



Playing with your code: a new approach to avoid potential hackers from doing exactly this!

Hugues Thiebeauld

*Android Security Symposium
Vienna – 8th of March*



Mobile application goes to a hacker
How do I know if this will be tough for him



Mobile application tools



*focus on finding vulnerabilities,
virus, malware*

*we want to know if the
armour is strong*



Mobile application tools



*focus on finding vulnerabilities,
virus, malware*

*we want to know if the
armour is strong*



Solving a critical problem



Security protections are per nature “transparent”

How confident the protections have been effectively enabled?



Protected mobile app

No other choice than hiring a security analyst to manually inspect

Are protections enabled at the right level?

Manual inspection



```
package at.univie.enaensorium;

import android.content.Context;
import android.content.SharedPreferences;
import android.preference.PreferenceManager;
import android.text.Html;
import android.util.Log;
import android.widget.TextView;
import android.widget.TextView.BufferType;
import at.univie.sensorium.extensions.interfaces.XMLRPCSensorServerThread;
import at.univie.sensorium.logging.JSONLogger;
import at.univie.sensorium.preferences.Preferences;
import at.univie.sensorium.privacy.Privacy;
import at.univie.sensorium.sensors.AbstractSensor;
import at.univie.sensorium.sensors.SensorValue;
import java.io.PrintWriter;
import java.io.StringWriter;
import java.io.Writer;
import java.lang.reflect.Field;
import java.util.LinkedList;
import java.util.List;
import org.xmlrpc.android.IXMLRPCSerializer;

public class SensorRegistry {
    private static final int MAXDEBUGLINES = 20;
    public static final String TAG = "Sensorium";
    private static SensorRegistry instance = null;
    private int bufferedLines = 0;
    private Context context;
    private StringBuffer debugBuffer = new StringBuffer();
    private JSONLogger jsonLoggers;
    private Preferences preferences;
    private List<AbstractSensor> sensors = new LinkedList();
    private TextView textOutput;
    Thread x;
    XMLRPCSensorServerThread xmlrpcserverthread;

    59 protected SensorRegistry() {
    }

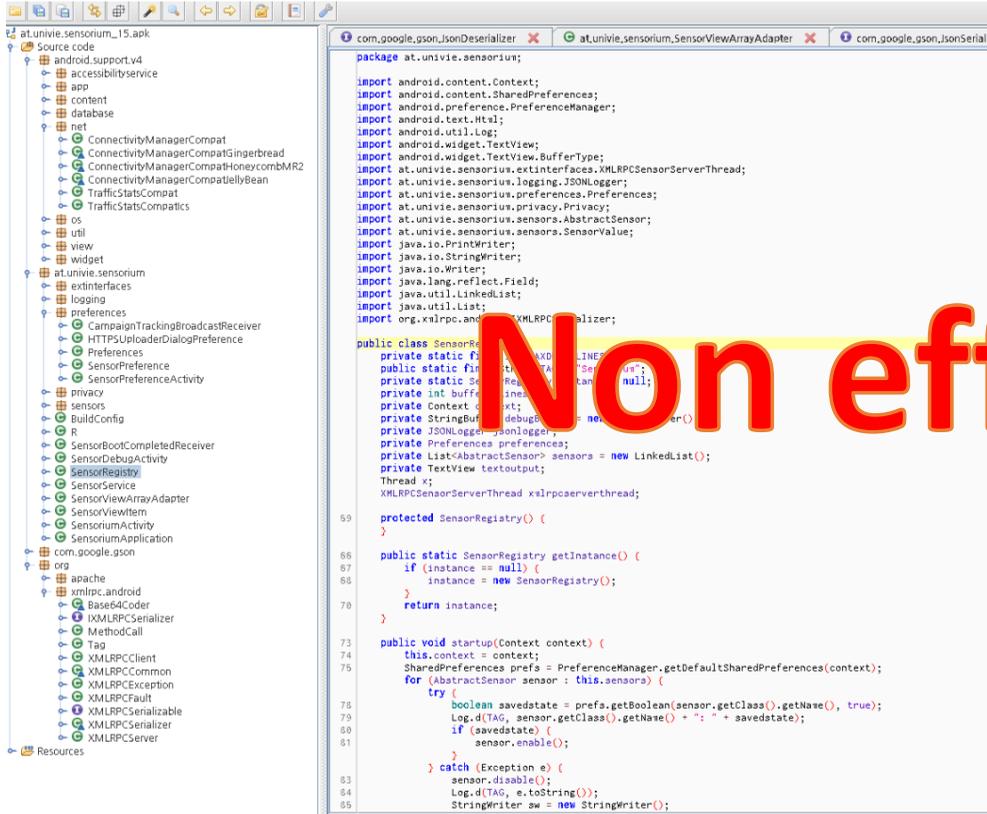
    66 public static SensorRegistry getInstance() {
    67     if (instance == null) {
    68         instance = new SensorRegistry();
    }
    70     return instance;
    }

    73 public void startup(Context context) {
    74     this.context = context;
    75     SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(context);
    for (AbstractSensor sensor : this.sensors) {
        try {
    78         boolean savedstate = prefs.getBoolean(sensor.getClass().getName(), true);
    79         Log.d(TAG, sensor.getClass().getName() + ": " + savedstate);
    80         if (savedstate) {
    81             sensor.enable();
        }
    } catch (Exception e) {
    83         sensor.disable();
    84         Log.d(TAG, e.toString());
    85         StringWriter sw = new StringWriter();
    }
```

How to know what to look for first?

*{Ctrl+F, Strings + Grep, Androguard, Smalisca, etc.} =
TEXTUAL, VERBOSE, SLOW,
TEDIOUS, BORING*

Manual inspection



How to know what to look for first?

Non effective !!

*{Crunchy Strings + Grep, Androguard, Smalisca, etc.} =
TEXTUAL, VERBOSE, SLOW,
TEDIOUS, BORING*

How many of you would like a tool...



Comprehensive

Easy to use

Visual

No black & white

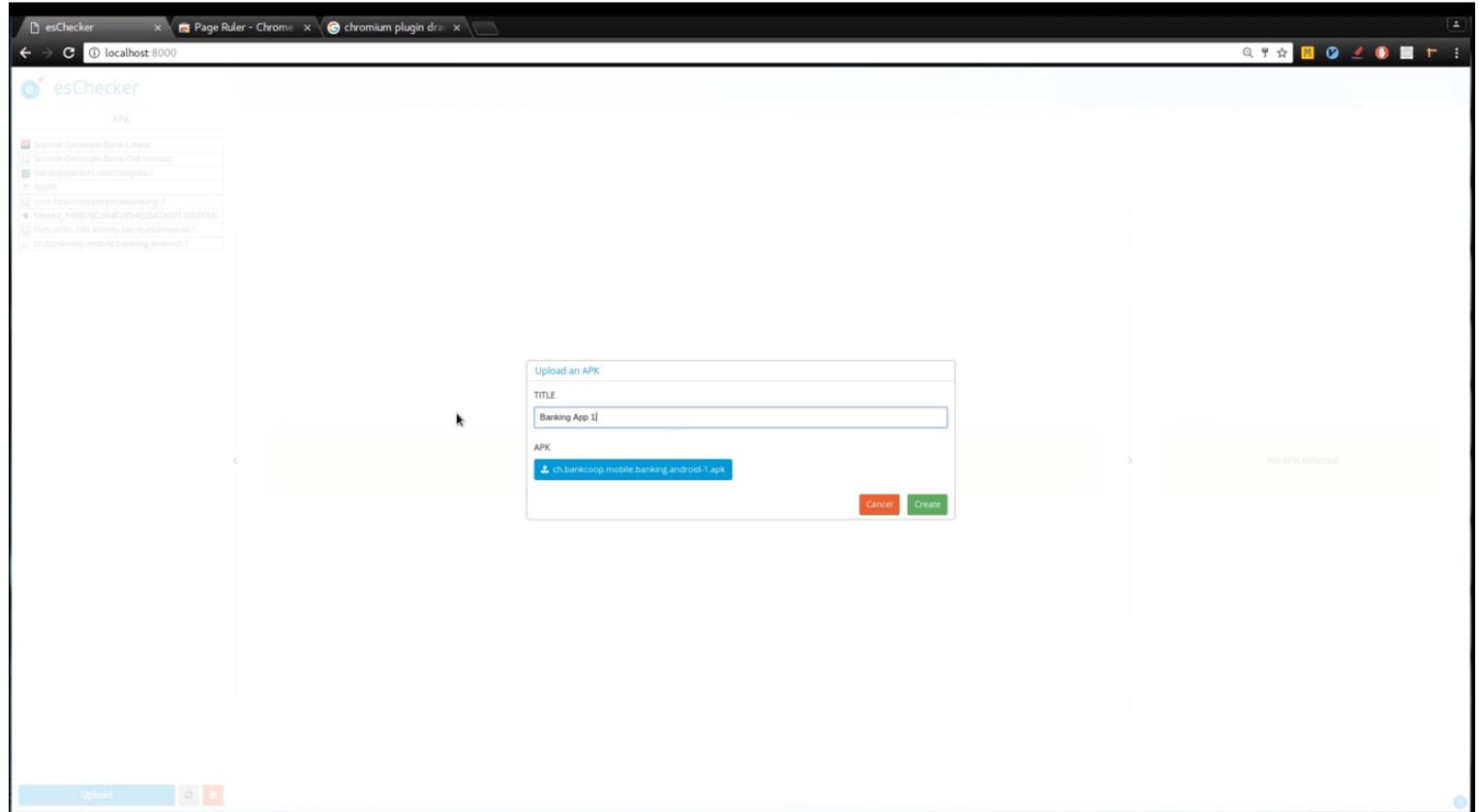
Customisable



Demo esChecker



- *Download your APK files*
- *The binary is disassembled and decompiled. Each piece of code is analysed against a set of heuristics*



Demo esChecker



- *As outcome of the analysis, a tile view is given.*
- *User has the opportunity to choose the APK*
- *User can browse into the file mapping view*
- *User can browse within the file tree*

The screenshot shows the esChecker web application interface. The main content area displays a 'File mapping' view for 'Banking App 1', which is a heatmap of red squares on a light background. The interface includes several panels:

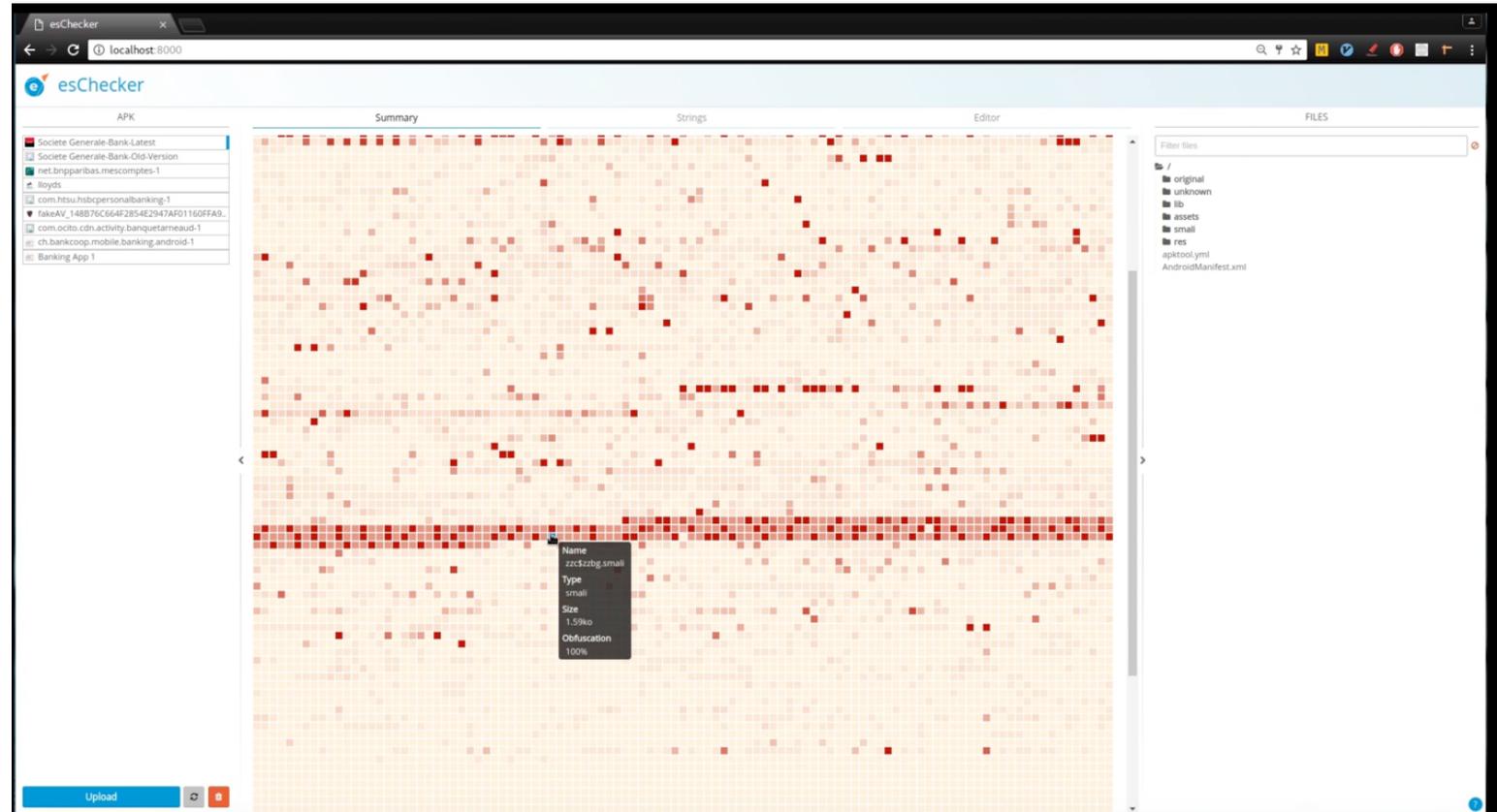
- Applications list:** A list of APKs on the left side, including 'Societe Generale-Bank-Latest', 'net.bnpparibas.mescomptes-1', and 'Banking App 1'.
- File tree:** A file tree on the right side showing the application's structure, including 'original', 'lib', 'assets', and 'AndroidManifest.xml'.
- Obfuscation summary:** A table showing the status of various obfuscation techniques.
- Compilers:** A section for selecting compilers, currently showing 'dexlib 1.x'.

Protection	Status
Class obfuscation	✓
Code obfuscation	✓
Symbol obfuscation	✓
Crypto API	✓

Demo esChecker



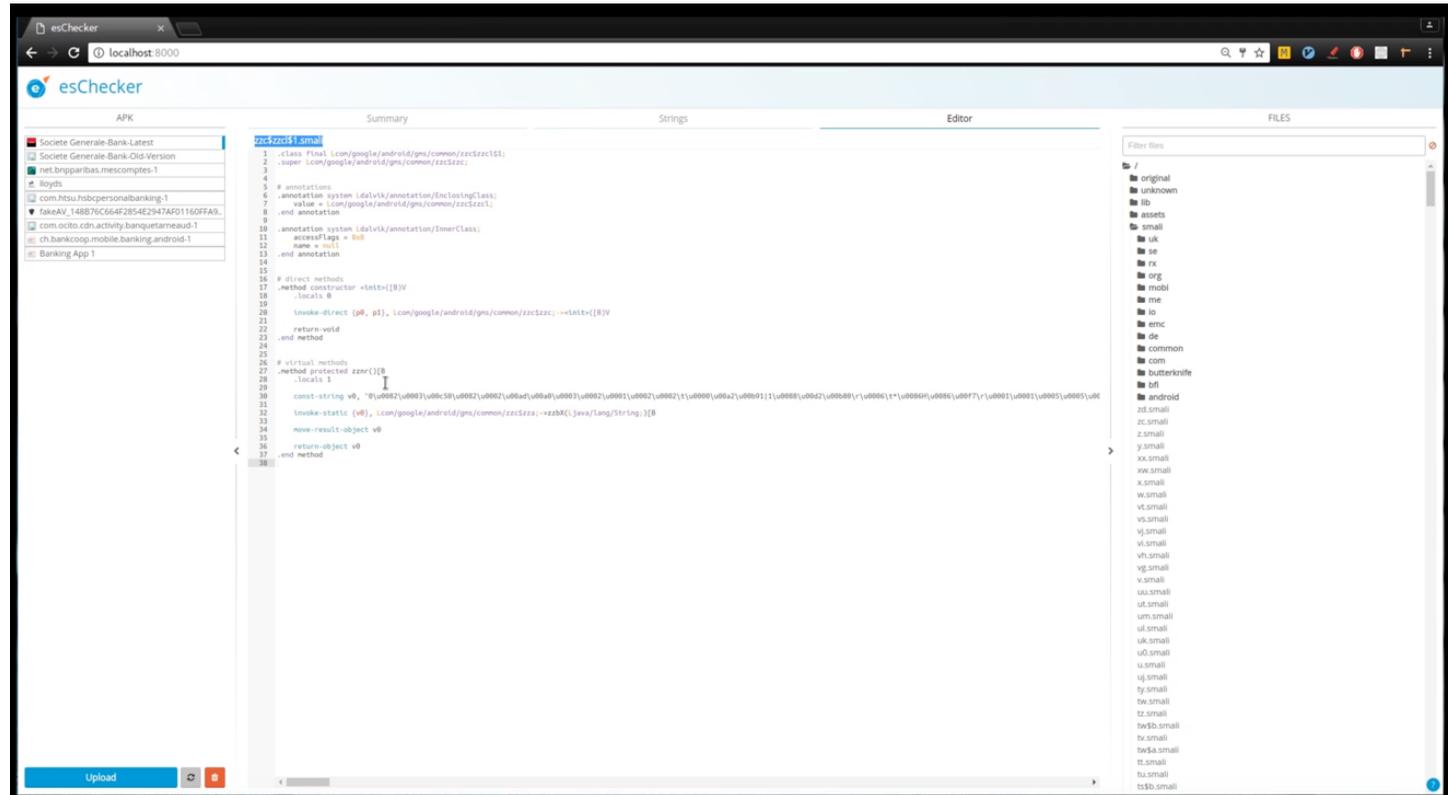
- *Each tile represents a piece of code*
- *An infobox displays information related to the file and shows how much obfuscation was found into it*
- *Different levels of colour helps to quickly see how much obfuscation was found*



Demo esChecker



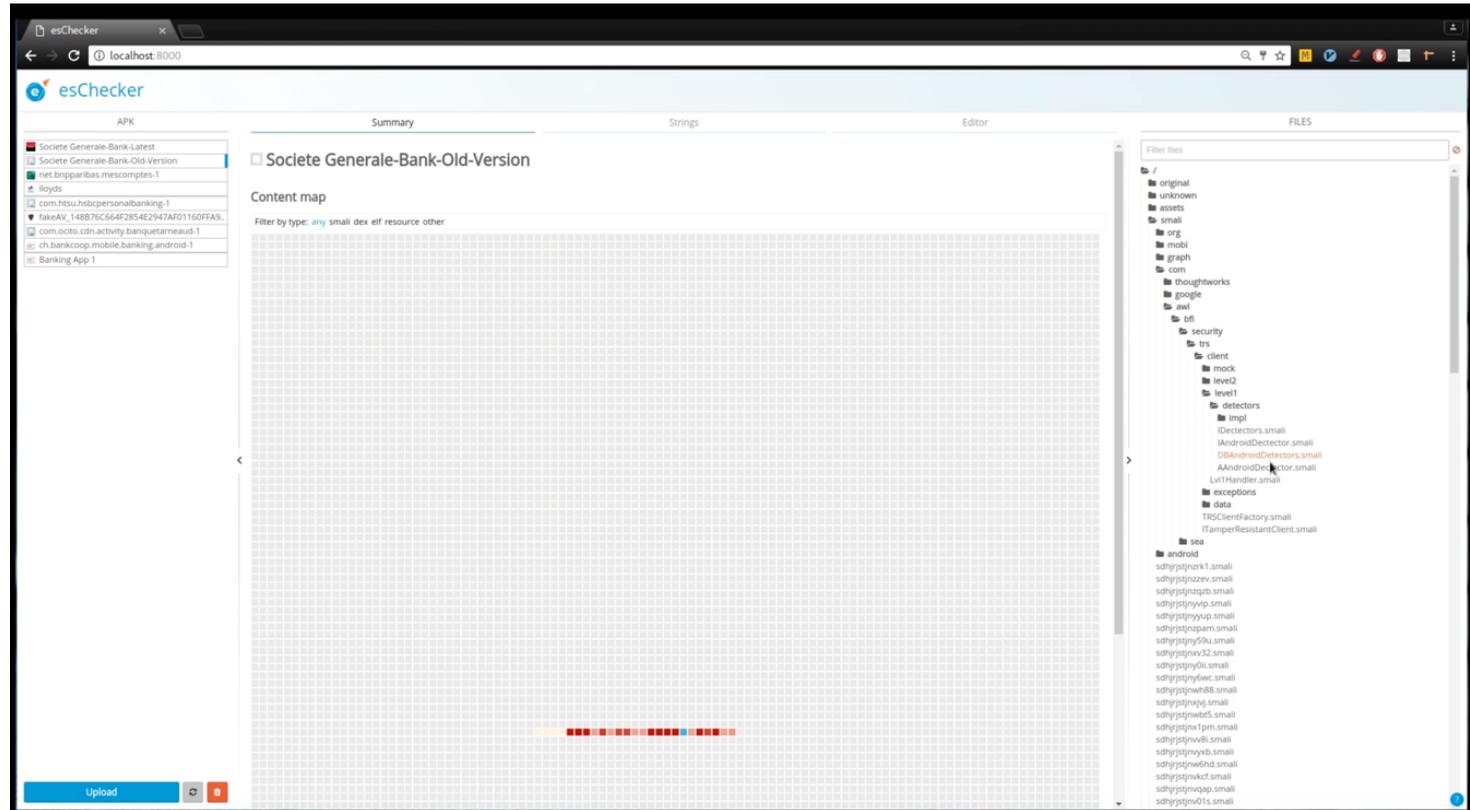
- *User can jump from the graphical view to the editor view or the other way around*
- *Corresponding obfuscation can be spotted... here, non ASCII characters and renaming*



Demo esChecker



- *The view can be restricted to a given package by simply selecting the folder on the right.*
- *User can just move up and down within the folder structures.*



Demo esChecker



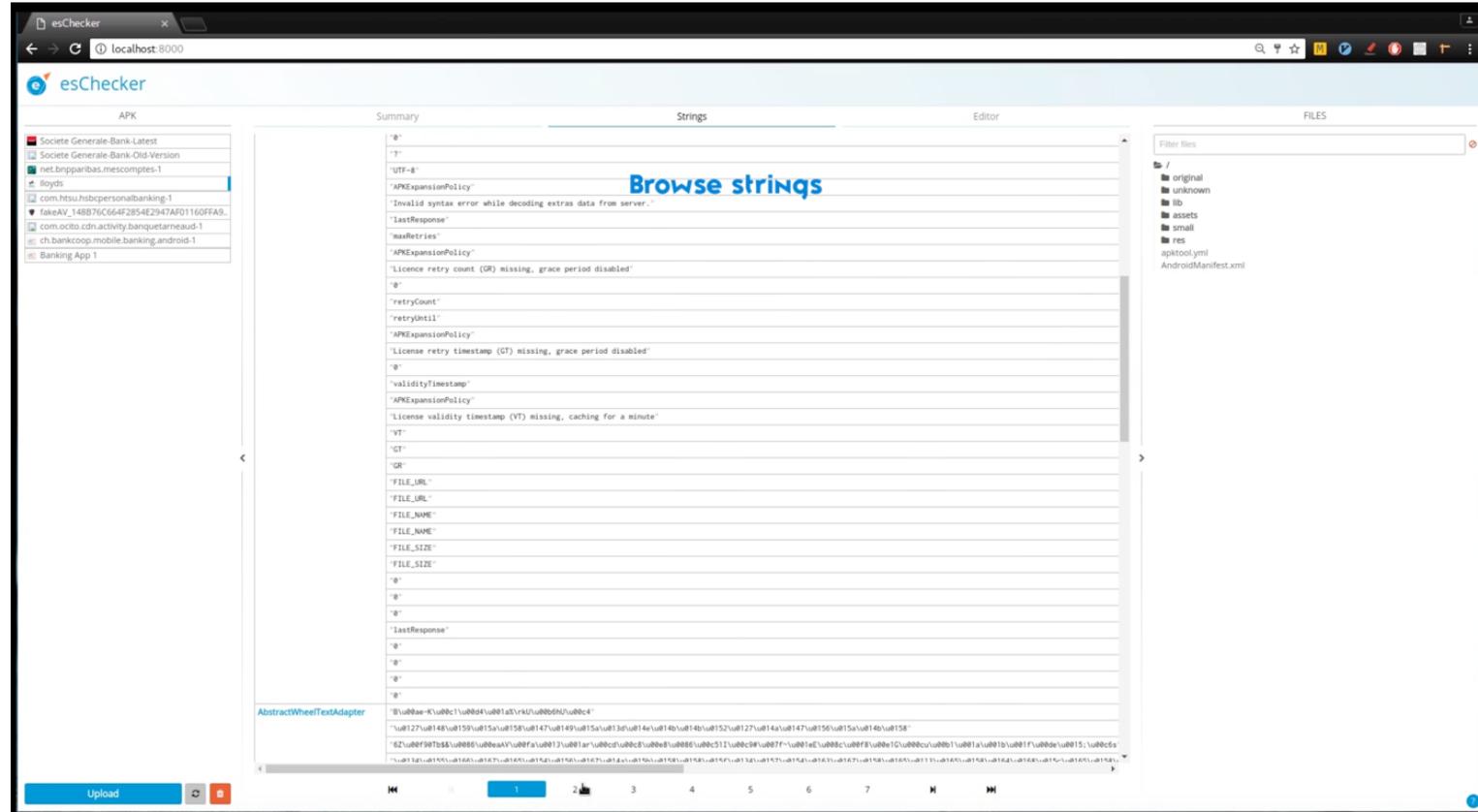
- *A search engine is available to collect all files with key words*
- *Helps to spot easily specific functions and locate them into the tile map.*

The screenshot displays the esChecker web application interface. The browser address bar shows 'localhost:8000'. The application title is 'esChecker'. The main content area is titled 'Societe Generale-Bank-Latest' and features a 'Search files' section. A 'Content map' is visible, showing a heatmap of search results. The search results are listed on the right side of the interface, showing various file paths and names, such as 'Payment\$PaymentDiscoverFragment\$1.small', 'Payment\$PaymentDiscoverFragment\$2.small', and 'Payment\$PaymentDiscoverFragment\$3.small'. The interface also includes a navigation menu with options like 'APK', 'Summary', 'Strings', 'Editor', and 'FILES'. A search bar at the top right contains the keyword 'Payment'. The bottom of the interface has an 'Upload' button and a refresh icon.

Demo esChecker



- *Strings are very useful to find his way into a binary*
- *A specific section of the tool is dedicated to strings*



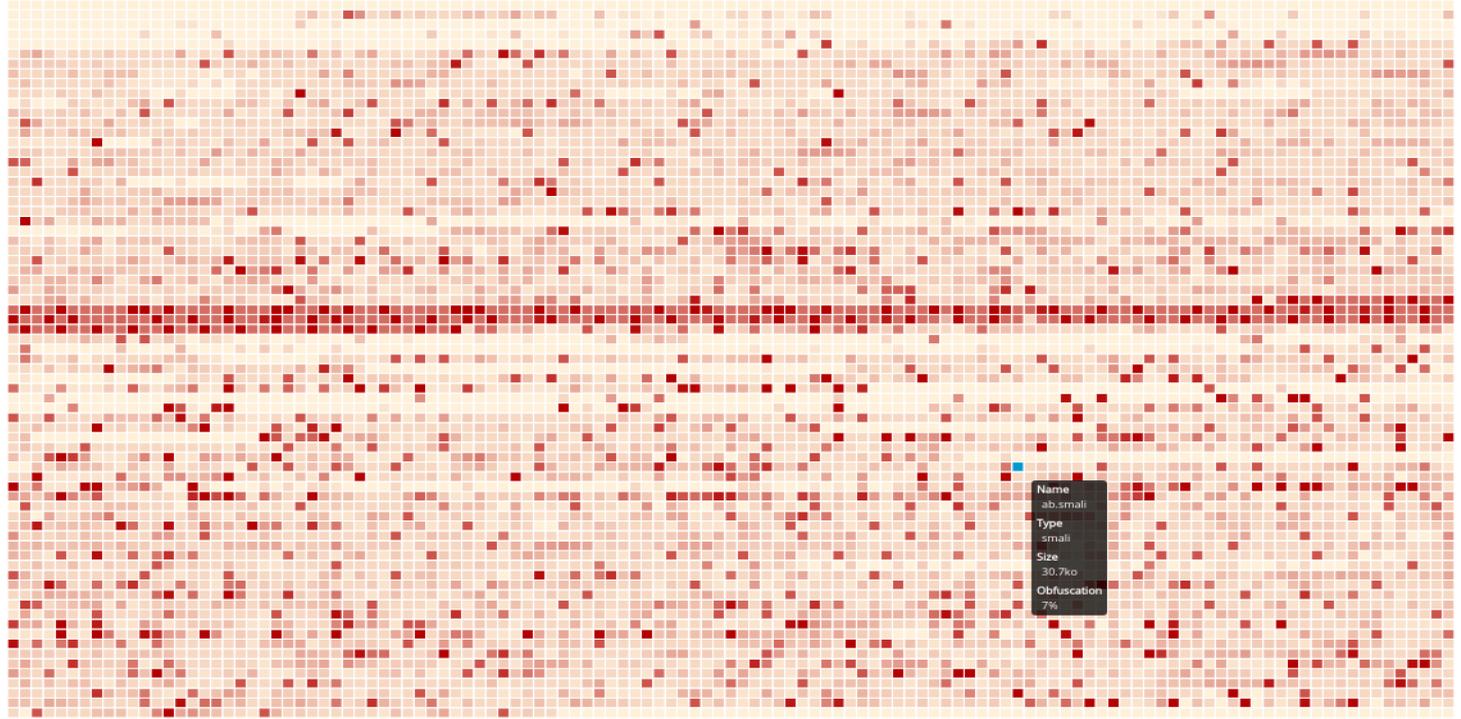
A first example



- *Weak obfuscation*
- *Strings are not obfuscated and remain way too much informative*
- *Control flow remains very easy to understand despite classes/methods/fields renaming*
- *GMS (in middle) looks more obfuscated than the rest of the app*

Content map

Filter by type: any small dex elf resource other

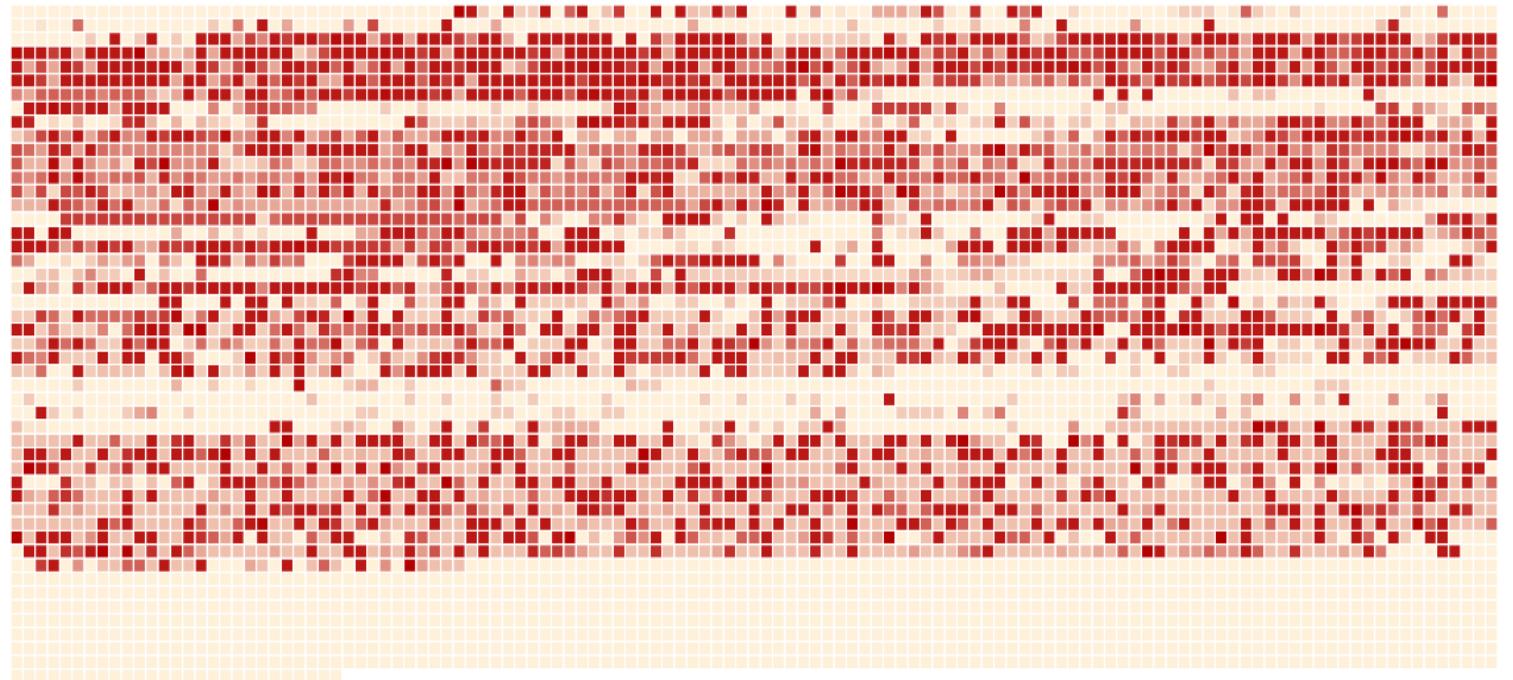


A second one...

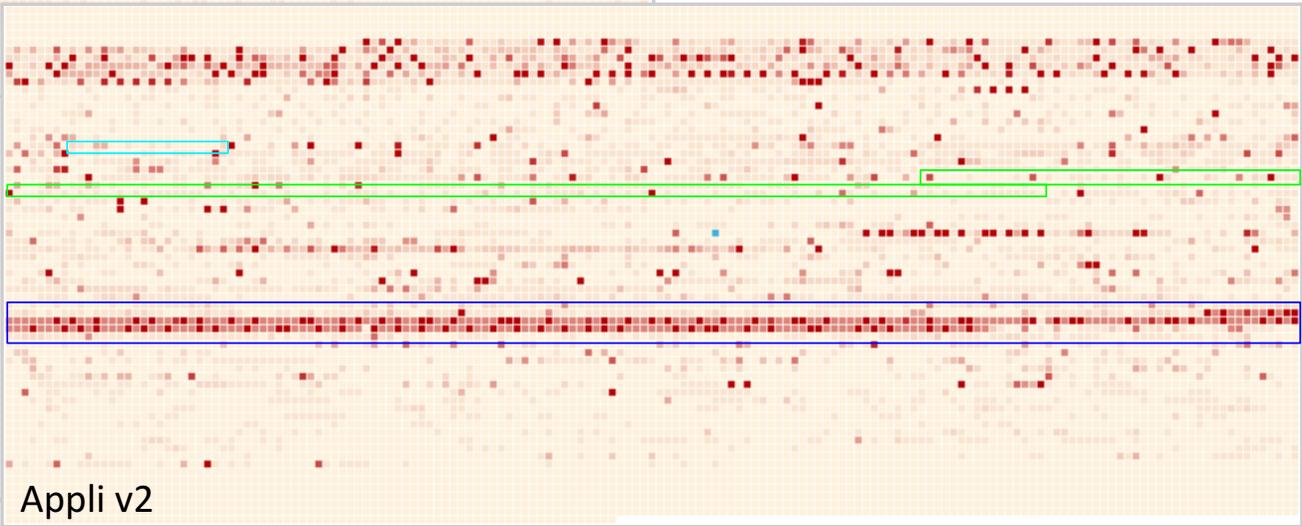
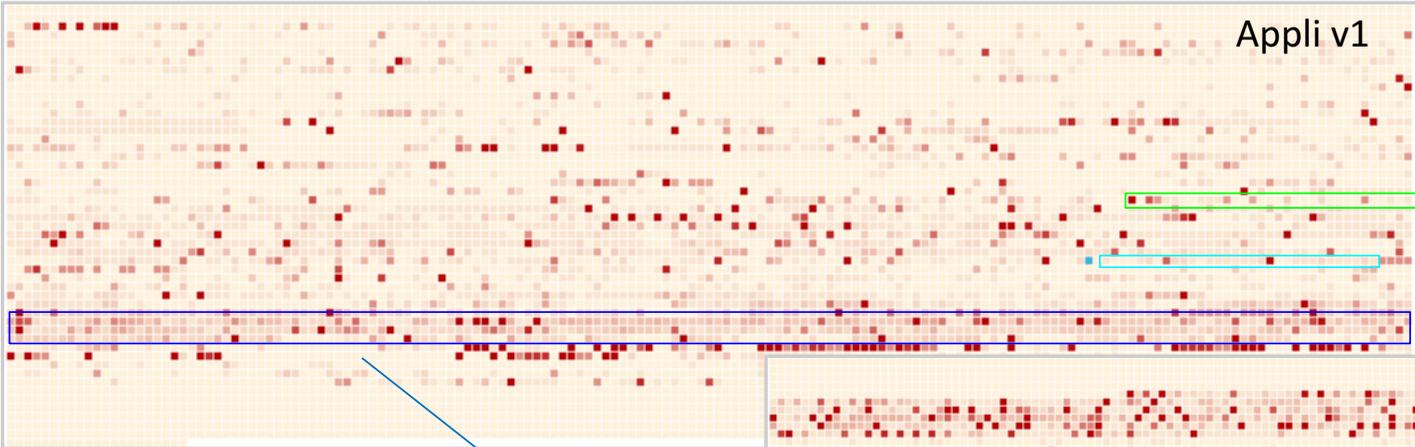
- *Medium obfuscation*
- *Classes/Methods/Fields are renamed*
- *Usage of non-ascii characters for the renaming*
- *Some strings are obfuscated/encoded*
- *Control flows have been slightly obfuscated*

Content map

Filter by type: [any](#) [smali](#) [dex](#) [elf](#) [resource](#) [other](#)



Delta version





**provides an easy
access to mobile app
protections**

Security Analysts

Save your time
Go straight to the point
Reach comprehensiveness

Everyone in the area

No need to be an expert to get a first insight
Manage the versioning and avoid regression
Use it as a benchmark
Reproducible process



Want to know more?

www.eshard.com



@eshardNews



eshard