed by hard-coding

Thank **you!**

Questions? Please ask us!

Andre Carvalho is a software engineer at Guardsquare.

With a background in the consulting field, he is part of the AppSweep development team and he is currently focusing on OWASP and MAST.
andre.carvalho@guardsquare.com

Daniel Schwendner is a software developer at Guardsquare and master's student at TUM.

He is a cyber security enthusiast: with a background in software development and DevOps, his focus is on mobile application security in the AppSweep development team.
daniel.schwendner@guardsquare.com