apisecure

March 14&15, 2023

Gabrielle Botbol

Android Applications and APIs Hacking

apisecure Is powered by apidays & KNIGHT EVENTS

Who am I?

Gabrielle Botbol

@Gabrielle_BGB

/in/gabriellebotbol

https://csbygb.github.io/

CS/GB
Cyber Security by Gabrielle B.

From blogger to pentester

«Apprenance» is:

«a lasting set of dispositions… favourable to the act of learning… in all situations: formal or informal, experiential or didactic, self-directed or not, intentional or accidental».

Philippe Carré, 2005.

From blogger to pentester

Conferences

MOOC

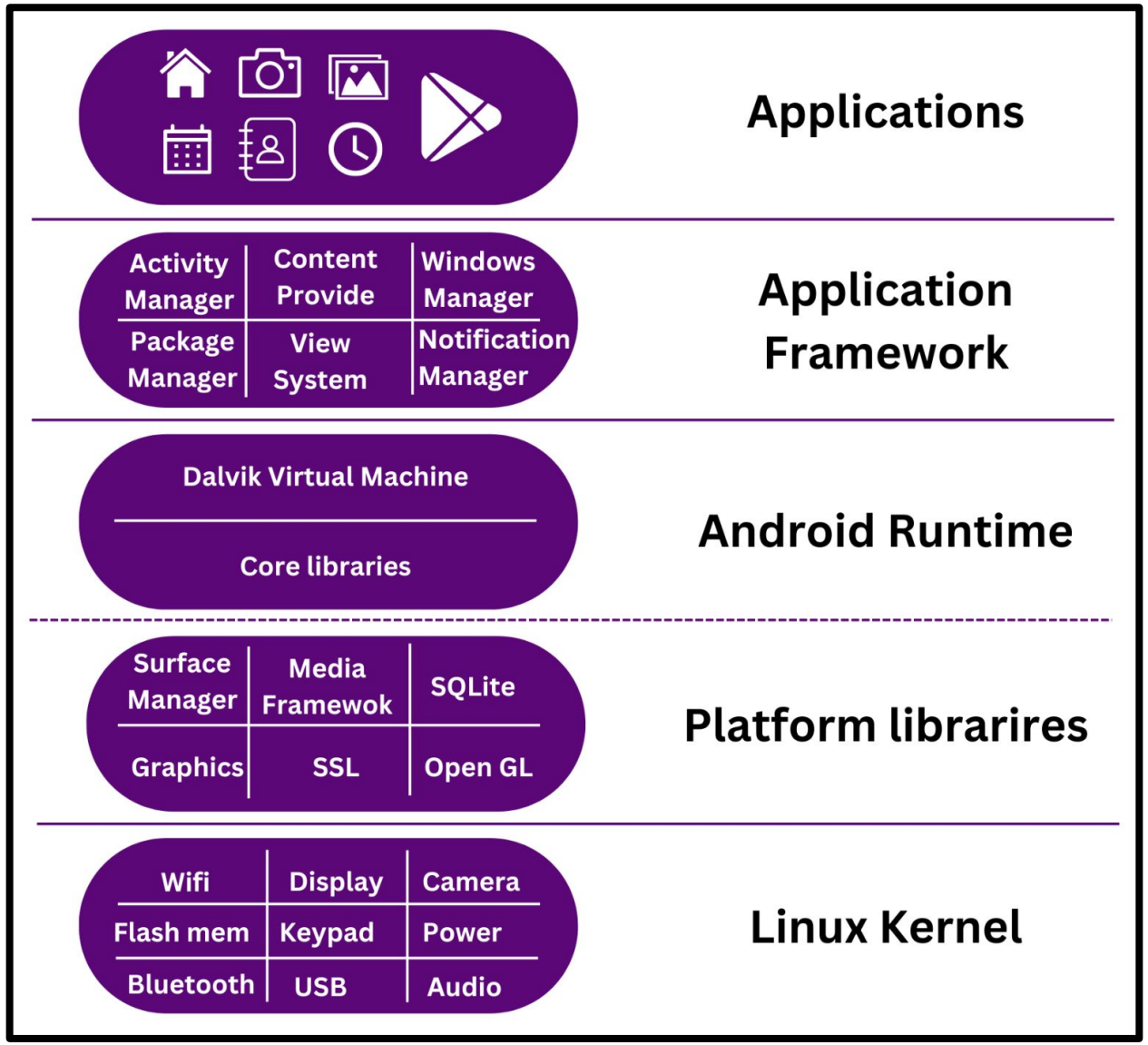
Volunteering

CTF

Summer Schools

Internship

What is Android



What is Android App Pentest?

Why Android App Pentest?

July 13th 2022

Maxime Ingrao
@IngraoMaxime

Found new family of malware that subscribe to premium services 👀

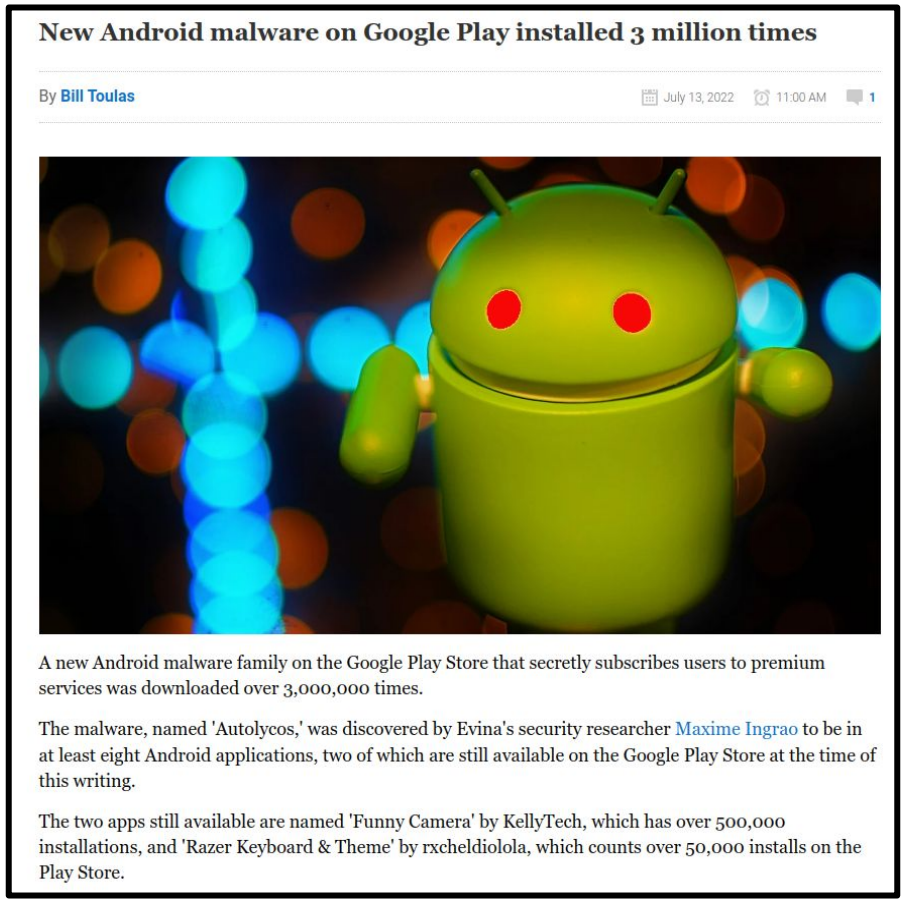
8 applications since June 2021, 2 apps always in Play Store, +3M installs 💀💀

No webview like #Joker 🤡 but only http requests

Let's call it #Autolycos 👾

#Android #Malware #Evina

Traduire le Tweet



New Android malware on Google Play installed 3 million times

By Bill Toulas — July 13, 2022 — 11:00 AM — 1

A new Android malware family on the Google Play Store that secretly subscribes users to premium services was downloaded over 3,000,000 times.

The malware, named 'Autolycos,' was discovered by Evina's security researcher Maxime Ingrao to be in at least eight Android applications, two of which are still available on the Google Play Store at the time of this writing.

The two apps still available are named 'Funny Camera' by KellyTech, which has over 500,000 installations, and 'Razer Keyboard & Theme' by rxcheldiolola, which counts over 50,000 installs on the Play Store.

Some figures

2,034,217+ New Mobile Malware Samples Detected in the Wild in 2021	**466%** Increase in Exploited, Zero-Day Mobile Vulnerabilities	**10M+** **Mobile Endpoints Impacted By Threats**
42% **Enterprises Reported Mobile Devices and Web Apps Led To A Security Incident**	**75%** Phishing Sites Specifically Targeted Mobile Devices	**23%** Of Mobile Devices Encountered Malicious Applications Worldwide

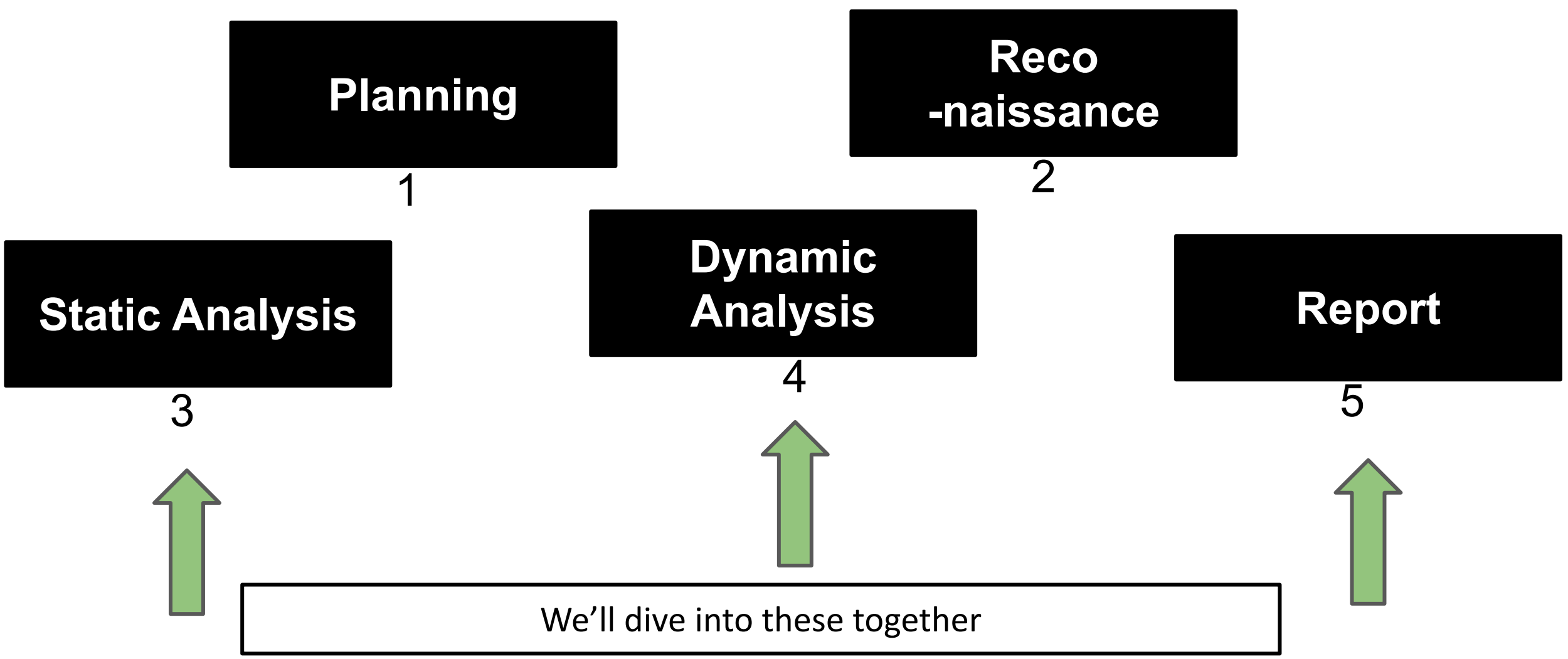
Source: https://www.zimperium.com/global-mobile-threat-report/

What about Android APIs?

Why dev use APIs?

- Manipulate data from remote locations
- Third party services
- Improve performance
- Code Reuse
- Flexible and scalable
- They can also make their own APIs

Android App pentest process



The importance of the lab

What you will need

Tools:

- Jadx
- apktool
- ADB
- Android Studio
- Burp Suite

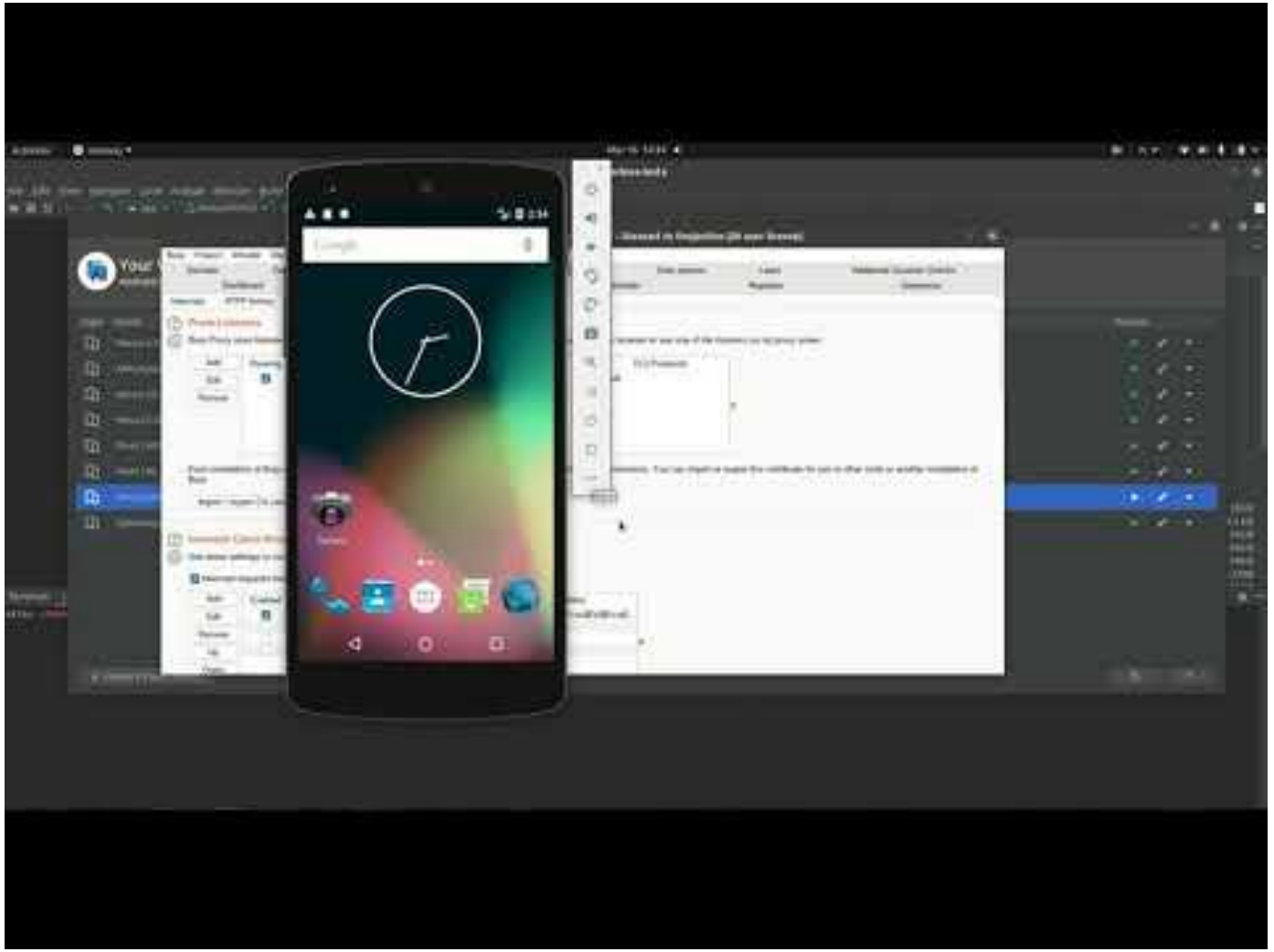
Set up the lab - Installs

Install Jadx	`sudo apt install default-jdk`
	`sudo apt install jadx`
	`./jadx-gui`

Install adb	`sudo apt-get install adb`
Install apktool	https://ibotpeaches.github.io/Apktool/install/
Install Android Studio	**Download** https://developer.android.com/studio
Install Burp Suite	**Download** and **install** the version according to your system here https://portswigger.net/burp/releases/professional-community-2021-12-1?requestededition=community
For more info on these installs	- JADX https://github.com/skylot/jadx - ADB https://www.xda-developers.com/install-adb-windows-macos-linux/

Set up the lab - Create an emulator

Set up the lab - Configure burp



How to Bypass certificate pinning: https://csbygb.gitbook.io/pentips/mobile-app-pentest/android#how-to-bypass-certificate-pinning

Practical examples of bypass of cert pinning: https://csbygb.gitbook.io/pentips/writeups/htbtracks/htb-intro-to-android-exploitation-track

=> Challenge: Pinned

=> Challenge: Anchored

Vuln Apps used for the examples

Get PIVAA here:
https://github.com/HTBridge/pivaa
Purposefully Insecure and Vulnerable Android Application.

Get InjuredAndroid here:
https://github.com/B3nac/InjuredAndroid/releases/tag/v1.0.12

Static Analysis

What to check:

- AndroidManifest.xml
- Strings.xml
- Enumerate Database
- Search for secrets and sensitive data

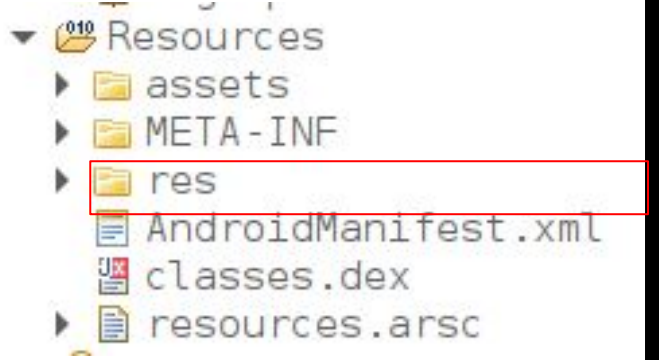
How to check the code

Jadx	`./jadx-gui`
apktool	`apktool d app.apk`

Decompiled files with apktool

```
AndroidManifest.xml  apktool.yml  original  res  smali
```

Example PIVAA - AndroidManifest 1




```
<uses-permission android:name="android.permission.GET_ACCOUNTS"/>
<uses-permission android:name="android.permission.READ_PROFILE"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.NFC"/>
<uses-permission android:name="android.permission.CALL_PHONE"/>
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
```

- List of permissions here: https://developer.android.com/reference/android/Manifest.permission
- List of permissions considered Dangerous: https://mas.owasp.org/MASTG/Android/0x05h-Testing-Platform-Interaction/#android-permissions

Example PIVAA - AndroidManifest 2

```
android:allowBackup="true"
```

(ON by default)

OWASP MSTG-STORAGE-8:

https://github.com/OWASP/owasp-mstg/blob/8d67a609ecd095d1bb00aa6a3e211791af5642e8/Document/0x05d-Testing-Data-Storage.md#static-analysis-7

```
android:debuggable="true"
```

(OFF by default)

OWASP MSTG-CODE-2:

https://github.com/OWASP/owasp-mstg/blob/53ebd2ccc428623df7eaf2361d44b2e7e31c05b9/Document/0x05i-Testing-Code-Quality-and-Build-Settings.md#testing-whether-the-app-is-debuggable-mstg-code-2

Static Analysis: Find the API endpoints

- Search for keywords “http”, “https”, etc.
- Look for function or classes (requests & responses)
- Manifest: permissions for network communications
- Check the JS files or AIDL files

Static Analysis: How are APIs called - Example

```
[STRIPPED]
public class ApiCallTask extends AsyncTask<String, Void, String> {
[STRIPPED]
        try {
            URL url = new URL(apiUrl);
            HttpURLConnection con = (HttpURLConnection) url.openConnection();
            con.setRequestMethod("GET");
            int responseCode = con.getResponseCode();
            Log.d(TAG, "API response code: " + responseCode);
            BufferedReader in = new BufferedReader(new
InputStreamReader(con.getInputStream()));
            String inputLine;
            StringBuffer responseBuffer = new StringBuffer();
            while ((inputLine = in.readLine()) != null) {
                responseBuffer.append(inputLine);
            }
            in.close();
            response = responseBuffer.toString();
        } catch (IOException e) {
            Log.e(TAG, "API call failed", e);
        }
        return response;
    }
[STRIPPED]
```

Class used and executed in an instance

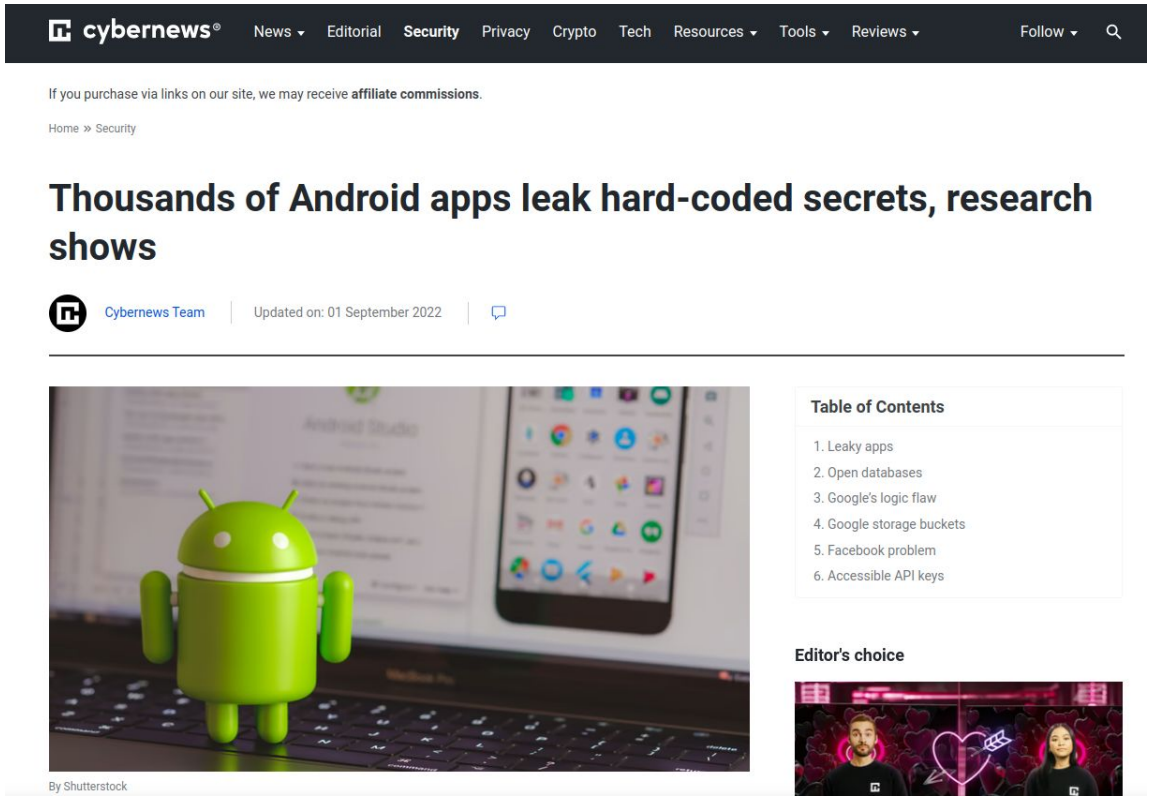
```
new
ApiCallTask().execute("http
s://api.example.com/data");
```

Static Analysis: Fetch API Javascript - Example

```
function fetchData() {
  var apiUrl = "https://api.example.com/data";
  var xhr = new XMLHttpRequest();
  xhr.open("GET", apiUrl, true);
  xhr.onreadystatechange = function() {
    if (xhr.readyState === 4 && xhr.status === 200) {
      var data = JSON.parse(xhr.responseText);
      displayData(data);
    }
  };
  xhr.send();
}
```

Static Analysis: API vulnerabilities

"This is a private key! WTF, man!" - Alissa Knight - 2019



Thousands of Android apps leak hard-coded secrets, research shows - Cybernews

2022

How I hacked 30 mobile banking apps & the future of API Security, Alissa Knight

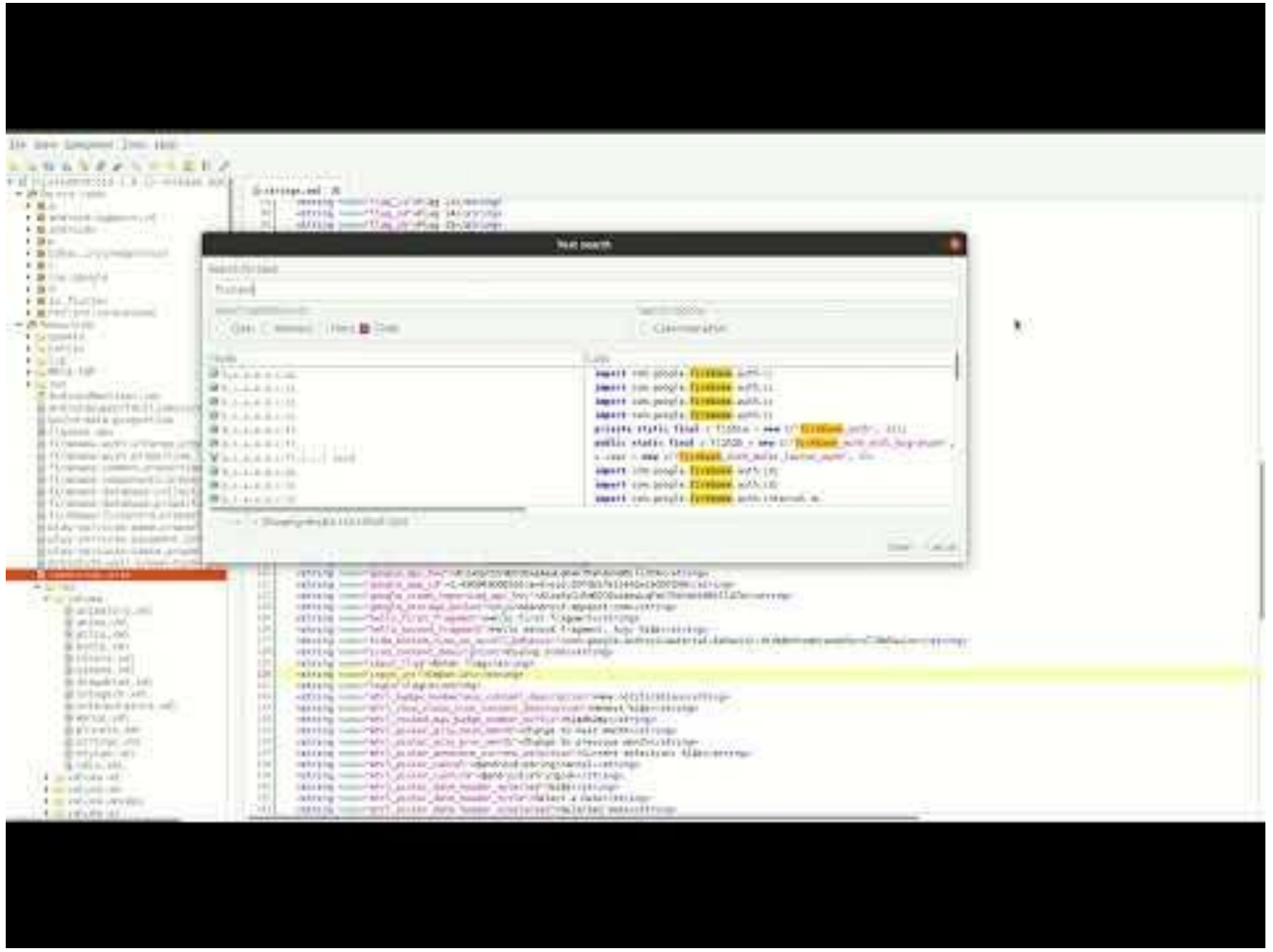
2019

Example with InjuredAndroid - Strings

/res/values/strings.xml

```
<string
name="google_api_key">AIzaSyCUImEIOSvqAswLqFak75xhskkB6illd7A</string>
<string
name="google_app_id">1:430943006316:android:d97db57e11e42a1a037249</str
ing>
<string
name="google_crash_reporting_api_key">AIzaSyCUImEIOSvqAswLqFak75xhskkB6
illd7A</string>
<string
name="google_storage_bucket">injuredandroid.appspot.com</string>
```

General Tips for static analysis



Grep it!


```
grep -r “unsafe secret”
```

APKTOOL

```
apktool d app.apk
```

MOBSF

/uploads directory

More tips on grep here:
https://csbygb.gitbook.io/pentips/digital-skills/useful-linux#grep

Tools for static analysis

- Firebase Enum Github:
  https://github.com/Sambal0x/firebaseEnum

- FireBaseScanner:
  https://github.com/shivsahni/FireBaseScanner

- Cloud Enum https://github.com/initstring/cloud_enum

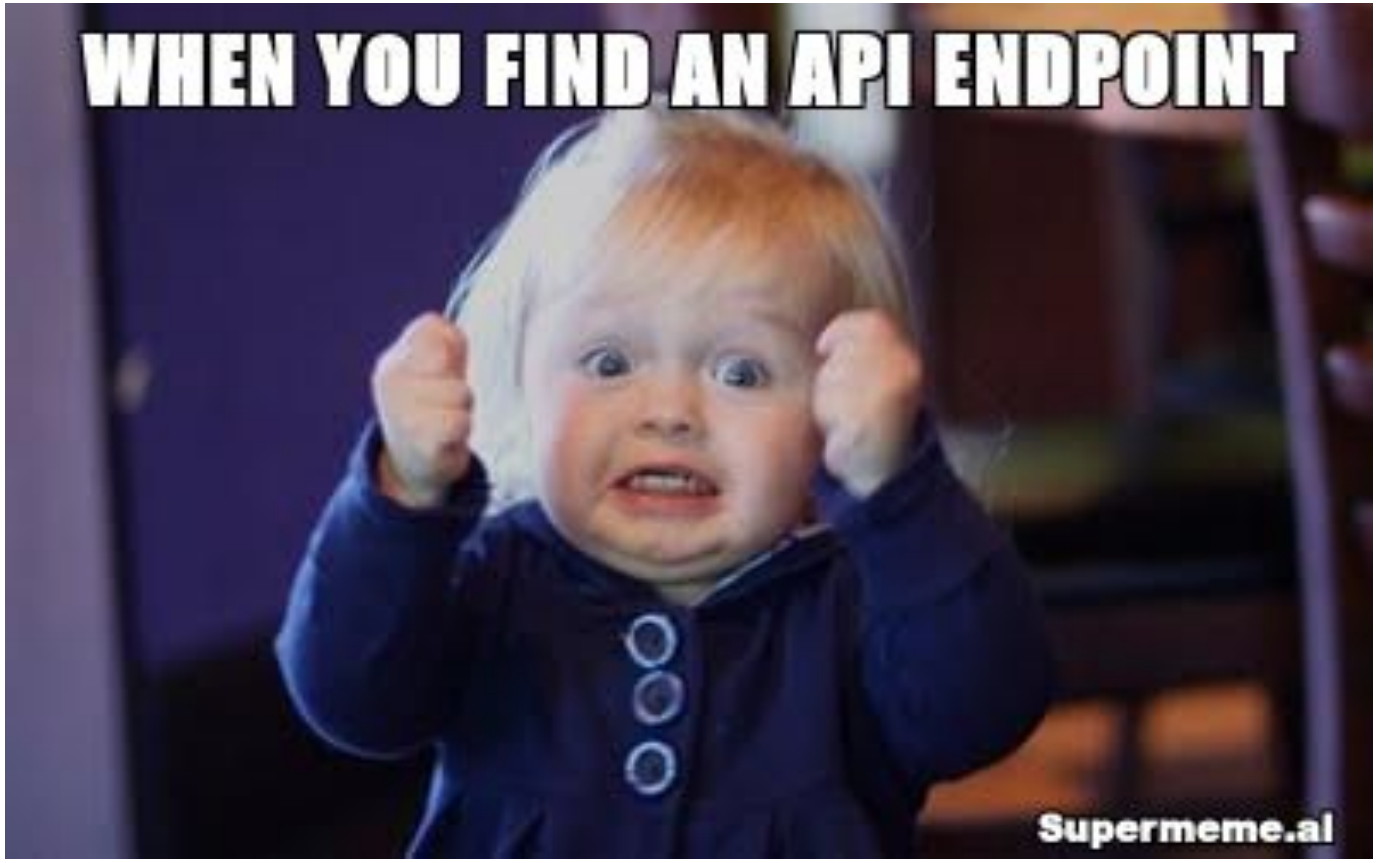
Dynamic Analysis

What to check:

- Tapjacking
- Can you capture screens with sensitive data
- OWASP Top 10
- Analyse traffic with burp to find odd things

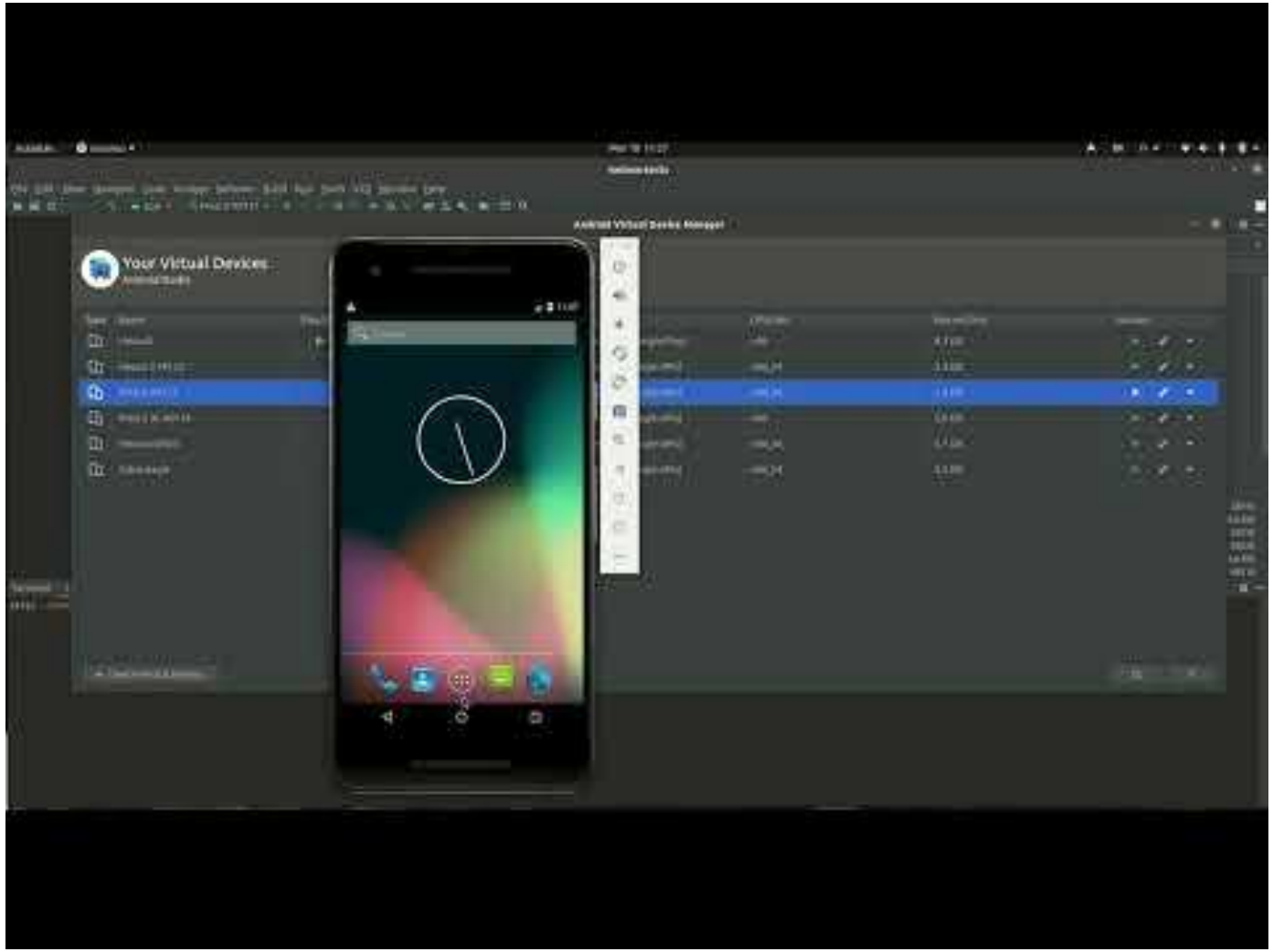
Dynamic Analysis: Find API endpoint

```
/api
/api/v1
/v1
/docs
/rest
/v1
/v2
/v3
/swagger
/swagger.json
/doc/graphql
```



Use a wordlist and FUZZ:
https://github.com/danielmiessler/SecLists/blob/master/Discovery/Web-Content/api/api-endpoints.txt

Example with PIVAA - BG capture



Automatic tools

- MobSF https://github.com/MobSF/Mobile-Security-Framework-MobSF

- Qark https://github.com/linkedin/qark

General tips: Common API vulnerabilities to look for

- API1:2019 Broken Object Level Authorization
- API3: 2019 Excessive Data Exposure
- API7:2019 Security Misconfiguration
- API9:2019 Improper Assets Management

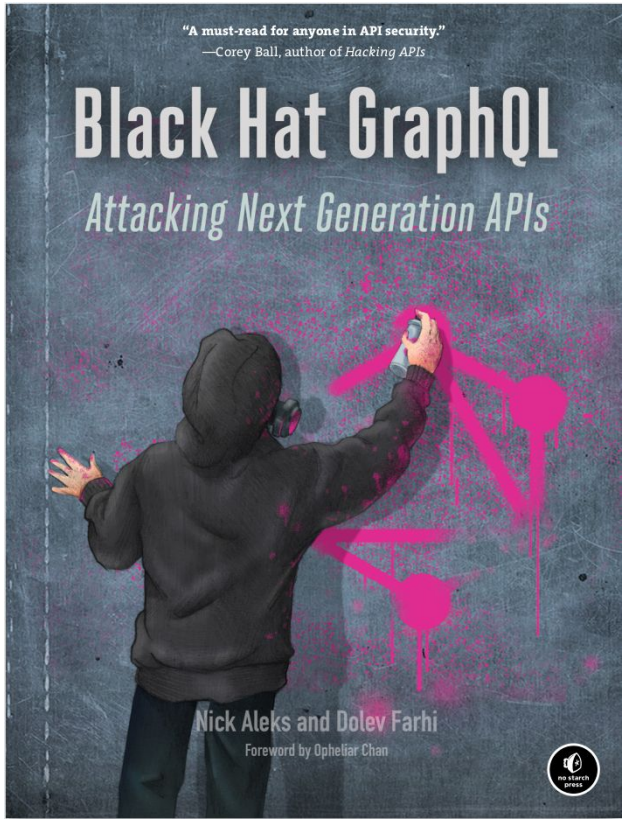
Find more here:
https://github.com/OWASP/API-Security/tree/master/2019/en/src

More tips on API pentest here: https://csbygb.gitbook.io/pentips/web-pentesting/api

General tips: Use checklists

Authentication and Authorization

- ☐ Test access to the following:
  - ☐ The API without authentication headers
  - ☐ Restricted fields by using alternate paths
  - ☐ The API by using both the GET and POST methods
- ☐ Test signature validation in JSON Web Token (JWT).
- ☐ Attempt to brute-force mutations or queries that accept secrets, such as tokens or passwords, using the following:
  - ☐ Alias-based query batching
  - ☐ Array-based query batching
  - ☐ CrackQL
  - ☐ Burp Suite



MindAPI - David Sopas: https://dsopas.github.io/MindAPI/play/

Official OWASP MAS Checklist: https://mas.owasp.org/MAS_checklist/

How to report

EXECUTIVE SUMMARY

VULNERABILITY REPORT

- Severity
- CVSS Score or OWASP Risk rating
- Affected item
- Description
- Remediation
- Evidence

How to report - Example

Broken Object Access Control

Severity: Medium

CVSS:3.1/AV:L/AC:L/PR:N/UI:R/S:U/C:L/I:N/A:N

Description

A BOLA (Broken Object Level Authorization) vulnerability is a security issue that allows an attacker to access or manipulate sensitive data or functionality in an application by modifying the object ID in the API requests. This vulnerability arises when the application lacks proper authorization checks and fails to enforce access control restrictions on user input.

In our context, we identified a BOLA vulnerability in the API of the application. This vulnerability could allow an attacker to bypass the access control measures and gain unauthorized access to sensitive data or functionality in the application.

How to report - Example

Broken Object Access Control

Remediation

We recommend that the development team implement proper authorization checks in the API to prevent this vulnerability from being exploited. Additionally, we suggest conducting a thorough review of the application's access control mechanisms to identify and address any other potential BOLA vulnerabilities.

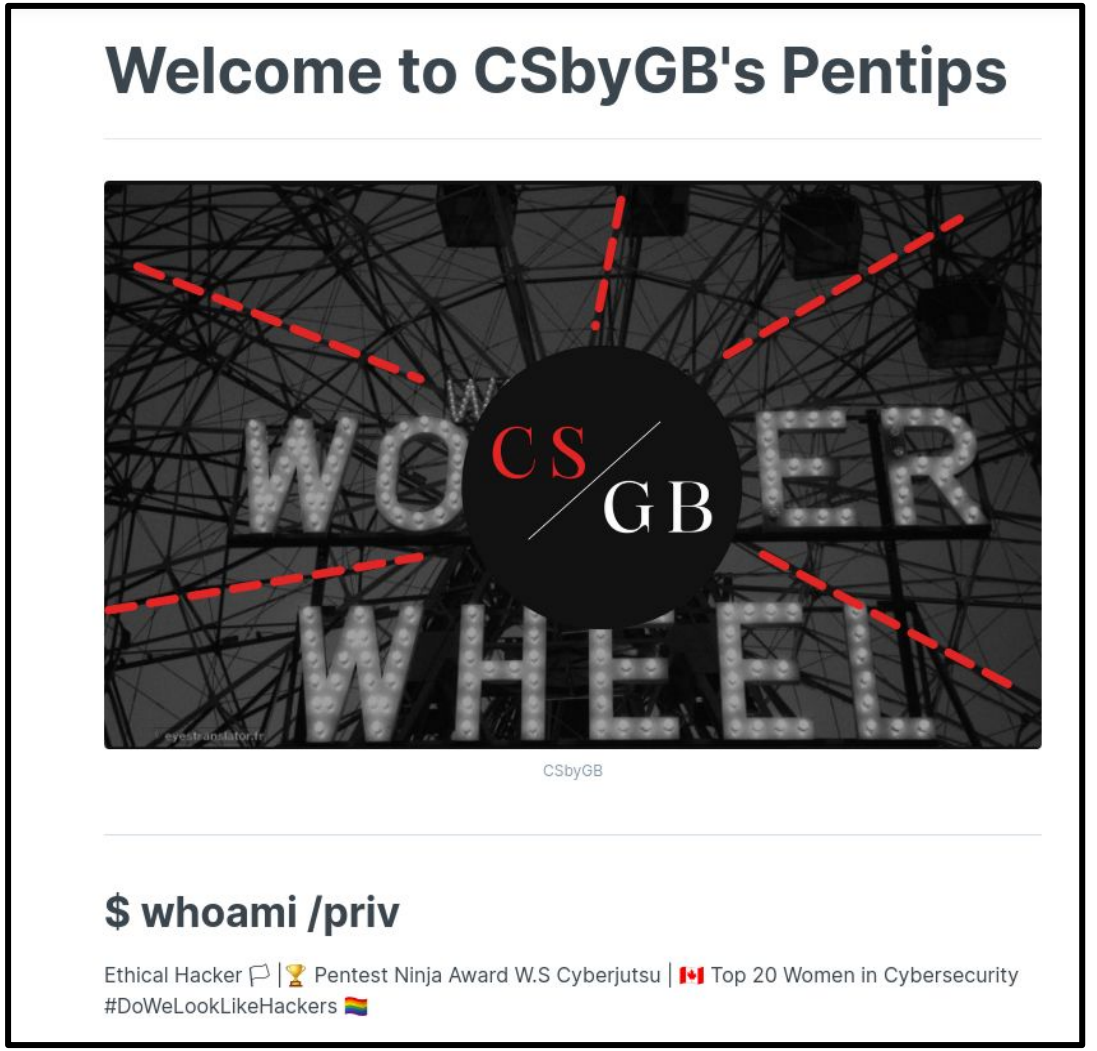
Resource

https://github.com/OWASP/API-Security/blob/master/2019/en/src/0xa1-broken-object-level-authorization.md

Get these slides and all the resources

https://csbygb.gitbook.io/

Android tips and BIG list of FREE resources:
https://csbygb.gitbook.io/pentips/mobile-app-pentest/android



Android Application Pentest Article - Pentest Magazine

- My article about Android Application Pentest
https://pentestmag.com/product/pentest-play-in-your-own-pentest-lab-in-2022/

Quiz to go

Check out the quiz about this presentation here:

https://forms.gle/GPymC3RrsmCRLxYC6

Special shout out

https://www.apisecuniversity.com/

Thanks