



Desenvolvedor, cuidado!
O seu aplicativo pode
estar vulnerável.

Misael Costa Júnior, M.Sc

Quem sou eu?



Misael Júnior

- Mestre em Ciência da Computação e Matemática Computacional no ICMC-USP
- Doutorando em Ciência da Computação e Matemática Computacional no ICMC-USP
- Analista de QA na TC - *Traders Club*;
- Tem experiência nos seguintes temas: Engenharia de Software, Teste de Software, Performance e Segurança de aplicativos móveis.





Importante!!!



- O conteúdo ministrado nessa oficina tem como objetivo apresentar os principais enganos cometidos pelo desenvolvedor que podem ser a causa de vulnerabilidades em aplicações Android;
- Desse modo, o objetivo não é ensinar como HACKEAR aplicativos Android, mas ajudar desenvolvedores a tornar os seus aplicativos mais seguros;
- Todo conteúdo ministrado na apresentação pode ser usado como uma base de conhecimento inicial em *pentest* e segurança de aplicativos Android.

PARTE 1



Contexto atual

01

- Mercado atual de SOs para aplicativos móveis
- Categorias de aplicativos mais usadas
- Por que vulnerabilidades ainda são frequentes?

Android

02

- Um pouco sobre o Android
- Arquitetura do Android
- Principais componentes de um aplicativo Android

Problemas e Riscos de segurança

03

- OWASP
- Top 10 Mobile
- Vamos conhecer algumas vulnerabilidades comuns em apps Android

PARTE 2



Ferramentas

01

- Ferramentas a serem utilizadas
- Alguns comandos importantes

Aplicações alvo

02

- DIVA

Identificando vulnerabilidades

03

- Um passo a passo
- DIVA
- Comunicação Insegura

Indicações

04

- Livros e Ferramentas
- Sites, Cursos e afins

PARTE 1

Vamos nos atualizar

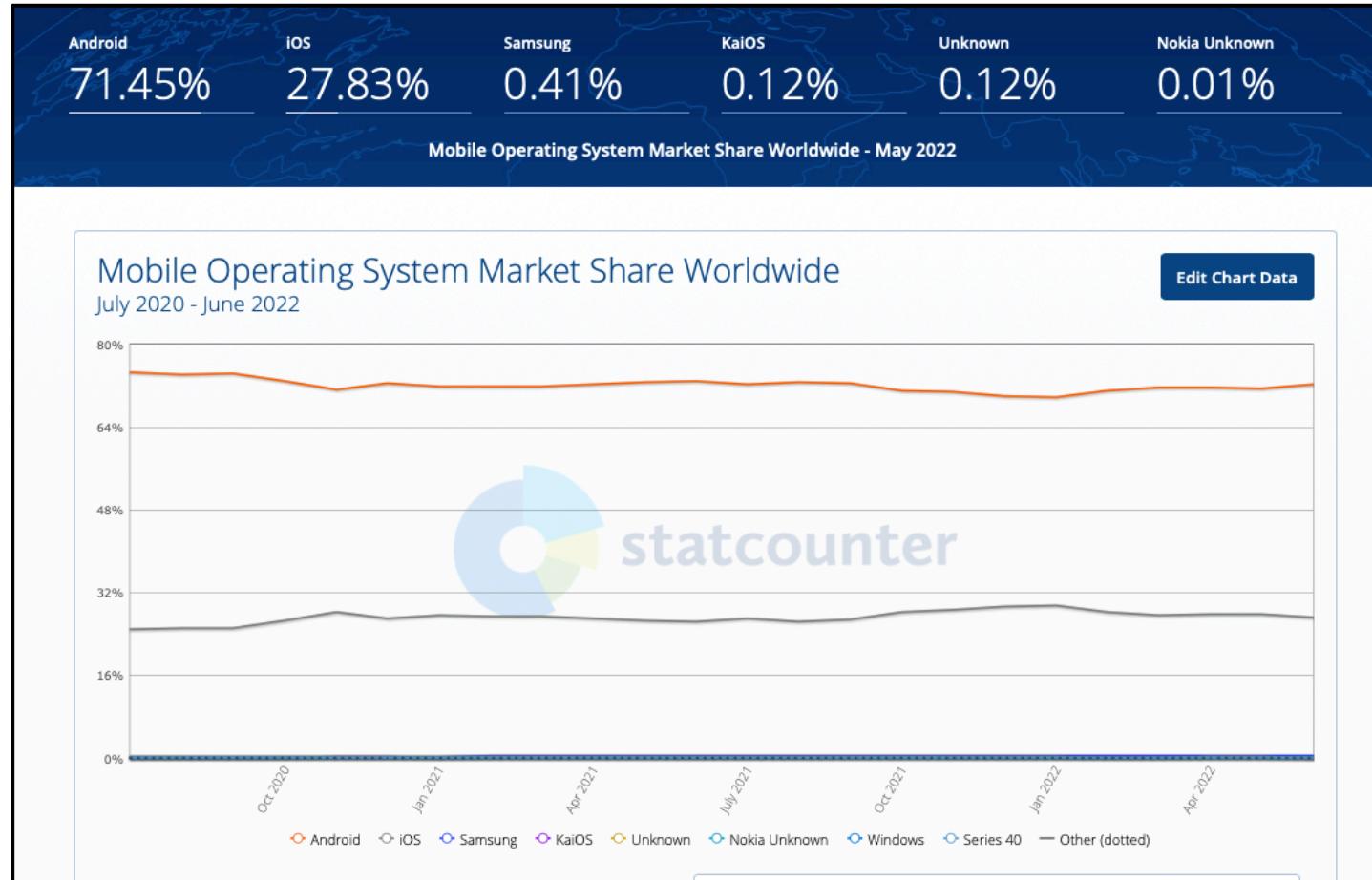




Contexto atual

Como está o contexto atual de aplicativos móveis?

Mercado de SOs móveis - Mundial



>70%

Dos dispositivos móveis usam o Android

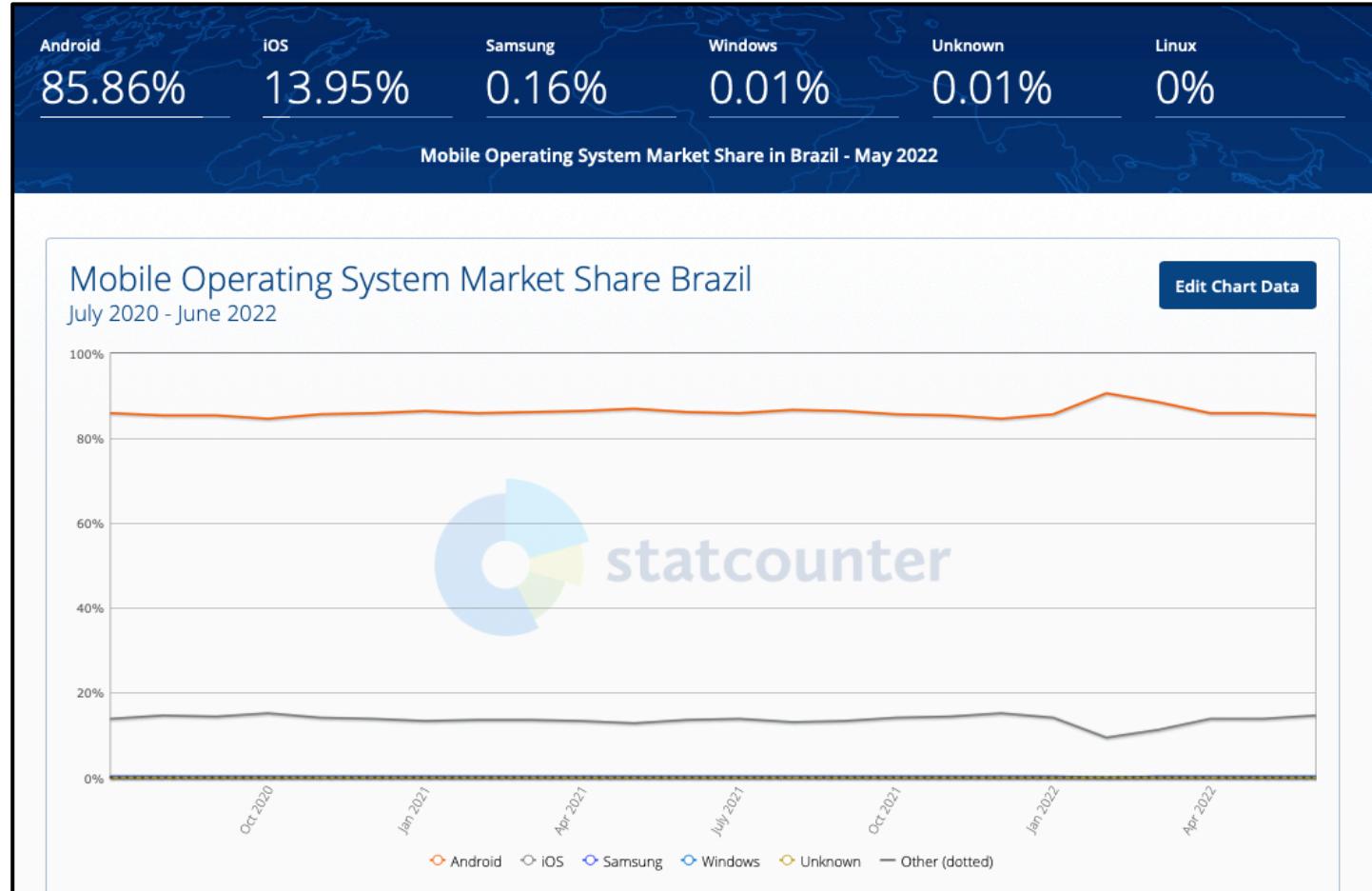
<30%

Dos dispositivos móveis atuais usam iOS

>90%

Dos dispositivos móveis usam Android ou iOS

Mercado de SOs móveis - Brasil



>80%

Dos dispositivos móveis usam o Android

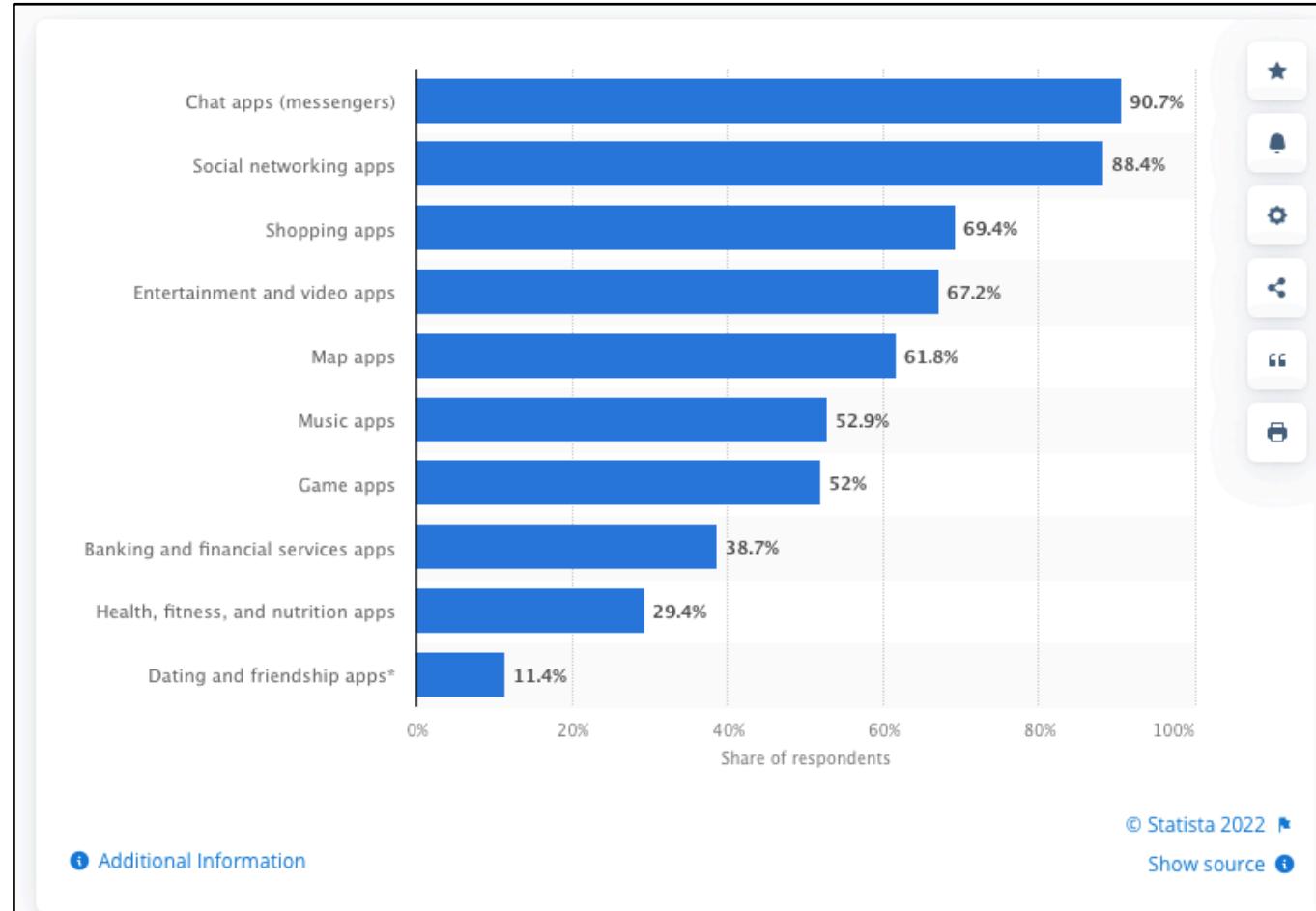
<15%

Dos dispositivos móveis atuais usam iOS

>95%

Dos dispositivos móveis usam Android ou iOS

Quais categorias de aplicativos são mais usadas?



- Ranking de Categorias de aplicativos mais usadas:
1. Apps de Chat (Whatsapp, Telegram, etc.);
 2. Redes Sociais (Instagram, Facebook, TikTok, Twitter, etc.);
 3. Apps de Compras (iFood)
 4. Entretenimento e videos (Youtube)
 5. Aplicativos de localização (Google Maps)

Aumento de vulnerabilidades

The Daily Swig
Cybersecurity news and views

Vulnerability in Facebook Android app nets \$10k bug bounty

Jessica Haworth 05 October 2020 at 13:56 UTC
Updated: 06 October 2020 at 11:25 UTC

Regions ▾ Hacking News ▾ Data Breaches ▾ Cyber-attacks ▾ Vulnerabilities ▾ Bug Bounties ▾ More ▾

Path traversal flaw leads to RCE

Latest Posts

HID Mercury access control vulnerabilities leave door open to lock manipulation

14 June 2022

Twitter, WhatsApp, Facebook, LinkedIn, Email icons

Microsoft | Microsoft Security Solutions ▾ Products ▾ Services Partners More ▾ Start free trial All Microsoft ▾

May 27, 2022 • 13 min read

Android apps with millions of downloads exposed to high-severity vulnerabilities

Microsoft 365 Defender Research Team

Popular mobile banking apps are riddled with security flaws, and Android users are more at risk

Brandon Vigliarolo in Security on June 18, 2020, 12:11 PM PDT

A study of banking apps for iOS and Android found poor source code protection, cleartext storage of sensitive data, and other serious flaws that make it easy for attackers to break into accounts.

wrike

Try more than just a calendar

Cyber Security News ▾ Hacking News ▾ News ▾ Vulnerabilities

Critical Vulnerability Discovered In The Uber App That Could Allow Account Takeovers

written by Abeerah Hashim | September 16, 2019

A hand holding a smartphone displaying the Uber app interface, showing the logo and some text, with a blurred background of a city street at night.

Home / Technology / Social / Android apps like Facebook, Instagram vulnerable even after updates: Check Point

Android apps like Facebook, Instagram vulnerable even after updates: Check Point

In a recent report, we found out that threat actors can execute code on the latest versions of various popular Android apps, even if the developers say they have patched the app.

By: Tech Desk | New Delhi | Updated: November 25, 2019 5:49:16 pm

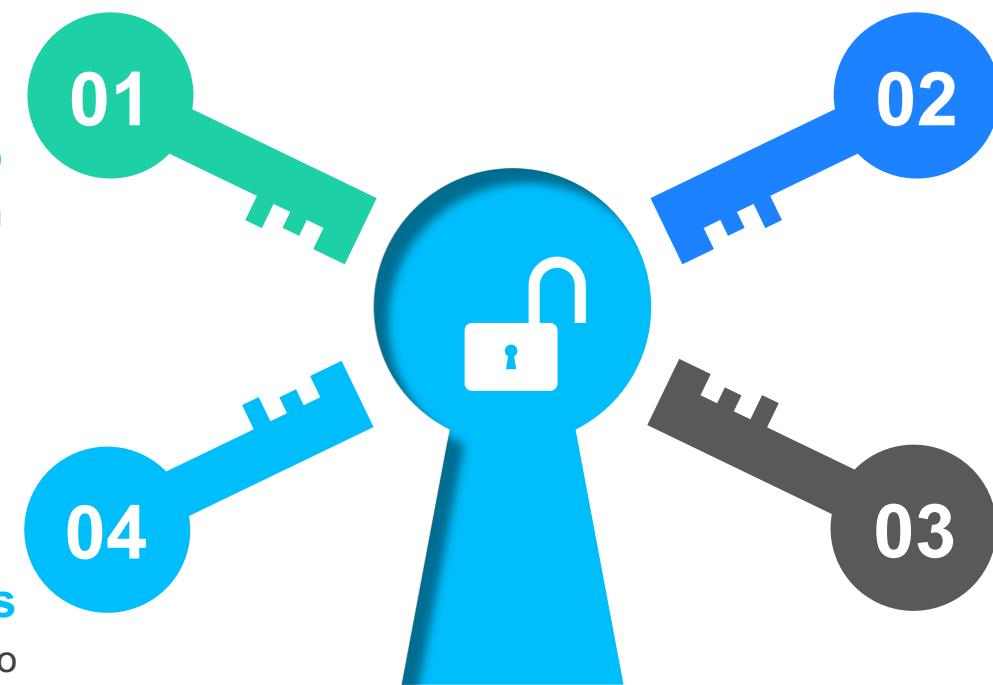
ADVERTISEMENT

Facebook, Twitter, Reddit, LinkedIn icons

Por que vulnerabilidades ainda são constantes?

Falta de conhecimento

Desenvolvedores não têm conhecimento sobre segurança.



Vulnerabilidades conhecidas

Os aplicativos móveis não são testados contra vulnerabilidades conhecidas.

Segurança é complexo

O teste de segurança não é uma tarefa simples e, frequentemente, necessita de pesquisa e investigação.

Segurança é negligenciada

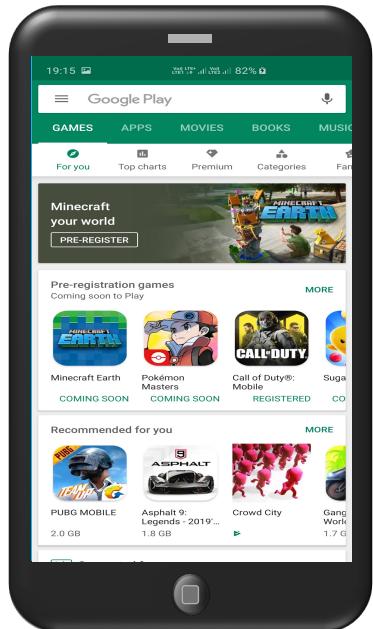
O teste de segurança é negligenciado com relação a outros tipos de testes



Android

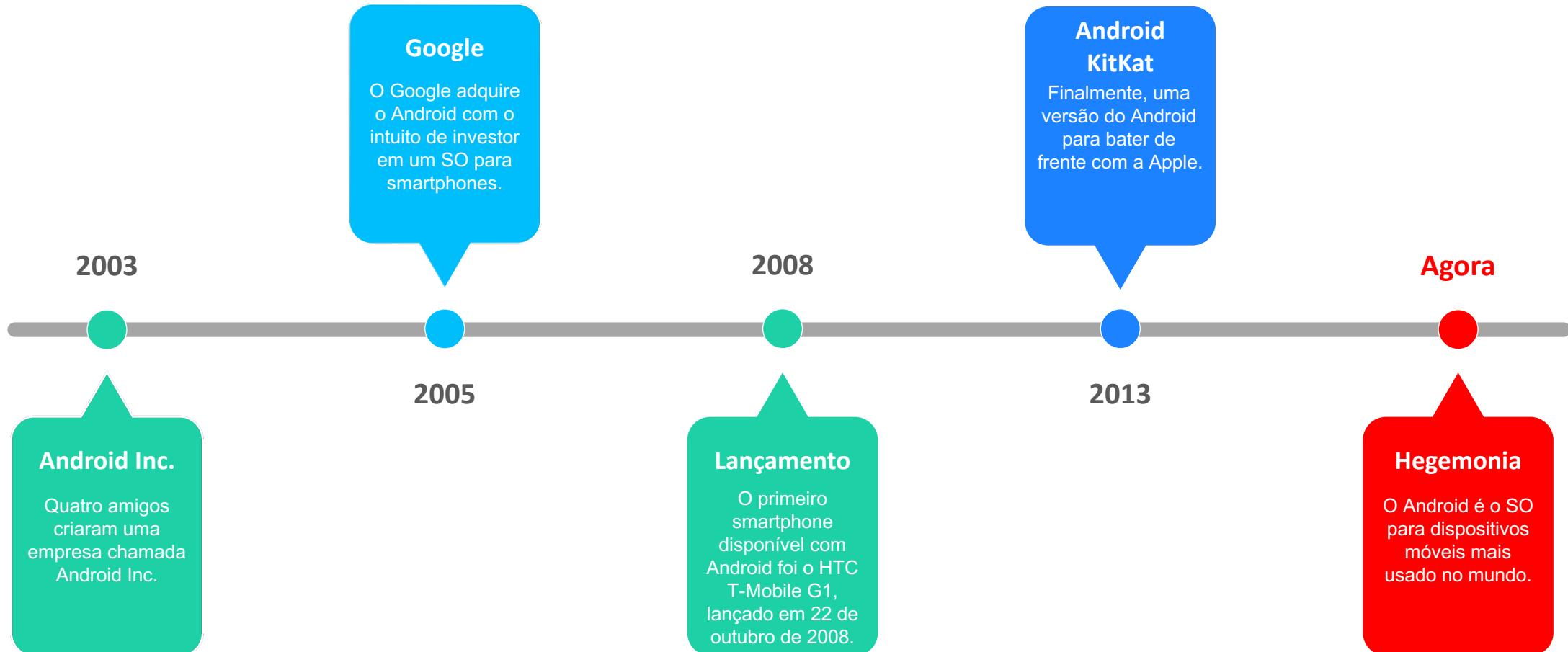
Conhecendo o Android, arquitetura e afins

Conhecendo o Android

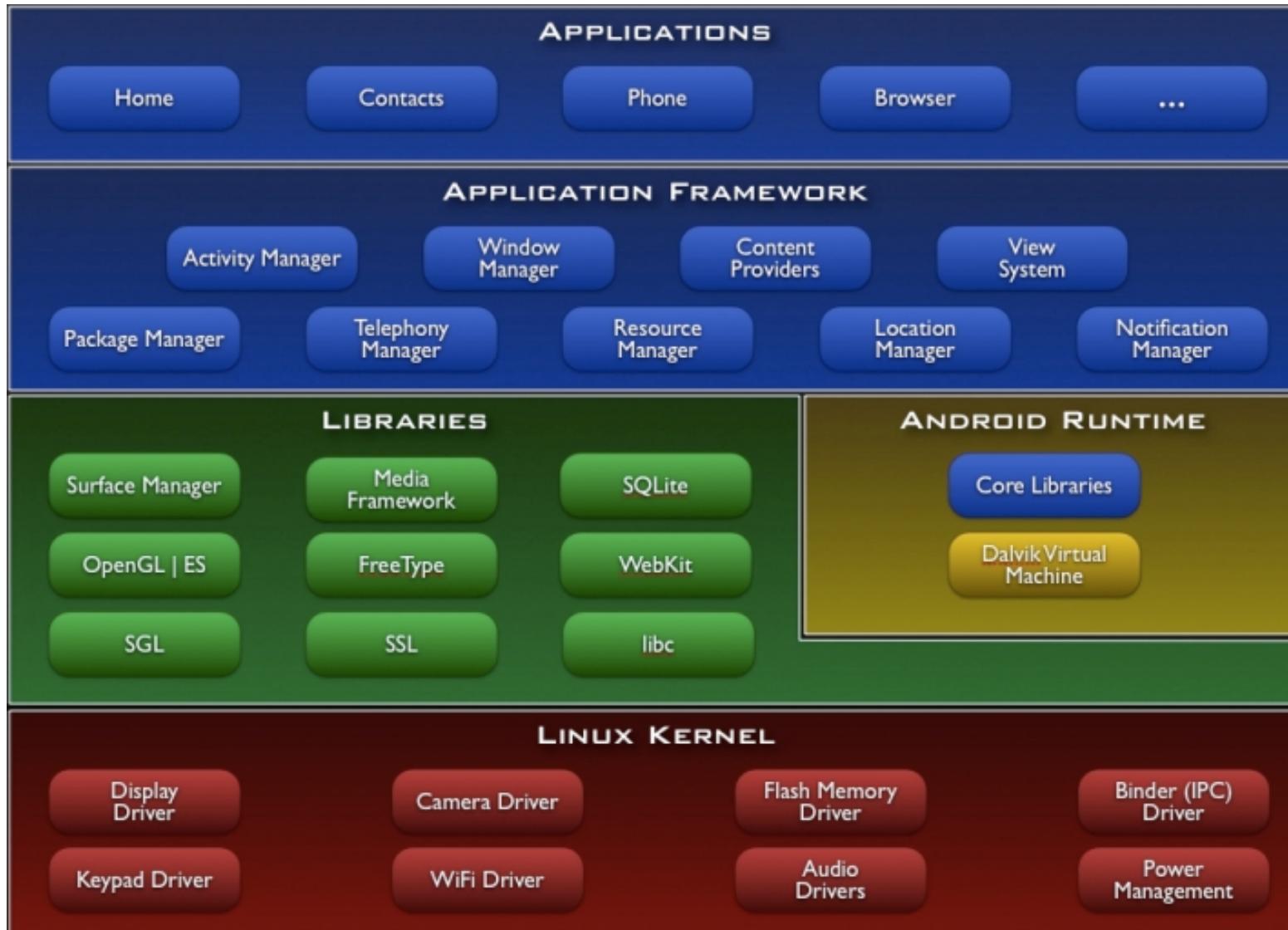


- Sistema Operacional mais utilizado em dispositivos móveis, tais como *smartphones* e *tablets*;
- Originalmente desenvolvido pela Android Inc em 2003 e adquirido pelo Google em 2005;
- Sistema Operacional aberto – *open-source* (<https://source.android.com>);
- Usa o Kernel Linux como base.

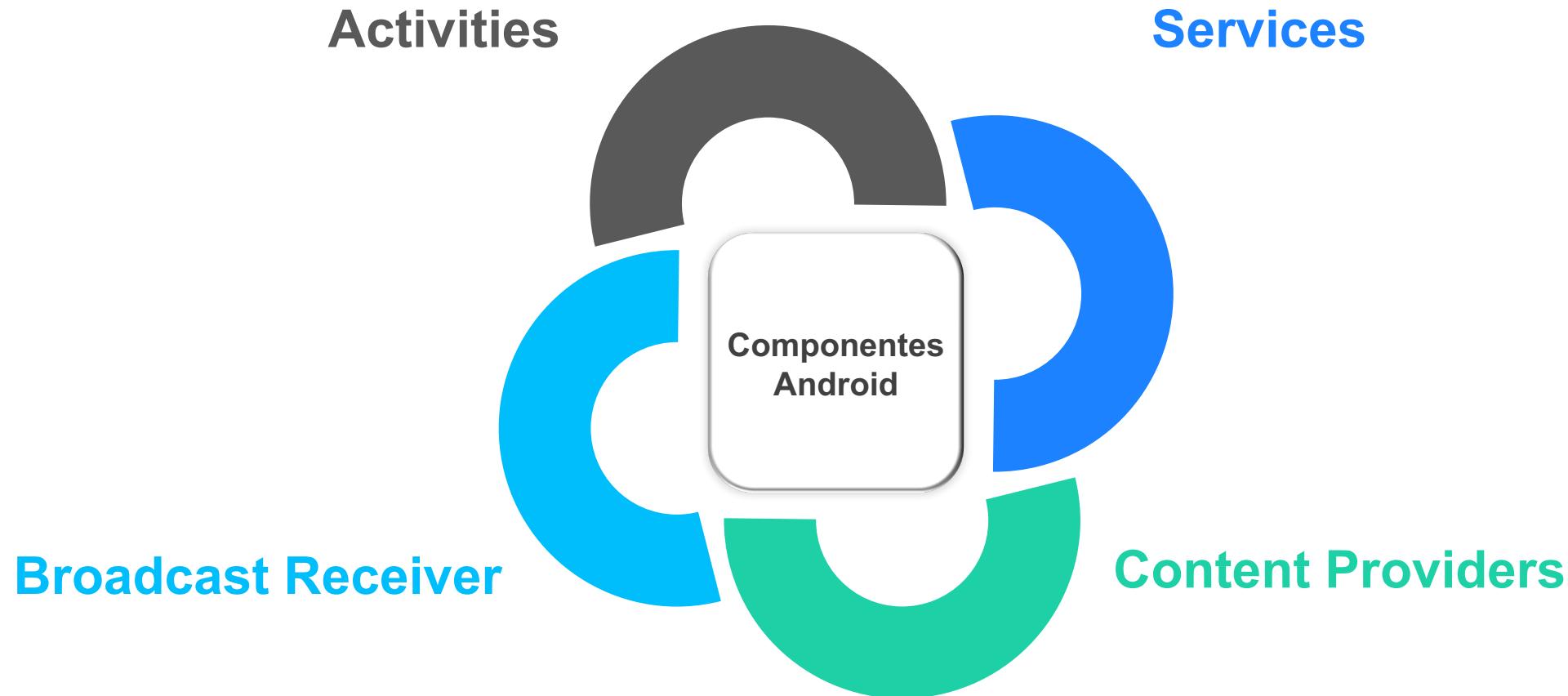
A História do Android



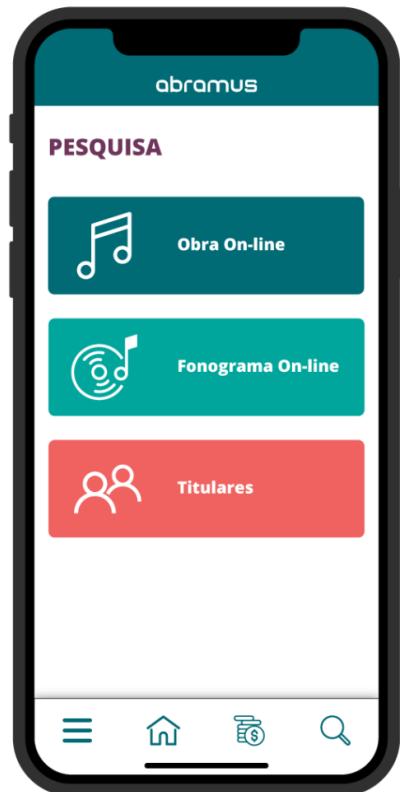
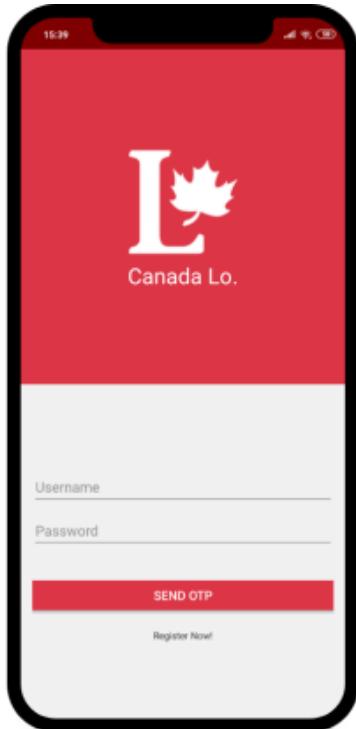
Arquitetura do Android



Principais componentes do Android



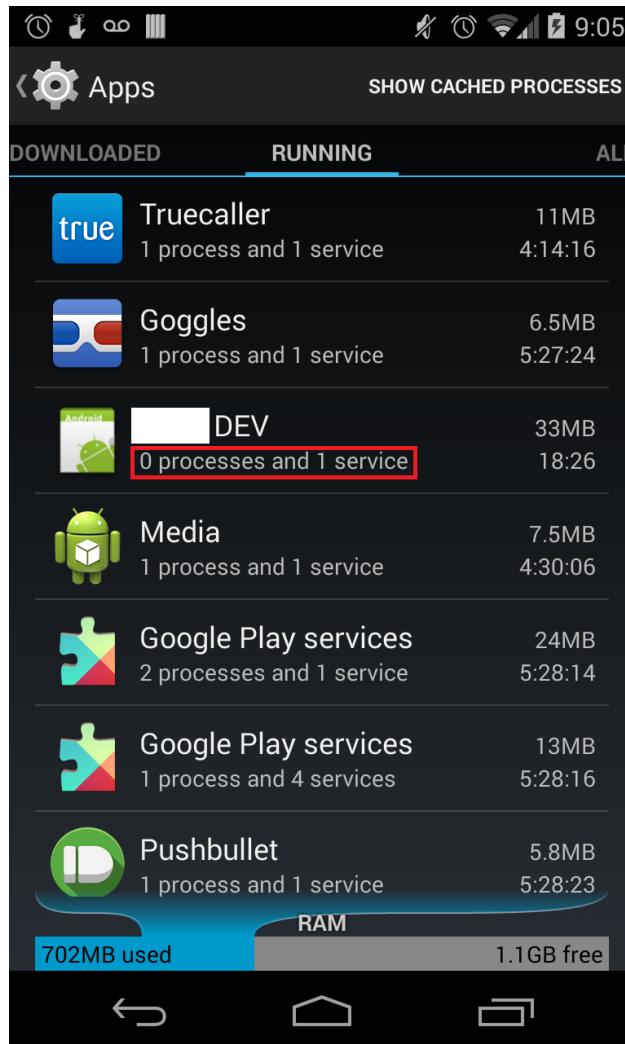
Activities



- Componente que normalmente está relacionado a uma *interface de usuário* e suas funcionalidades;
- Exemplos: tela que lista tarefas do dia, tela para inserir ou editar uma tarefa;
- O app pode ter uma ou várias Activities. Para criarmos um fluxo entre Activities, podemos usar *Intents*:

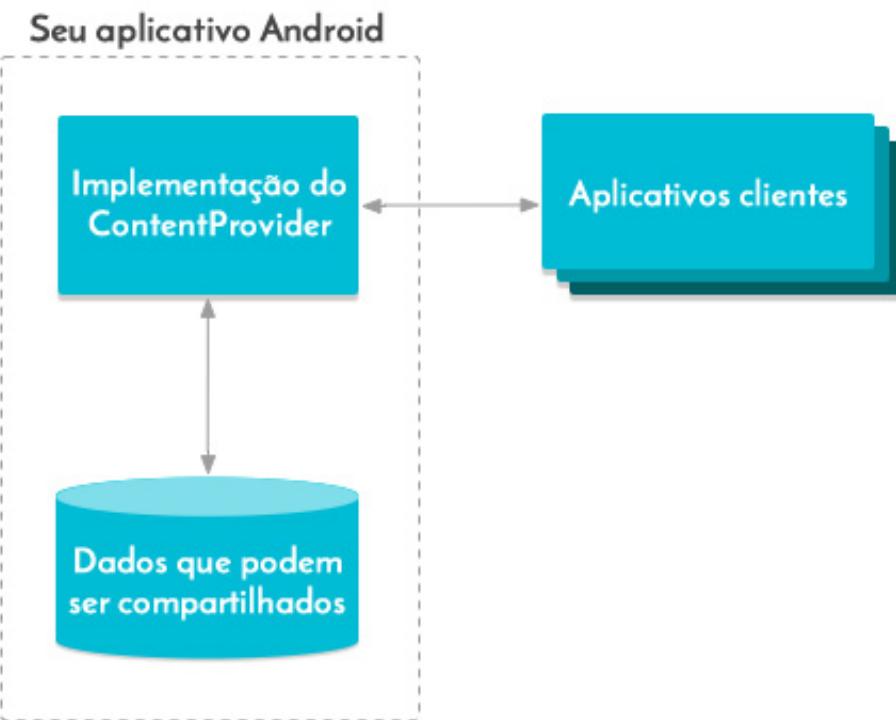
```
Intent intent = new Intent(this, SegundaActivity.class);  
startActivity(intent);
```

Services



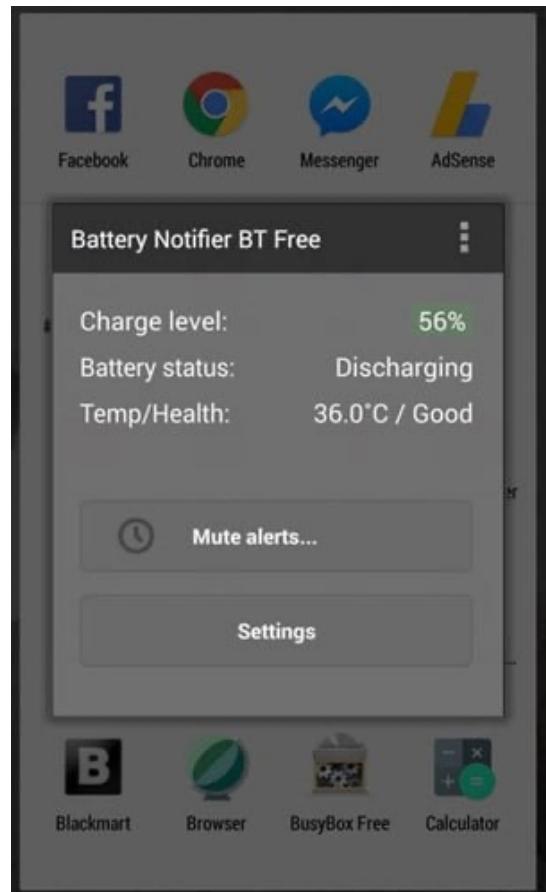
- Componentes que normalmente estão associados à realização de operações de longa duração e não fornecem qualquer interface gráfica ao usuário;
- Processos em segundo plano para lidar com trabalhos de longo prazo;
- Exemplos: notificações e sincronização da programação de dados.
- Permite que o app permaneça ativo enquanto uma outra atividade é executada em segundo plano.

Content Provider



- Componente que permite compartilhamento de dados entre diferentes aplicações;
- Gerencia o acesso a um repositório central de dados fornecidos por um aplicativo;
- Compartilhamento de dados de maneira segura, padronizada e controlada;
- Permite salvar ou consultar informações do app.

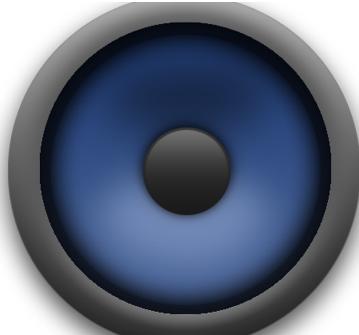
Broadcast Receiver



- Componentes responsáveis por receber eventos que são enviados pelo sistema ou até mesmo por outras aplicações;
- Exemplos de eventos: pouca bateria, desligamento da tela e chamada recebida;
- Esses eventos são enviados em forma de Intent:

```
if (action.equals(Intent.ACTION_BATTERYLOW)) {  
    Intent i = new Intent(context, ServiceBattery.class);  
    context.stopService(i);  
} else if (action.equals(Intent.ACTION_BATTERY_OKAY)) {  
    Intent i = new Intent(context, ServiceBattery.class);  
    context.startService(i);  
}
```

Vamos conhecer o Android - Prática



App Music
Android

\$./adb shell

➤ Activity

am start -n com.android.music/com.android.music.MusicBrowserActivity

am start -n com.android.music/com.android.music.MediaPlaybackActivity

➤ Service

\$ am startservice -n

com.android.music/com.android.music.MediaPlaybackService

\$ am stopservice -n

com.android.music/com.android.music.MediaPlaybackService

➤ Broadcast Receiver

\$ am broadcast -a android.intent.action.BOOT_COMPLETED

➤ ContentProvider

content query --uri content://sms/sent

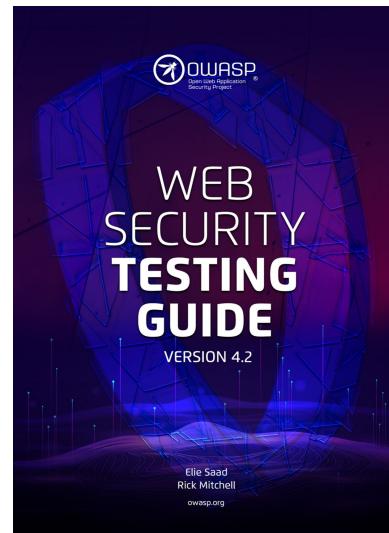
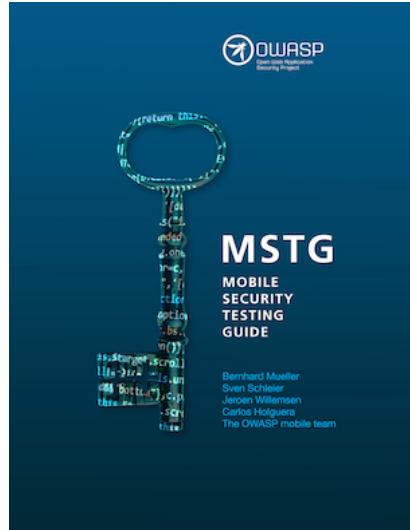


Problemas de segurança

Vamos conhecer algumas vulnerabilidades comuns



OWASP



- Comunidade aberta que capacita organizações com o objetivo de desenvolver, adquirir e manter a segurança;
- Organização sem fins lucrativos que conta com a colaboração de profissionais e entusiastas em segurança de todo o mundo;
- Iniciou com foco em aplicações *Web*, mas atualmente também foca em segurança de aplicações móveis;
- Disponibiliza gratuitamente artigos, metodologias, diretrizes, documentações, ferramentas e afins sobre segurança;
- Projeto Top 10 Mobile e Top 10 Web.

Top 10 Mobile

M1: Improper Platform Usage

Vulnerabilidades derivadas do uso indevido dos controles de segurança da plataforma

M2: Insecure Data Storage

Vulnerabilidades relacionadas ao grau de proteção dos dados armazenados pelo app

M3: Insecure Communication

Vulnerabilidades derivadas da comunicação entre clientes (app) e servidor (API)

M4: Insecure Authentication

Vulnerabilidades relacionadas a problemas de autenticação.

M5: Insufficient Cryptography

Vulnerabilidades relacionadas a problemas em serviços de criptografia

M6: Insecure Authorization

Vulnerabilidades relacionadas a serviços inseguros de autorização.

M7: Client Code Quality

Inclui vulnerabilidades de entradas não confiáveis para chamadas de método feitas no código móvel

M8: Code Tampering

Vulnerabilidade relacionada a métodos que alteram o Código do app para gerar outras versões

M9: Reverse Engineering

Inclui vulnerabilidades associadas à leitura e análise do código fonte do app.

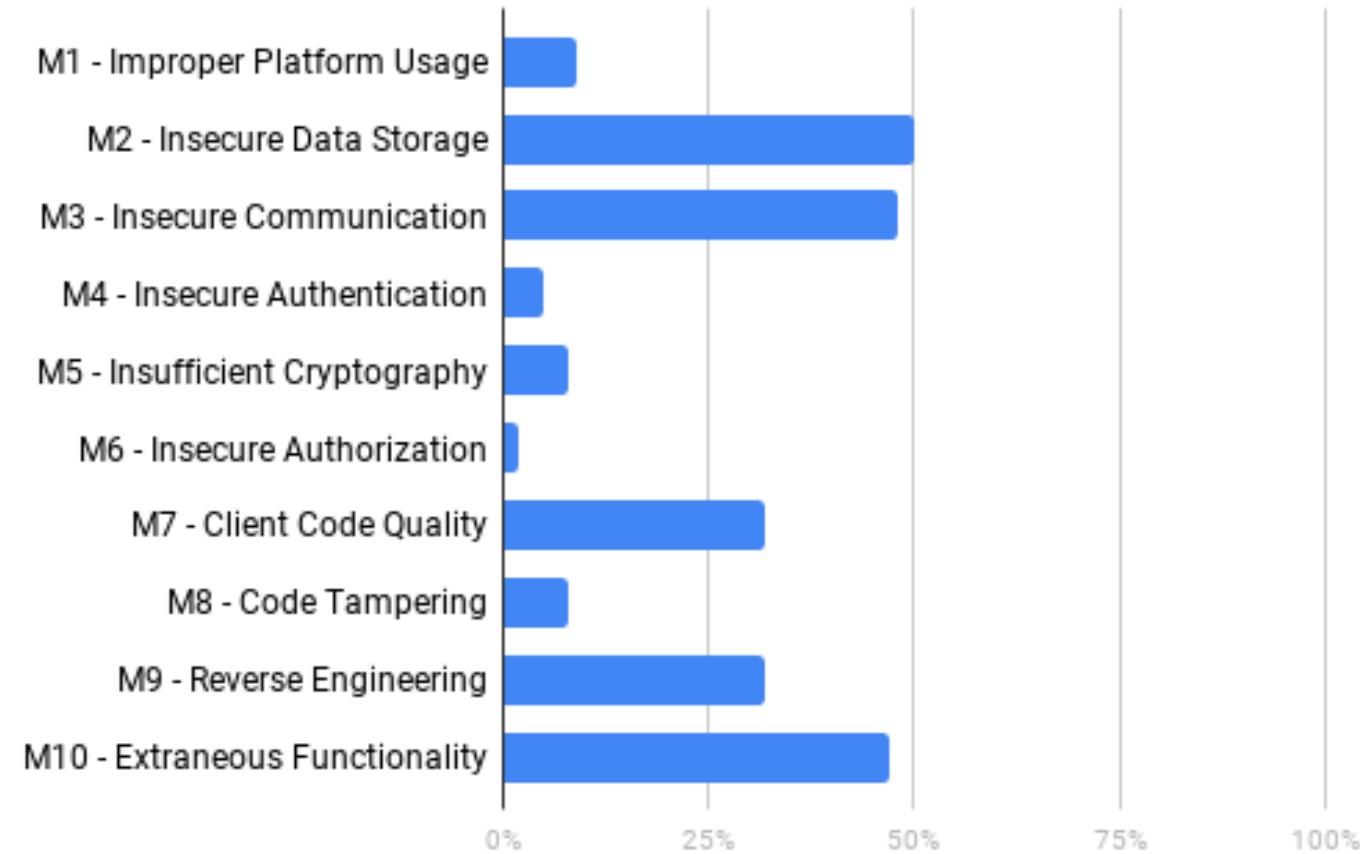
M10: Extraneous Functionality

Funções ou segredos escondidos dentro do aplicativo que permitem realizar ações maliciosas



Top 10 Mobile

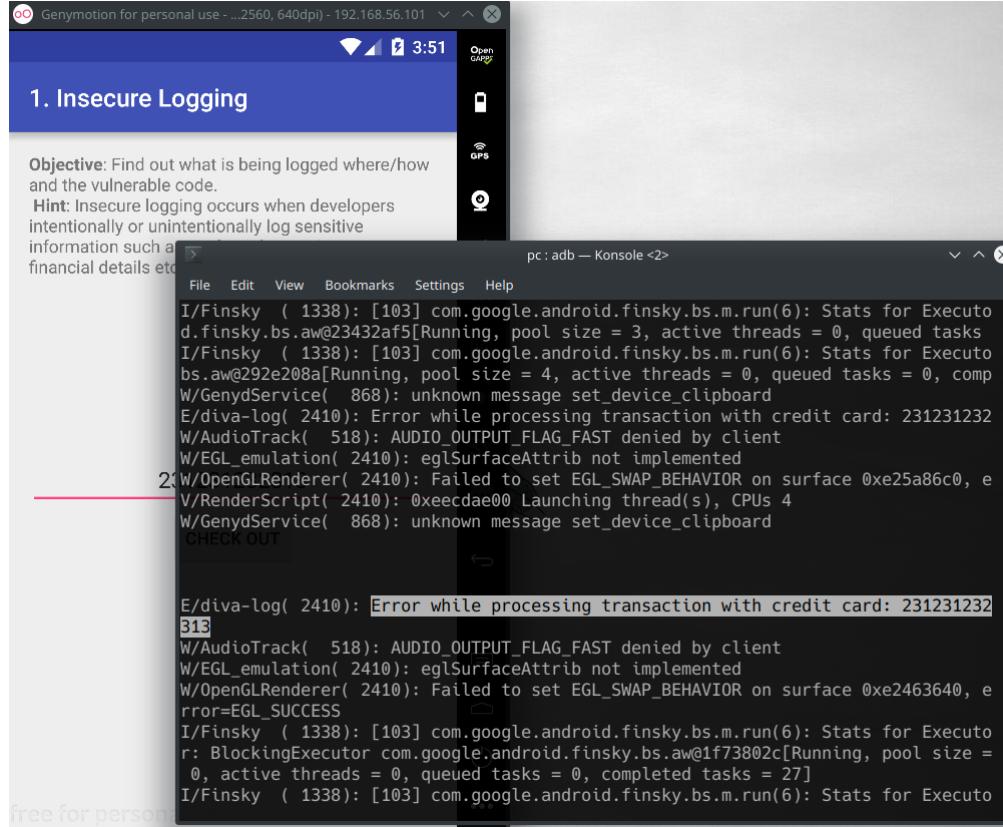
OWASP MOBILE TOP 10 VIOLATION RATES



Algumas Vulnerabilidades

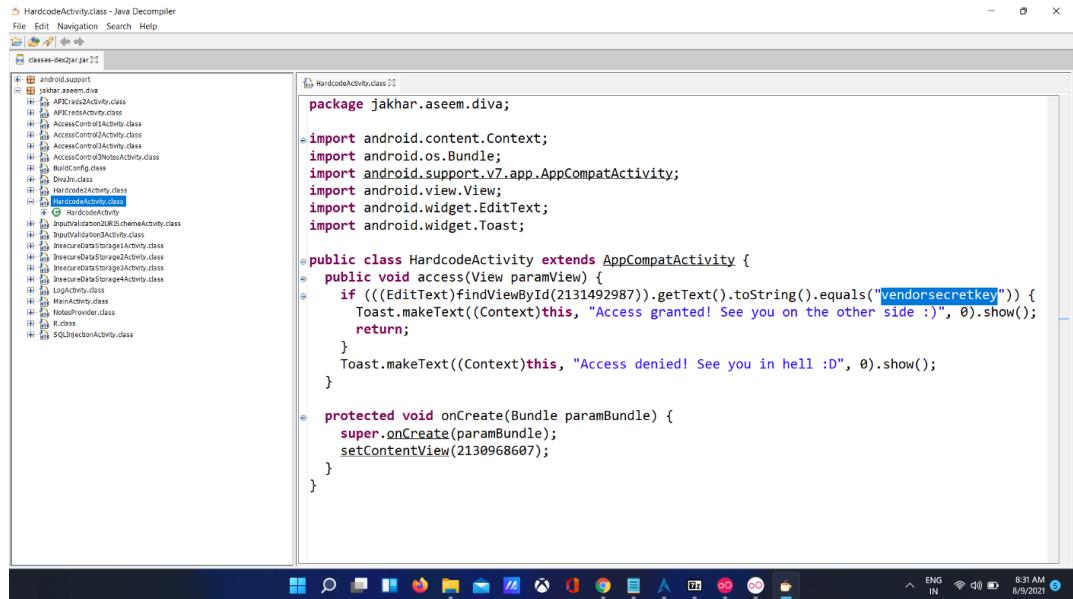
1. Log Inseguro - *Insecure Logging*
2. Problemas de Codificação - *Hardcoding Issues*
3. Armazenamento Inseguro de Dados - *Insecure Data Storage*
4. Problemas de Validação de Entrada - *Input Validation Issues*
5. Problemas de Controle de Acesso - *Access Control Issues*
6. Comunicação Insegura – *Insecure Communication*

Log Inseguro



- Método usado por desenvolvedores para rastrear o código e observar avisos ou erros;
- Logs são armazenados no dispositivo, possibilitando serem consultados em seguida;
- Portanto, o real problema é quando dados sensíveis são gravados em logs;
- Nesses casos, outros aplicativos podem obter acesso e ler os logs para obter informações confidenciais;
- **Método de mitigação:** não armazenar dados sensíveis em logs.

Problemas de Codificação



The screenshot shows a Java decompiler interface with the title "HardcodeActivity.class - Java Decompiler". The left pane displays a tree view of class files, including "jakhar.aseem.diva.HardcodeActivity.class" which is currently selected. The right pane shows the decompiled code for this class:

```
package jakhar.aseem.diva;

import android.content.Context;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class HardcodeActivity extends AppCompatActivity {
    public void access(View paramView) {
        if (((EditText)paramView).getText().toString().equals("vendorsecretkey")) {
            Toast.makeText((Context)this, "Access granted! See you on the other side :)", 0).show();
            return;
        }
        Toast.makeText((Context)this, "Access denied! See you in hell :D", 0).show();
    }

    protected void onCreate(Bundle paramBundle) {
        super.onCreate(paramBundle);
        setContentView(2130968607);
    }
}
```

The taskbar at the bottom shows various application icons, and the system tray indicates the date and time as 8:31 AM on 8/9/2021.

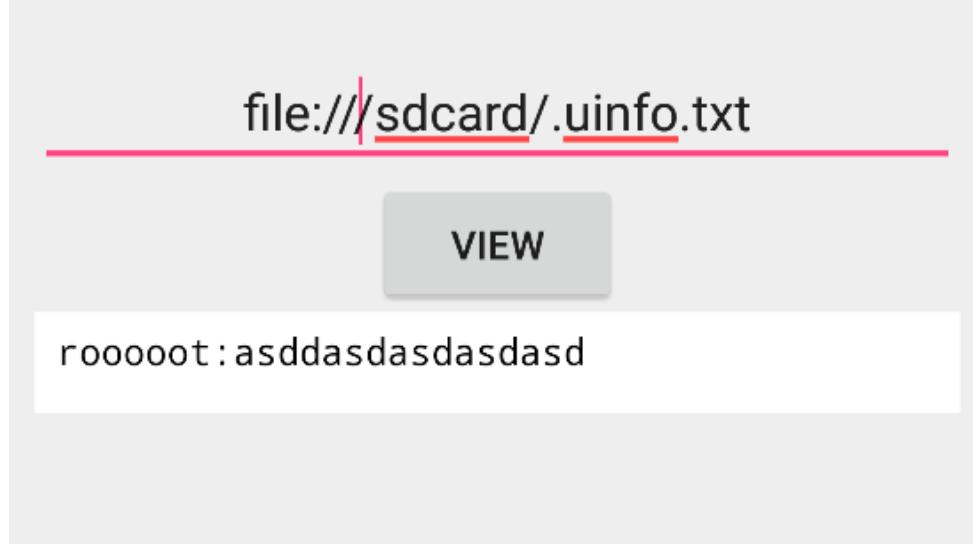
- Vulnerabilidades causadas por desenvolvedores que escrevem credenciais ou informações confidenciais no código-fonte da aplicação;
- Tais vulnerabilidades podem ser dar acesso a funções críticas da aplicação a usuários sem real autorização;
- **Método de mitigação:** sempre verificar se algum dado sensível está incluído no código-fonte da aplicação.

Armazenamento Inseguro de Dados

```
root@vbox86p:/data/data/jakhar.aseem.diva/shared_prefs #  
ls -l  
-rw-rw---- u0_a51 u0_a51 157 2019-03-26 08:07 jakhar.aseem.diva_preferences.xml  
root@vbox86p:/data/data/jakhar.aseem.diva/shared_prefs #  
cat jakhar.aseem.diva_preferences.xml <  
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>  
<map>  
    <string name="password">foooooooo</string>  
    <string name="user">fooooo</string>  
</map>  
root@vbox86p:/data/data/jakhar.aseem.diva/shared_prefs #
```

- Essas vulnerabilidades ocorrem quando os desenvolvedores assumem que usuários ou *malwares* não terão acesso ao sistema de arquivos do dispositivo;
- Caso haja a necessidade de armazenar dados confidenciais no dispositivo, deve-se criptografá-los antes;
- **Método de mitigação:** sempre validar qualquer dado de entrada.

Problema de Validação de Entrada



- Vulnerabilidade associada com a entrada de dados não validados pelo desenvolvedor antes de, por exemplo, armazená-los ou enviá-los ao serviço *back-end*;
- Se não há qualquer validação dos dados de entrada digitados pelo usuário, então é possível entrar com comandos personalizados para ler dados sensíveis ou executar scripts maliciosos.
- **Método de mitigação:** caso haja a necessidade de armazenar dados confidenciais no dispositivo, deve-se criptografá-los antes.

Problemas de Controle de Acesso

10. Access Control Issues - Part 2

Objective: You are able to access the Third Party app TVEETER API credentials after you have registered with Tweeter. The user can register with the application and the vendor gives the user the API credentials from the PIN. This is a need to see the components.

Tweeter API Credentials

TVEETER API Key: secrettveeterapikey

API User name: diva2

API Password: p@ssword2

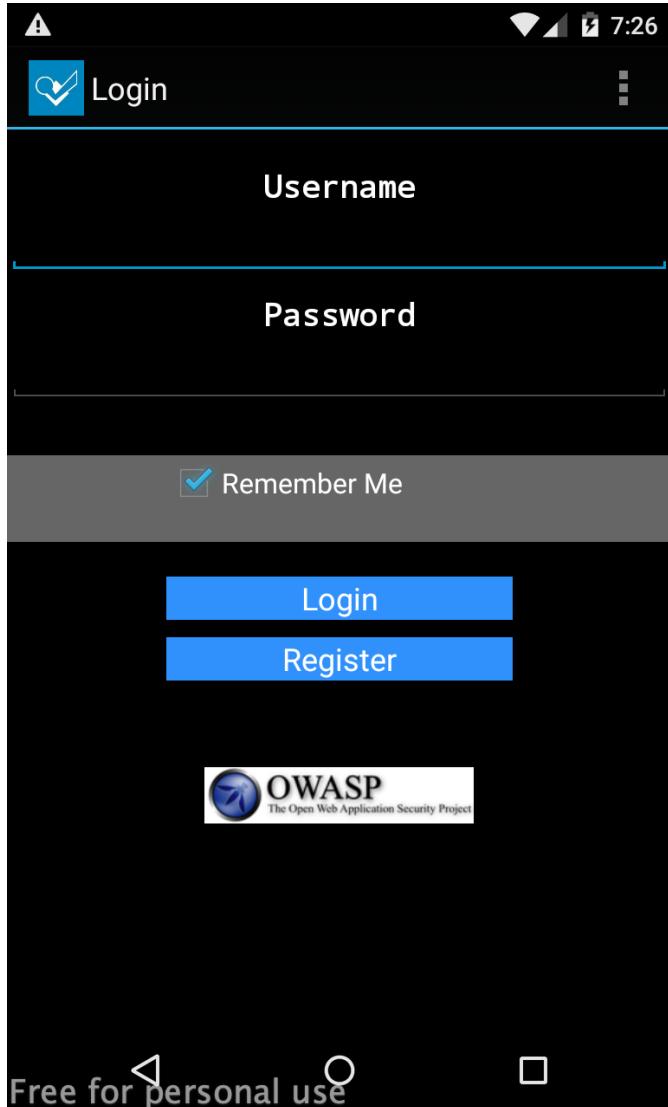
Hint: Components other apps or use and some may also be available. Components such as providers are provided by the developer.

Register Now. Already Registered.

VIEW TVEETER API CREDENTIALS

- Essas vulnerabilidades ocorrem em casos em que o aplicativo fornece funcionalidades específicas para usuários com autorização (Login -> Home);
- Desse modo, o aplicativo deve estar corretamente configurado para não permitir acesso a usuários que não tenham real autorização para tanto;
- **Método de mitigação:** assegurar que a Activity não pode ser acessada externamente.

Comunicação Insegura



- Vulnerabilidades associadas com cenários em que ocorre comunicação entre o app e um serviço back-end;
- Tais vulnerabilidades ocorrem quando não há proteção ao tráfego de rede.
- **Método de mitigação:** assegurar que os dados de tráfego de rede entre app e serviço back-end estejam sendo enviados de forma segura.

PARTE 2

Vamos praticar?





Ferramentas

Quais ferramentas estão ao nosso favor?

Ferramentas

JD-GUI



JD-GUI

eSoftner



APK tool

Android SDK
Manager



Genymotion



Dex2jar



BurpSuite



Comandos importantes – Parte 1

AVD - Android Virtual Device

-- Listas AVDs disponíveis: /Android/sdk/tools/bin

```
$ ./avdmanager list avd ou emulator -list-avds
```

-- Executar AVD via linha de comando: /Android/sdk/tools/

```
$ ./emulator @nome_AVD
```

ADB - Android Debug Bridge

-- Verificar dispositivos disponível (AVD em execução): /Android/sdk/platform-tools

```
$ ./adb devices
```

-- Entrar no emulador: /Android/sdk/platform-tools

```
$ ./adb shell
```

-- Listar diretório do device:

```
$ ls -la
```

Comandos importantes – Parte 2

ADB - Android Debug Bridge

-- Enviar arquivo para qualquer diretório do dispositivo: /Android/sdk/platform-tools

```
$ ./adb push teste.txt /data/local/tmp
```

-- Fazer download de um arquivo e salvar localmente: /Android/sdk/platform-tools

```
$ ./adb pull /data/local/tmp/teste.txt
```

-- Listar diretório do device:

```
$ ls -la
```

-- Instalar app no device: /Android/sdk/platform-tools

```
$ ./adb install /path_app/app.apk
```

-- Desinstalar app: /Android/sdk/platform-tools

```
$ ./adb uninstall /path_app/app.apk
```

-- Acessar um componente específico do app: /Android/sdk/platform-tools

```
$ ./adb shell am start path_Activity
```

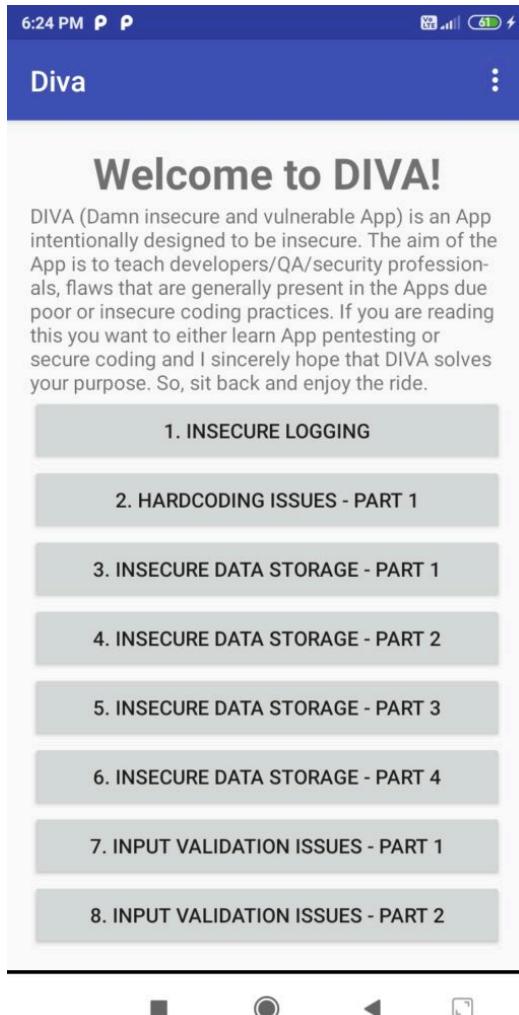


Aplicações alvo

Conhecendo as aplicações alvo



DIVA



- DIVA é um aplicativo projetado intencionalmente para ser inseguro;
- Ensinar aos desenvolvedores/QA/profissionais de segurança, falhas que geralmente estão presentes nos apps devido a práticas de codificação ruins ou inseguras;
- Abrange vulnerabilidades comuns em aplicativos Android, desde armazenamento inseguro, validação de entrada até problemas de controle de acesso.



Problemas de segurança

Vamos conhecer algumas vulnerabilidades comuns na nossa aplicação alvo

Sugestão de Metodologia



Extrair o código fonte
da aplicação a partir
do APK para análise



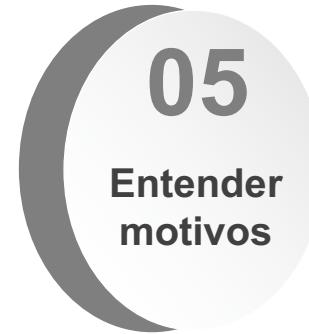
Usar a aplicação,
entender o fluxo e
disponibilidade
das
funcionalidades



Rotular possíveis
vulnerabilidades e
métodos de
identificação e
ataque.



Aplicar método
para identificar
ou atacar
vulnerabilidades



Entender as
possíveis razões
para a
vulnerabilidade

Identificando Vulnerabilidades - DIVA



The screenshot shows a mobile application titled "Diva". The top status bar indicates the time as 6:24 PM and battery level. The app's title "Diva" is displayed in the top left corner. A vertical ellipsis menu is visible in the top right corner. The main screen features a large heading "Welcome to DIVA!" followed by a descriptive paragraph about the app's purpose: "DIVA (Damn insecure and vulnerable App) is an App intentionally designed to be insecure. The aim of the App is to teach developers/QA/security professionals, flaws that are generally present in the Apps due poor or insecure coding practices. If you are reading this you want to either learn App pentesting or secure coding and I sincerely hope that DIVA solves your purpose. So, sit back and enjoy the ride." Below the text, there is a list of 8 numbered items, each represented by a grey button-like background:

- 1. INSECURE LOGGING
- 2. HARDCODING ISSUES - PART 1
- 3. INSECURE DATA STORAGE - PART 1
- 4. INSECURE DATA STORAGE - PART 2
- 5. INSECURE DATA STORAGE - PART 3
- 6. INSECURE DATA STORAGE - PART 4
- 7. INPUT VALIDATION ISSUES - PART 1
- 8. INPUT VALIDATION ISSUES - PART 2

At the bottom of the screen, there are standard mobile navigation icons: a square, a circle, a triangle, and a rectangle.

Engenharia reversa

Dex2jar

-- Converter app para arquivo .jar (Executar a partir do diretório do Dex2jar)

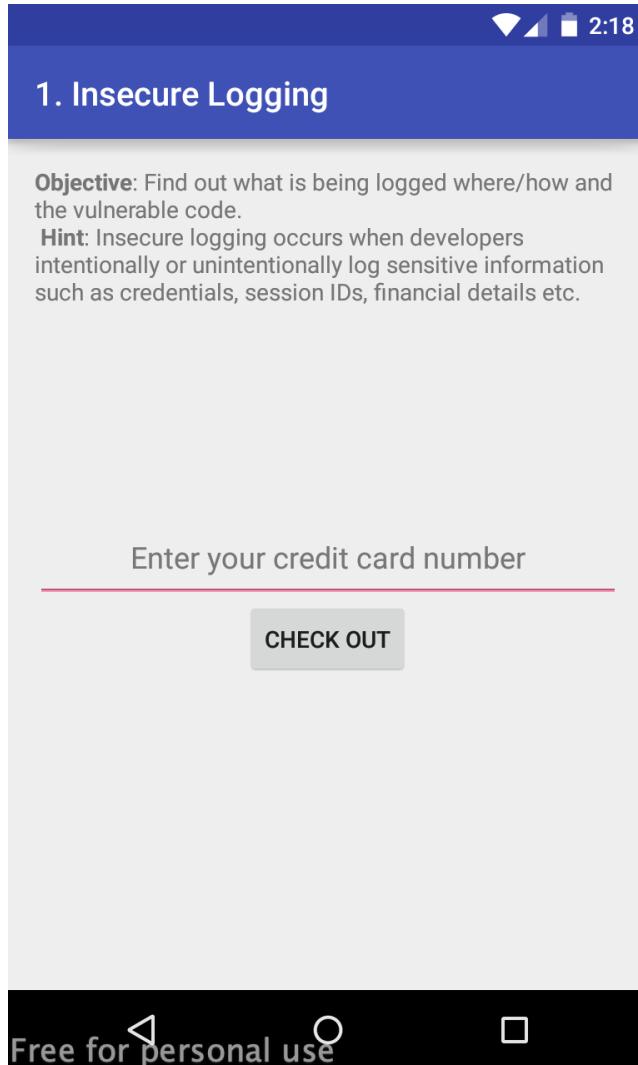
```
$ ./dex2jar.sh diva.apk -o diva.jar
```

JD-GUI

-- Abrir código-fonte do app a partir do arquivo .jar

```
$ java -jar jd-gui.jar diva.jar
```

1 - Log Inseguro



1. Execute o seguinte comando no terminal:

\$./adb shell logcat

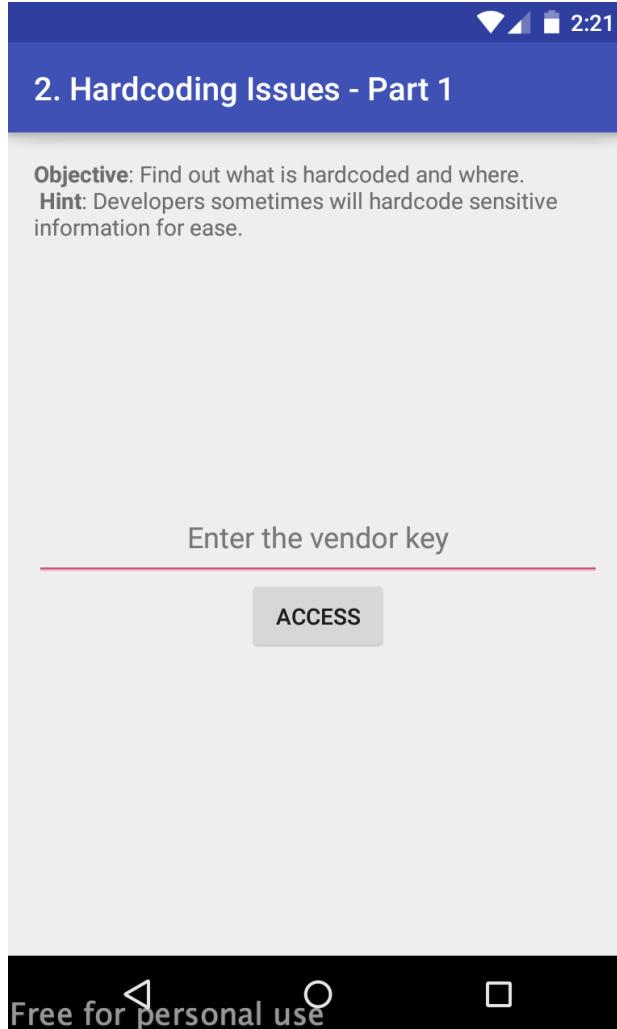
2. Insira um número de cartão de crédito

123456789

3. Verifique se o dado inserido é gerado via logcat

4. Identifique o falha no Código-fonte via JD-GUI.

2 - Problemas de Codificação



1. Insira qualquer código de acesso:

123456789

2. Identifique o código de acesso a partir do Código-fonte da aplicação

3. Insira o código de acesso identificado na aplicação

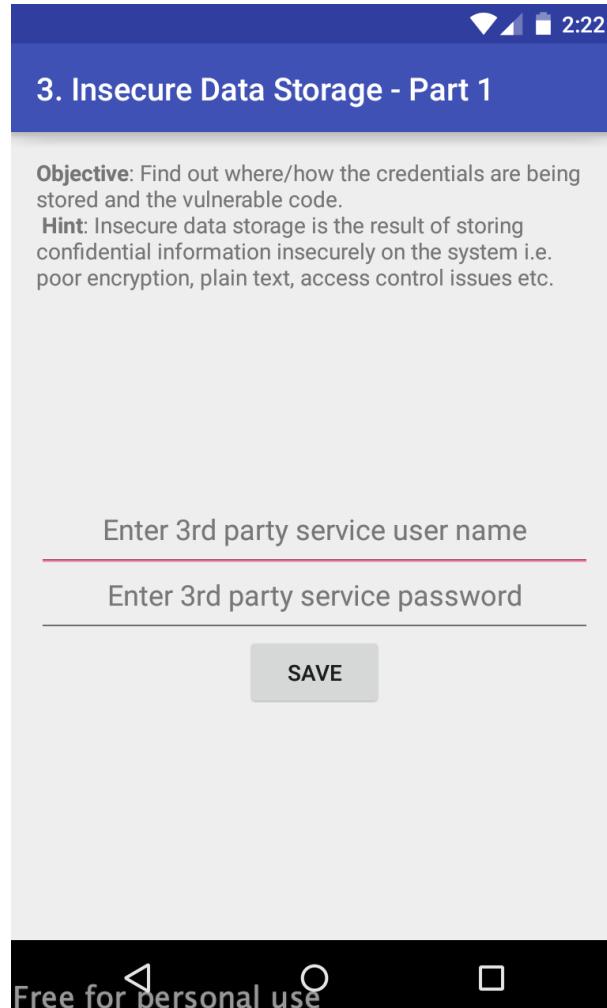
3 - Armazenamento de Dados

- Shared Preferences
- SQLite
- Armazenamento Interno
- Armazenamento Externo

3 - Armazenamento de Dados

- Shared Preferences
- SQLite
- Armazenamento Interno
- Armazenamento Externo

3 - Armazenamento Inseguro de Dados – PARTE 1



1. Identifique o diretório do app

\$./adb shell ps

ou

Verificar package no ManifestFile.xml

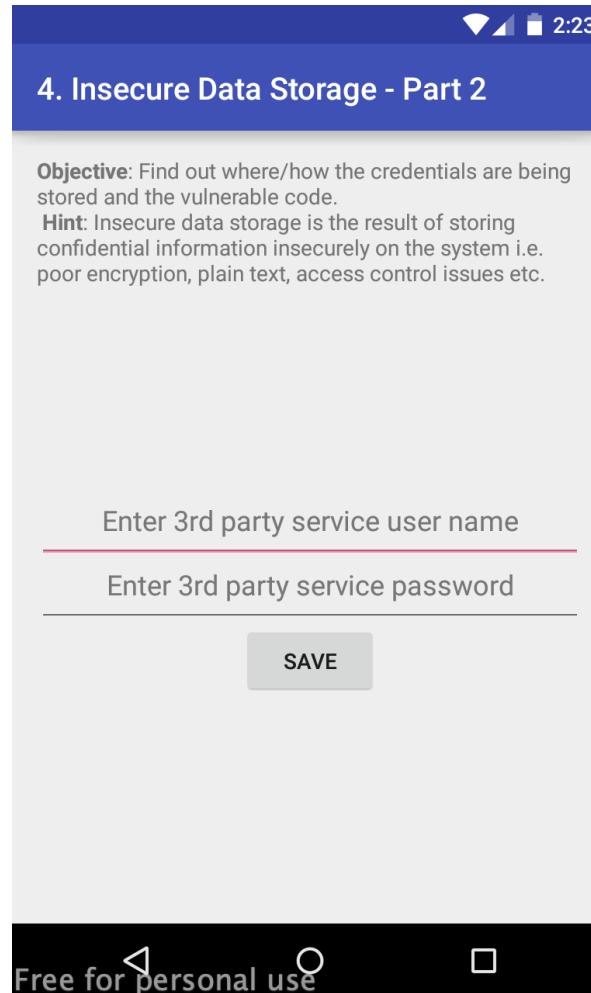
2. Entrar no armazenamento de dados do app

\$ cd /data/data/

3. Insira as credenciais no app para gerar dados

4. Acesse o diretório do sharedpreferences gerado e leia as informações no arquivo

3 - Armazenamento Inseguro de Dados – PARTE 2



1. Identifique o diretório do app

Verificar package no ManifestFile.xml

2. Entrar no armazenamento de dados do app

```
$ ./adb shell
```

```
$ cd /data/data/
```

3. Entrar no diretório databases

```
$ cd databases
```

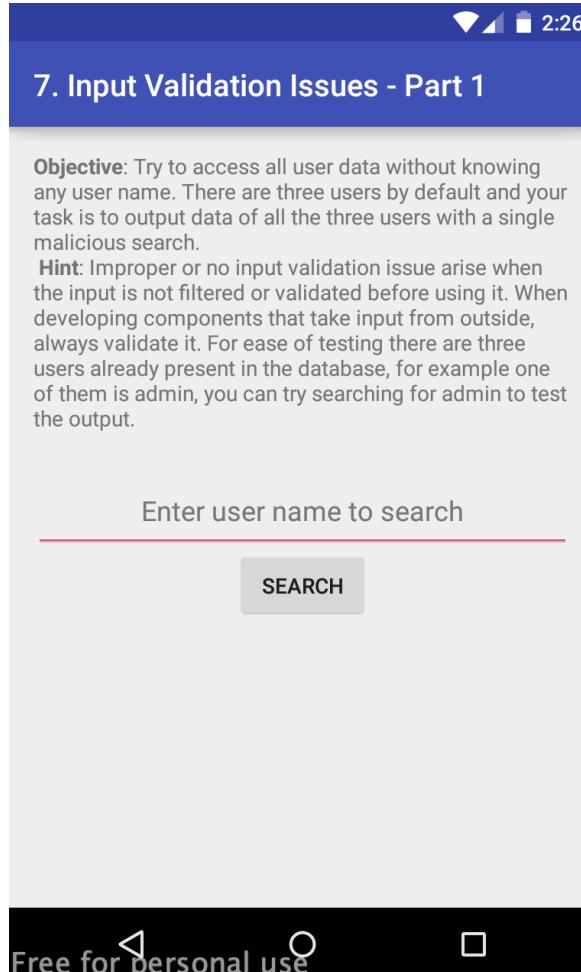
4. Executar comando sqlite3 no terminal

```
$ sqlite3
```

5. Ler a tabela disponível

```
$ select * from tablename;
```

4 - Problema de Validação de Entrada – SQL Injection



1. Habilite o logcat

```
$ ./adb shell logcat
```

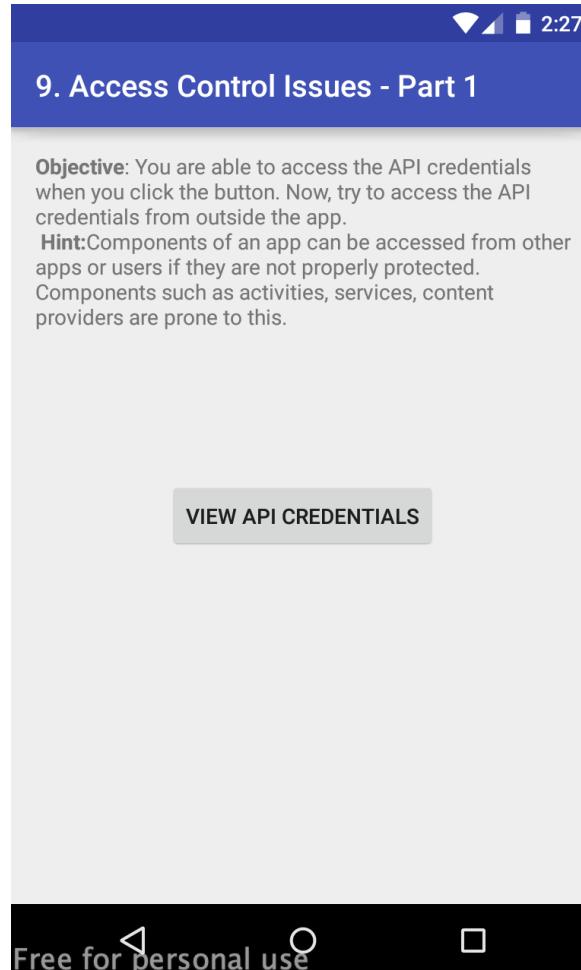
2. Teste a função de busca do app

3. Faça buscas e identifique um padrão

4. Tente burlar a busca inserindo um comando de consulta sql.

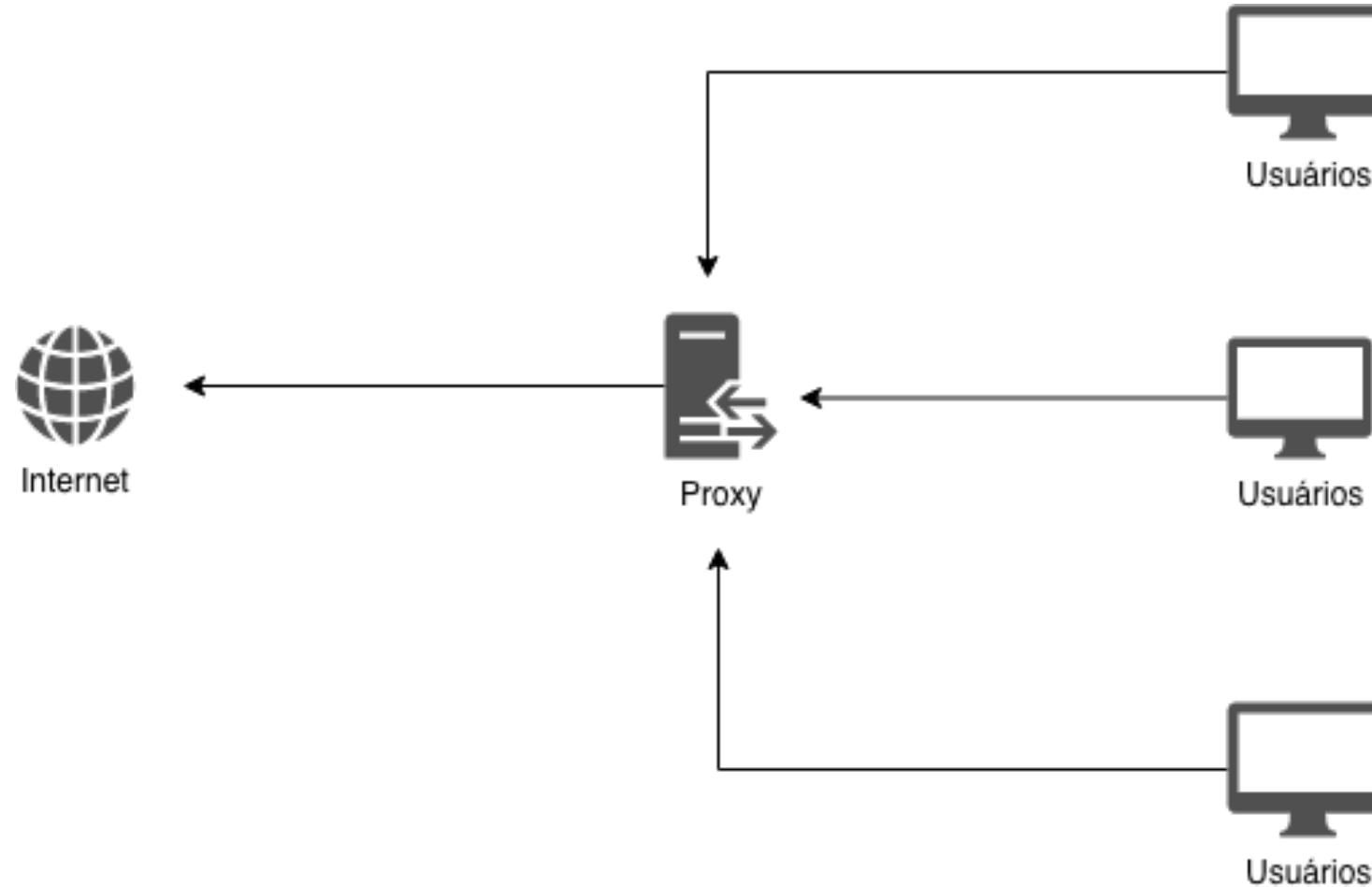
Mais consultas: <https://github.com/payloadbox/sql-injection-payload-list>

5 - Problemas de Controle de Acesso – Activity Vulnerável

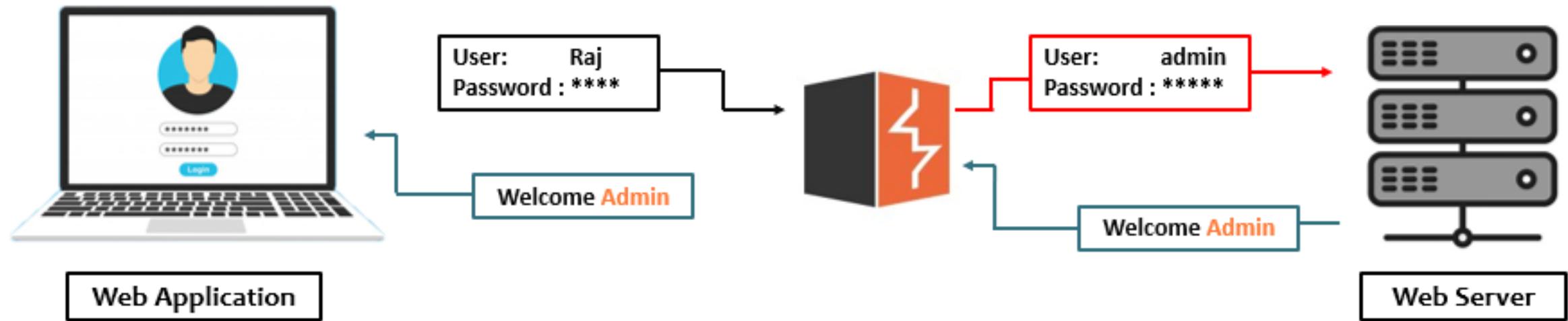


- 1. Analise o AndroidManifest.xml e verifique as definições da Activity**
- 2. Tente chamar a Activity via terminal usando o comando am**
`$./adb shell am start package_name./Activityname`

Identificando Vulnerabilidades – Comunicação Insegura



Identificando Vulnerabilidades – Comunicação Insegura



Configurando ambiente

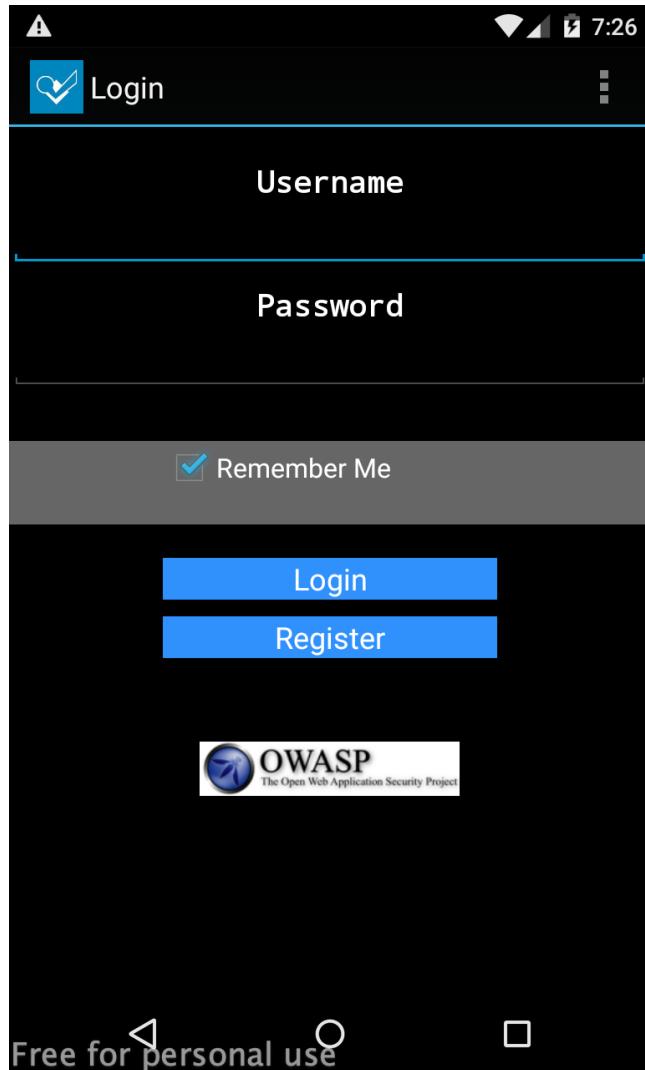
BurpSuite

- Criar proxy para a rede local
- Fazer *Download* do certificado digital e renomeá-lo com a extensão .cer

Emulador

- Alterar Proxy para manual e adicionar as informações da rede local
- Enviar certificado para o emulador
\$./adb push certificado.cer /sdcard/
- Realizar instalação do certificado digital (Configurações>Segurança>Instalar certificado a partir do SD card)

6 - Comunicação Insegura



- 1. Explore a aplicação e entenda o tráfego de rede**
- 2. Identifique um possível risco a partir do tráfego de rede**
- 3. Envie uma requisição para a função Repeater do Burp Suite e manipule ela.**

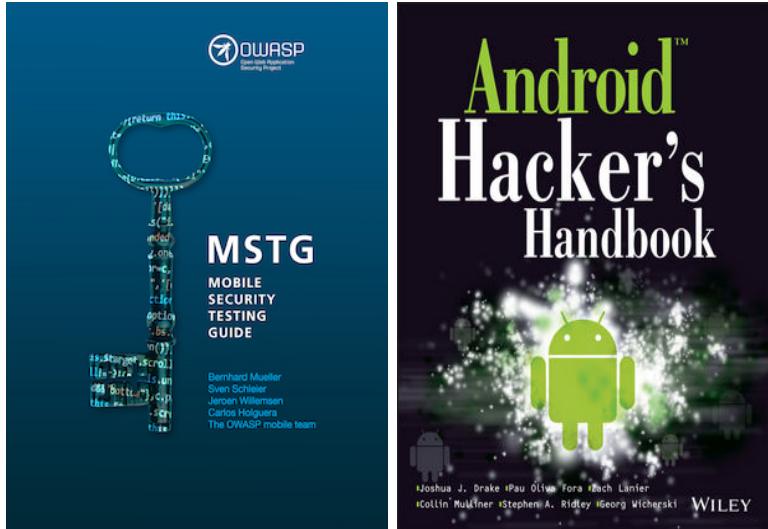


Indicações

Leituras, ferramentas, cursos e afins

Livros e Ferramentas

Livros



Ferramentas



Sites, Cursos e afins

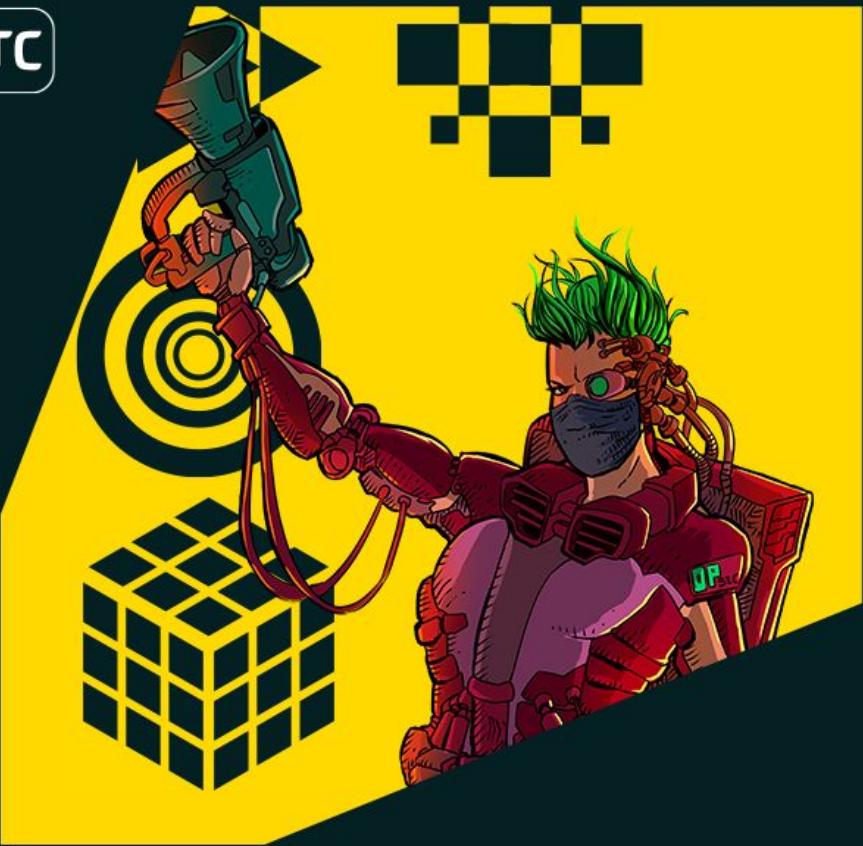
Sites



Cursos/Treinamentos



Roadsec 2022



O TC investe em Tech

Somos patrocinadores oficiais do

ROADSEC
2022 黑客文化
ROADSEC®

Material da oficina





Muito obrigado!

Dúvidas?