

Estácio

Desenvolvedor, cuidado!
O seu aplicativo pode
estar vulnerável

Me. Misael Costa Júnior

Quem sou eu?

Misael Costa Júnior



- Analista de Qualidade Sênior
- Orientador/Colaborador em projetos de pesquisa envolvendo a área de Teste de Software e Inteligência Artificial;
- Mestre e Doutorando em Ciências da Computação e Matemática Computacional na Universidade de São Paulo (USP)
- Bacharel em Ciência da Computação na Universidade Estadual do Piauí (UESPI)
- Certificações: CTFL (Certified Tester Foundation Level) e CTFL-AT (Certified Tester Foundation Level, Agile Tester)
- Experiência nos seguintes temas: Engenharia de Software, Teste de Software, Testes de API, Segurança (Android) e Performance.





Importante!!!



- O conteúdo ministrado nessa palestra tem como objetivo apresentar os principais erros 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 palestra pode ser usado como uma base de conhecimento inicial sobre *pentest* e segurança de aplicativos Android.

PARTE 1



Contextualização

01

- O que é Segurança?
- Especificação de critérios de segurança

Teste de Segurança

02

- O que é Teste de Segurança?
- Benefícios do Teste de Segurança
- Tipos de Teste de Segurança

PARTE 2



Contexto atual

01

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

02

Android

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

03

Problemas e Riscos de segurança

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

04

Indicações

- Livros e Ferramentas
- Sites, Cursos e afins

PARTE 1

Contextualização





Segurança

O que é segurança? Especificando critérios de segurança.

Casos Reais

g1

ECONOMIA

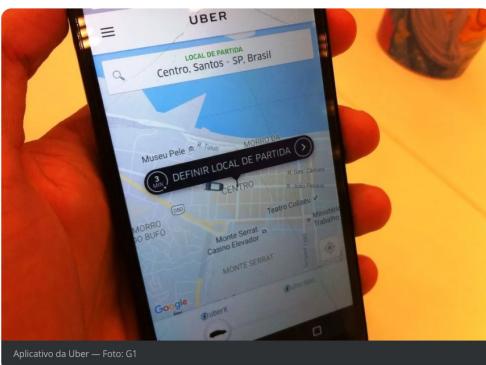
Q BUS

Uber avisa brasileiros que tiveram dados roubados em ataque que atingiu 57 milhões no mundo

Segundo a empresa, 196 mil brasileiros foram atingidos. Hackers roubaram nome, e-mail e número de celular de clientes.

Por G1

12/04/2018 11h40 · Atualizado há 5 anos



Aplicativo da Uber — Foto: G1

≡ MENU | G1

TECNOLOGIA E GAMES

27/04/2011 14h56 - Atualizado em 27/04/2011 14h57

Vazamento de dados da PSN é considerado o 5º maior da história

Classificação é realizada pelo site Dataloss DB. Site contabiliza mais de 70 milhões de registros afetados.

Altieres Rohr
Especial para o G1



Vazamento de dados da rede do PS3 é o quinto maior da história (Foto: Divulgação)

vazamento, ele precisa também ter incluído informações como endereços, informações bancárias ou números de documentos particulares.

O site **Dataloss DB** considera o vazamento dos dados de usuários da Playstation Network (PSN) da Sony o 5º maior da história.

Segundo a página, que registra todos os casos de vazamentos de dados, mais de 70 milhões de registros foram afetados.

Se o número do caso da PSN fosse revisado para 77 milhões, como a própria Sony divulga, o incidente passaria a ser o 4º pior da história, à frente de um vazamento ocorrido em 2009 em que o governo dos EUA perdeu um disco rígido com 76 milhões de registros.

Os números do Dataloss DB não consideram incidentes em que apenas nomes e e-mails foram vazados. Para o site considerar um

BBC NEWS BRASIL

Notícias Brasil Internacional Economia Saúde Ciência Tecnologia Vídeos

Facebook admite uso indevido de dados de 87 milhões de usuários, 443 mil no Brasil

4 abril 2018



REUTERS
Presidente e fundador da empresa testemunhará perante congressistas americanos em 11 de abril

Casos Reais

Home > Notícias > Segurança

Sony | Divisão de PlayStation confirma vazamento de dados

Por [Felipe Demartini](#) | Editado por [Wallace Moté](#) | 04 de Outubro de 2023 às 10h41

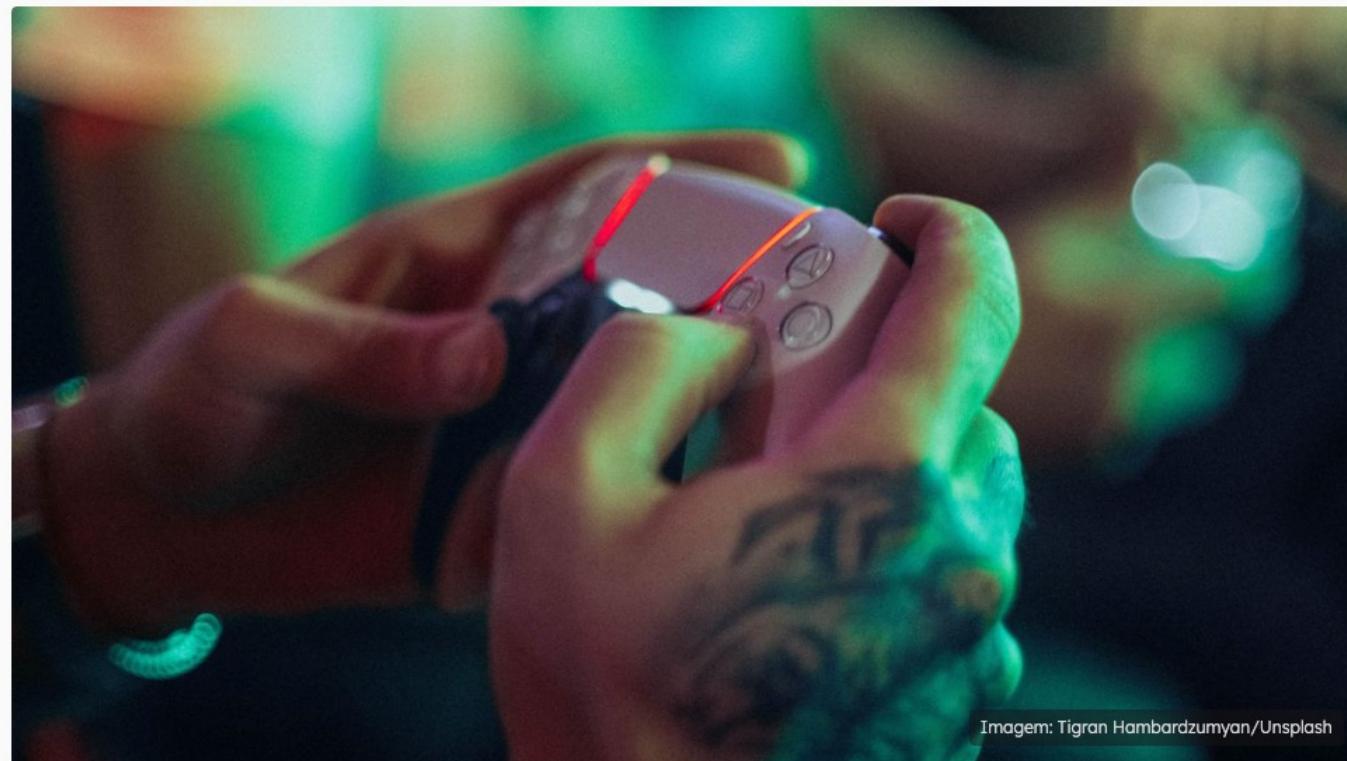


Imagem: Tigran Hambardzumyan/Unsplash

Casos Reais

Ataques DDoS recordes desafiam Google, Amazon e Cloudflare

Google e Amazon enfrentaram ataques oito vezes maiores que ataques DDoS anteriores, atingindo 398 milhões de solicitações por segundo



por [Redação](#)

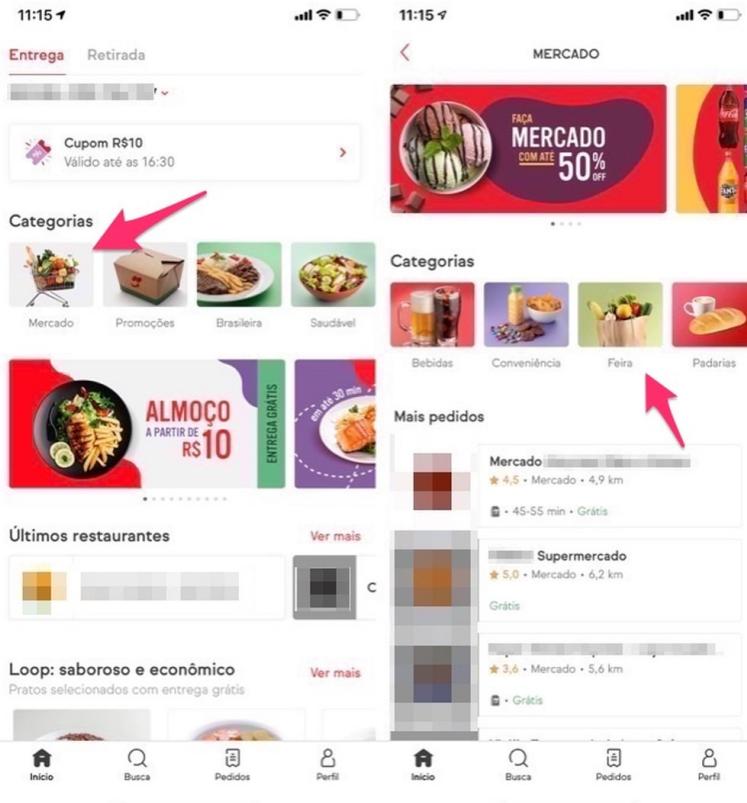
2:00 pm - 11 de outubro de 2023



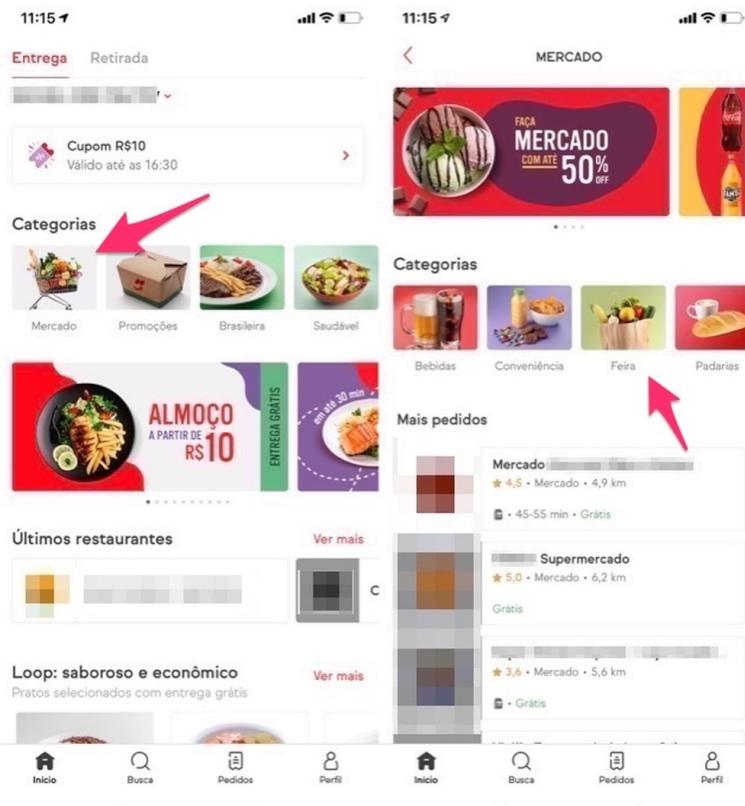
Segurança

“Grau em que um software *protege* as *informações e os dados* para que os usuários ou outros sistemas tenham *acesso aos dados adequados* aos seus tipos e *níveis de autorização.*”

Exemplo de Especificação

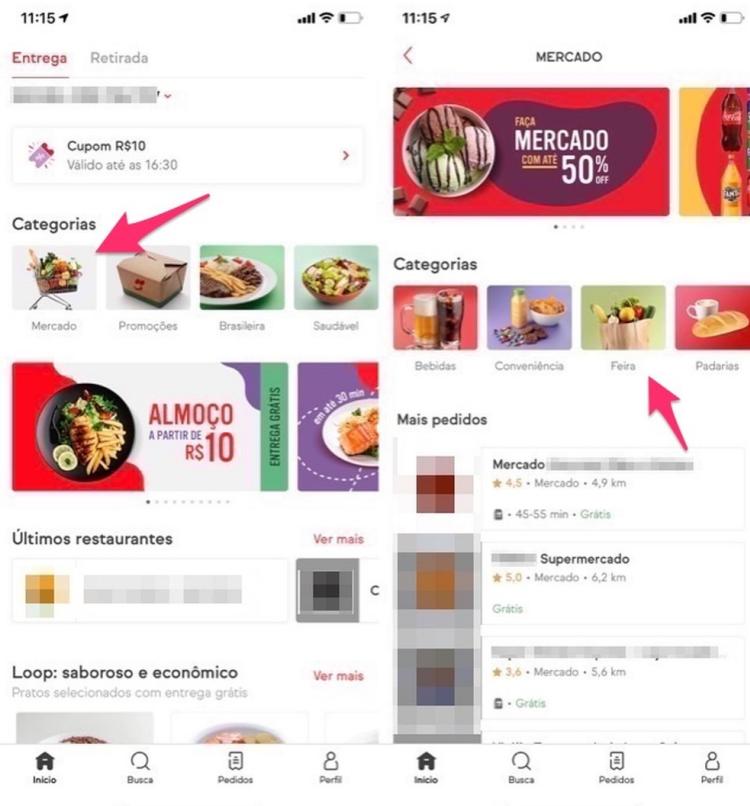


Exemplo de Especificação



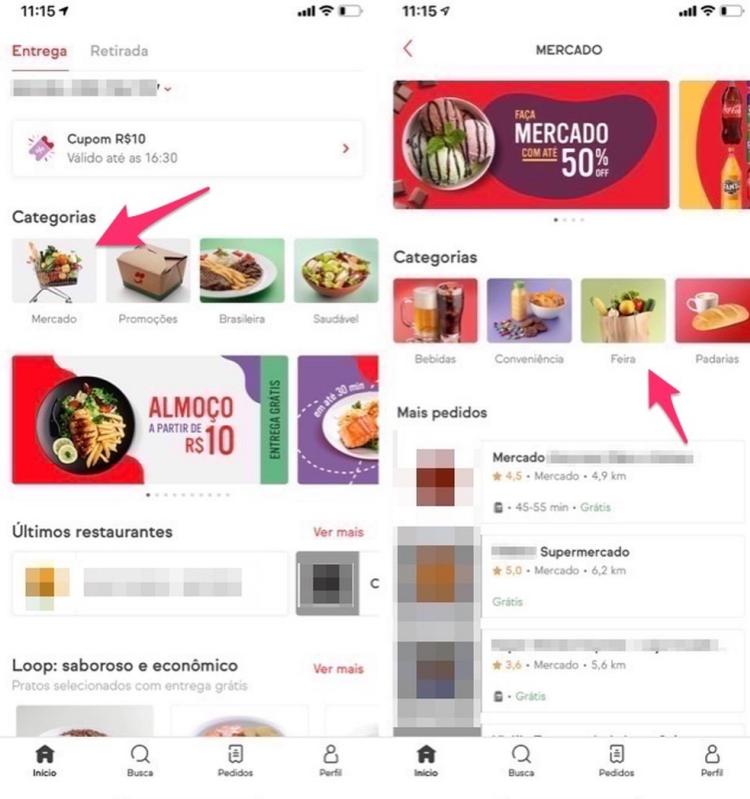
1. Autenticação: o software deve exigir credenciais de login seguras a ponto de evitar que hackers tenham acesso as contas dos usuários;

Exemplo de Especificação



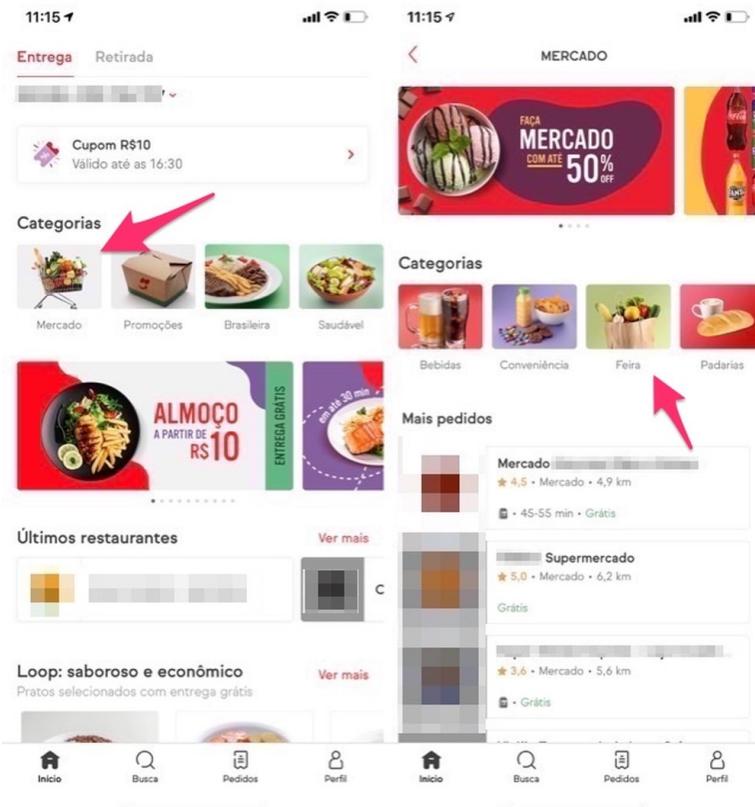
1. **Autenticação:** o software deve exigir credenciais de login seguras a ponto de evitar que hackers tenham acesso as contas dos usuários;
2. **Criptografia:** todas as transações e informações de pagamento devem ser criptografadas para evitar interceptação e leitura das informações confidenciais;

Exemplo de Especificação



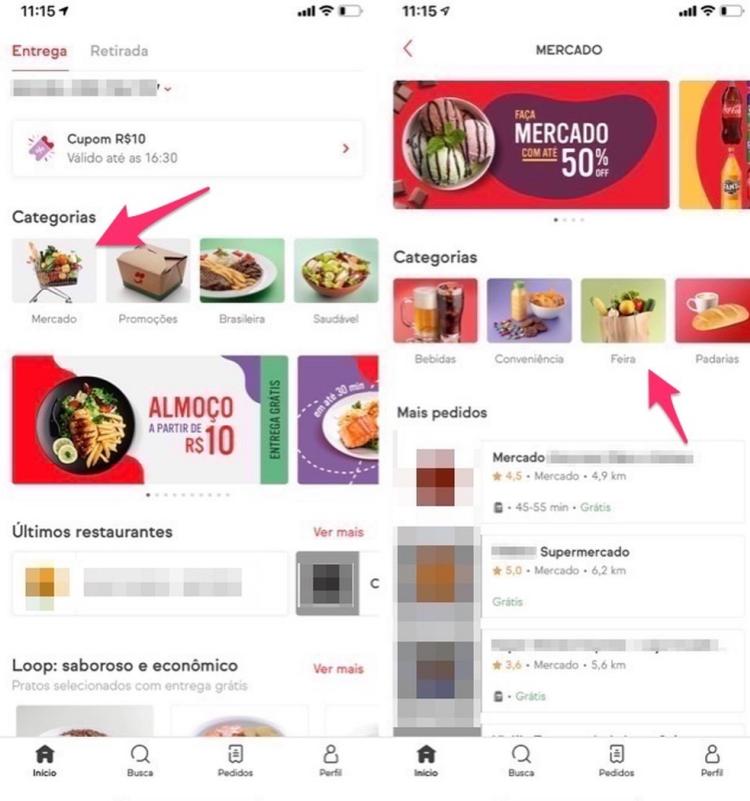
- 1. Autenticação:** o software deve exigir credenciais de login seguras a ponto de evitar que hackers tenham acesso as contas dos usuários;
- 2. Criptografia:** todas as transações e informações de pagamento devem ser criptografadas para evitar interceptação e leitura das informações confidenciais;
- 3. Controle de acesso:** o software deve garantir que somente usuários autenticados tenham acesso às suas funcionalidades;

Exemplo de Especificação



- 1. Autenticação:** o software deve exigir credenciais de login seguras a ponto de evitar que hackers tenham acesso as contas dos usuários;
- 2. Criptografia:** todas as transações e informações de pagamento devem ser criptografadas para evitar interceptação e leitura das informações confidenciais;
- 3. Controle de acesso:** o software deve garantir que somente os usuários autenticados tenham acesso às suas funcionalidades;
- 4. Gerenciamento de sessão:** o aplicativo deve gerenciar corretamente as sessões dos usuários;

Exemplo de Especificação



- 1. Autenticação:** o software deve exigir credenciais de login seguras a ponto de evitar que hackers tenham acesso as contas dos usuários;
- 2. Criptografia:** todas as transações e informações de pagamento devem ser criptografadas para evitar interceptação e leitura das informações confidenciais;
- 3. Controle de acesso:** o software deve garantir que somente os usuários autenticados tenham acesso às suas funcionalidades;
- 4. Gerenciamento de sessão:** o aplicativo deve gerenciar corretamente as sessões dos usuários;
- 5. Monitoramento:** o software deve monitorar continuamente a segurança para detectar e prevenir possíveis violações.



Teste de Segurança

O que é Teste de Segurança? Quais as principais técnicas de Teste de Segurança?

O que é Teste de Segurança?

“Processo para avaliar se os recursos da implementação do software são consistentes com os requisitos de segurança esperados.”

Michael Felderer, Matthias Buchler, Martin Johns, Achim D Brucker, Ruth Breu, and Alexander Pretschner.
Security testing: A survey. In Advances in Computers, volume 101, pages 1–51. Elsevier, 2016

Benefícios do Teste de Segurança

- 1 – Identificação antecipada de vulnerabilidades;
- 2 – Redução de riscos;
- 3 – Melhoria de reputação do negócio;
- 4 – Atenção e cuidado com o cliente;
- 5 – Melhora da Performance da aplicação.

Tipos de Teste de Segurança

Teste Funcional de Segurança

Teste de Vulnerabilidade

Tipos de Teste de Segurança

Teste Funcional de Segurança

“Avalia se os controles de segurança implementados estão funcionando adequadamente.”

Teste de Vulnerabilidade

Tipos de Teste de Segurança

Teste Funcional de Segurança

“Avalia se os controles de segurança implementados estão funcionando adequadamente.”

Teste de Vulnerabilidade

“Identifica vulnerabilidades que podem ser exploradas para obtenção de informações e controle da aplicação.”

Teste Funcional x Teste de Vulnerabilidade



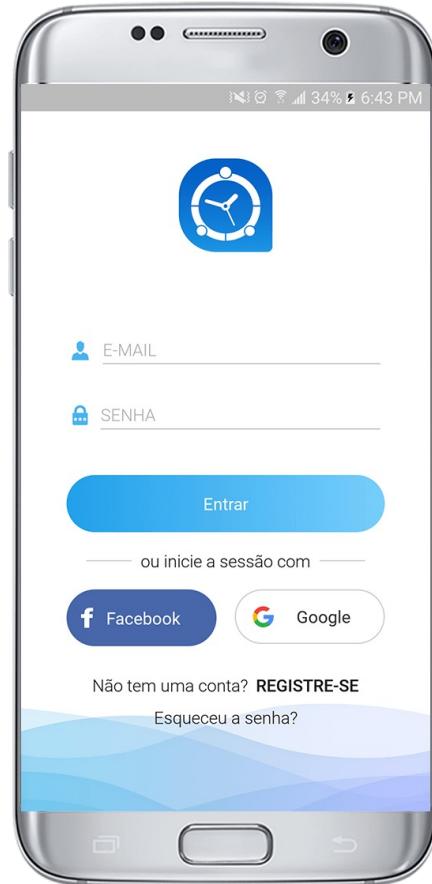
Teste Funcional x Teste de Vulnerabilidade

Cenário: Verificar autorização de acesso a funcionalidades

Dado que eu não esteja logado no sistema
E eu tente acessar uma funcionalidade que requer autorização

Quando eu tento acessar essa funcionalidade

Então devo ser redirecionado para uma página informando que não tenho permissão para acessar a funcionalidade



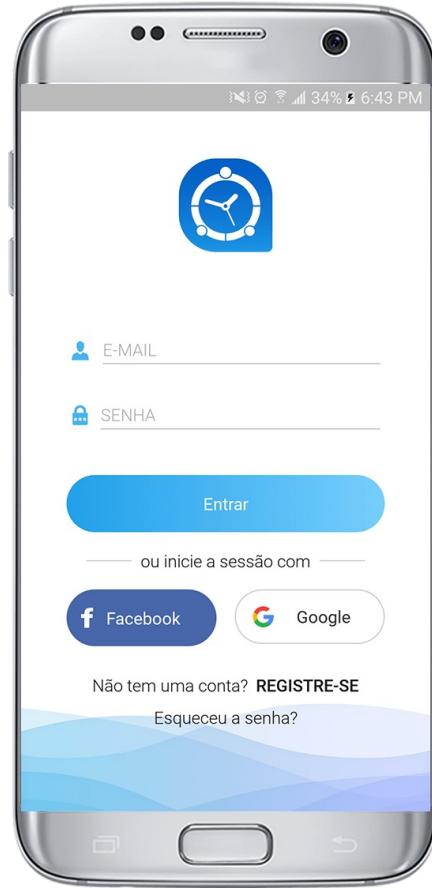
Teste Funcional x Teste de Vulnerabilidade

Cenário: Verificar autorização de acesso a funcionalidades

Dado que eu não esteja logado no sistema
E eu tente acessar uma funcionalidade que requer autorização

Quando eu tento acessar essa funcionalidade

Então devo ser redirecionado para uma página informando que não tenho permissão para acessar a funcionalidade



- SQL Injection
- Cross-site scripting (XSS)
- Injeção de código malicioso
- Teste de autenticação
- Teste de criptografia



Tipos de Teste de Vulnerabilidades



Penetration Testing



Avaliação de Vulnerabilidades

Tipos de Teste de Vulnerabilidades



Penetration Testing

- Verifica se é possível explorar uma vulnerabilidade específica na aplicação;
- Simula ataques para testar a capacidade de defesa da aplicação;
- Normalmente, usa-se uma variedade de ferramentas e técnicas para simular os ataques, incluindo engenharia social.



Avaliação de Vulnerabilidades

Tipos de Teste de Vulnerabilidades



Penetration Testing

- Verifica se é possível explorar uma vulnerabilidade específica na aplicação;
- Simula ataques para testar a capacidade de defesa da aplicação;
- Normalmente, usa-se uma variedade de ferramentas e técnicas para simular os ataques, incluindo engenharia social.



Avaliação de Vulnerabilidades

- Verifica se a aplicação está em conformidade com as normas e políticas de segurança;
- Não visa explorar uma vulnerabilidade, mas identificá-las para que sejam corrigidas;
- Identificar riscos e gerar testes;
- Pode ser automatizado por meio de ferramentas que possa escanear a aplicação em busca de vulnerabilidades.

PARTE 2

Vamos nos atualizar

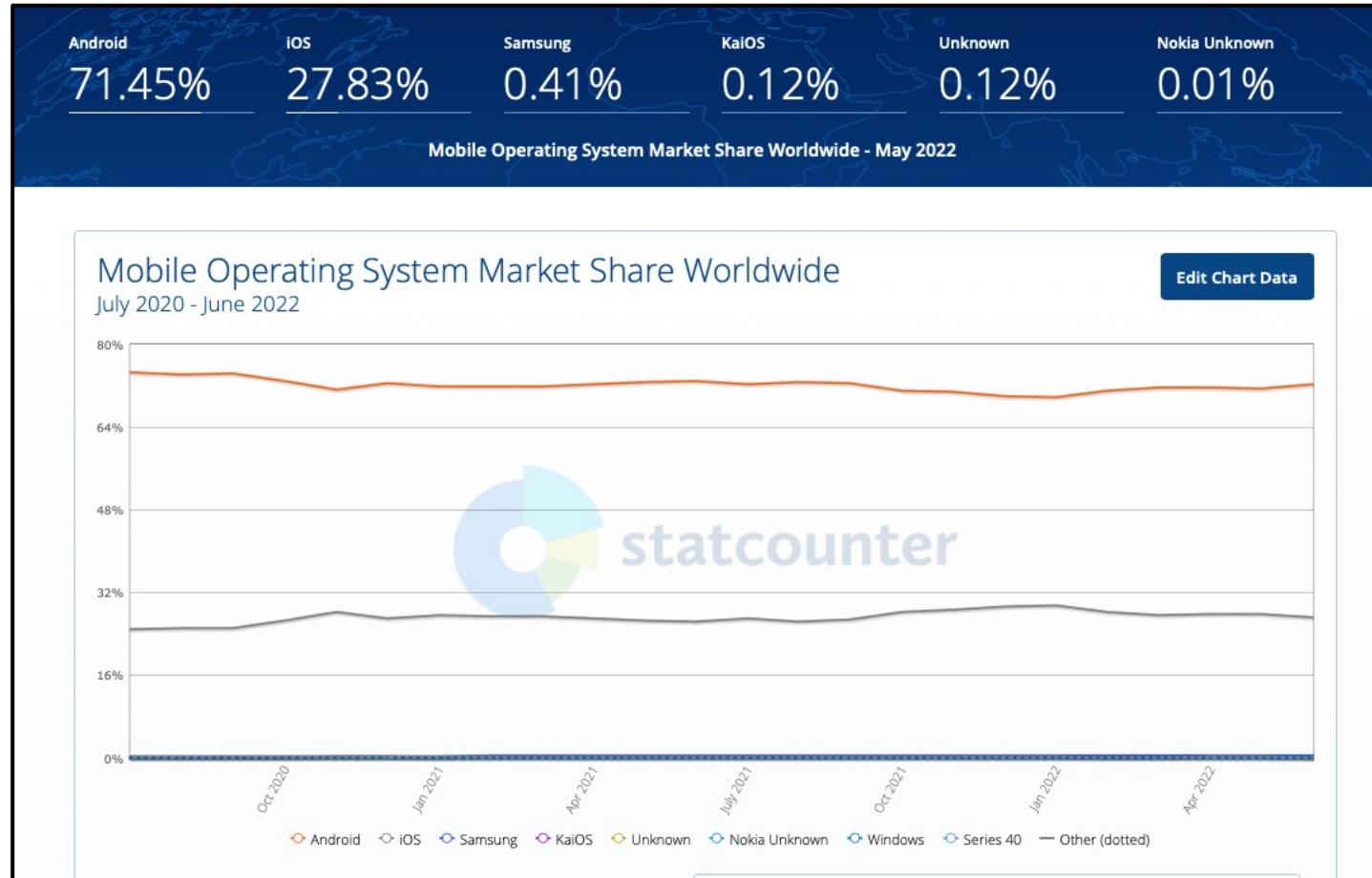




Contexto atual

Como está o contexto atual de aplicativos móveis?

Mercado de SOs móveis - Mundial



>70%

Dos dispositivos móveis usam o Android

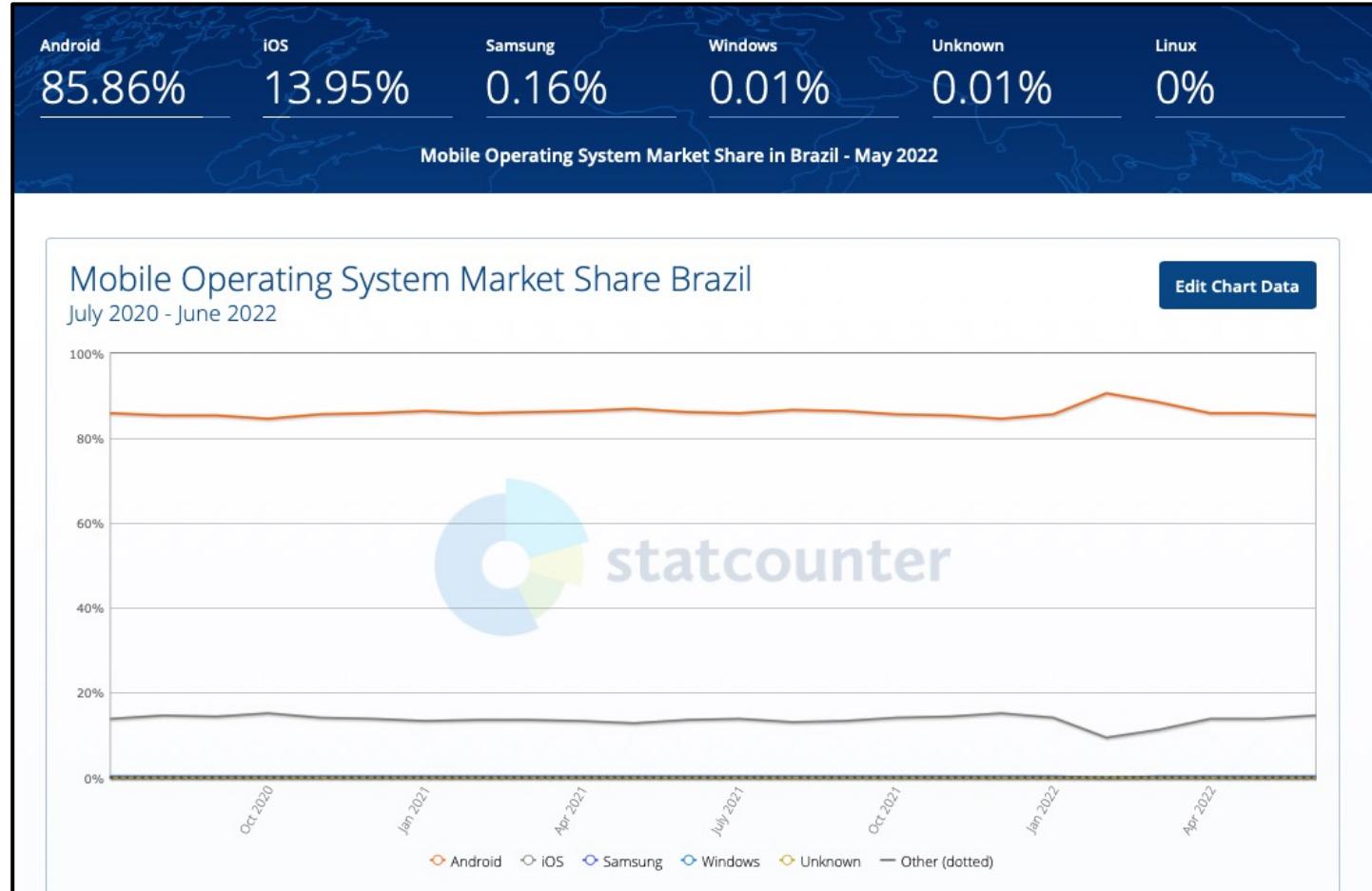
<30%

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

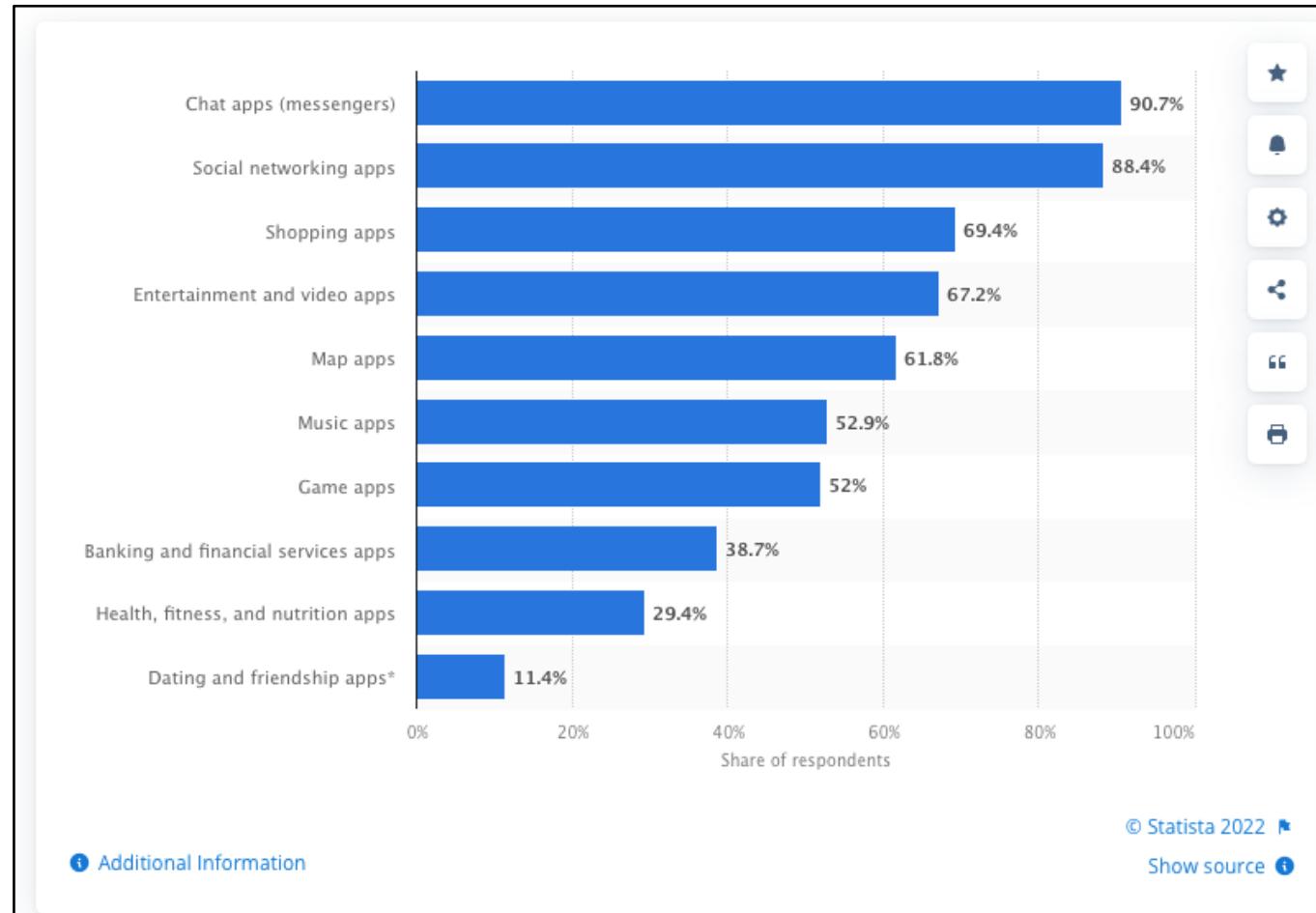
<15%

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

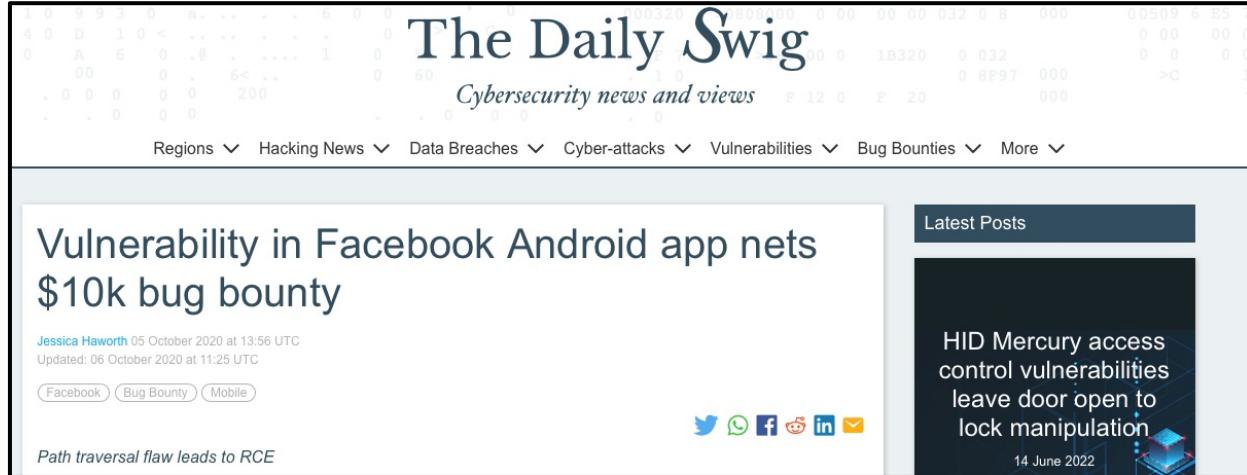
Facebook Bug Bounty Mobile

Path traversal flaw leads to RCE

Latest Posts

HID Mercury access control vulnerabilities leave door open to lock manipulation

14 June 2022



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

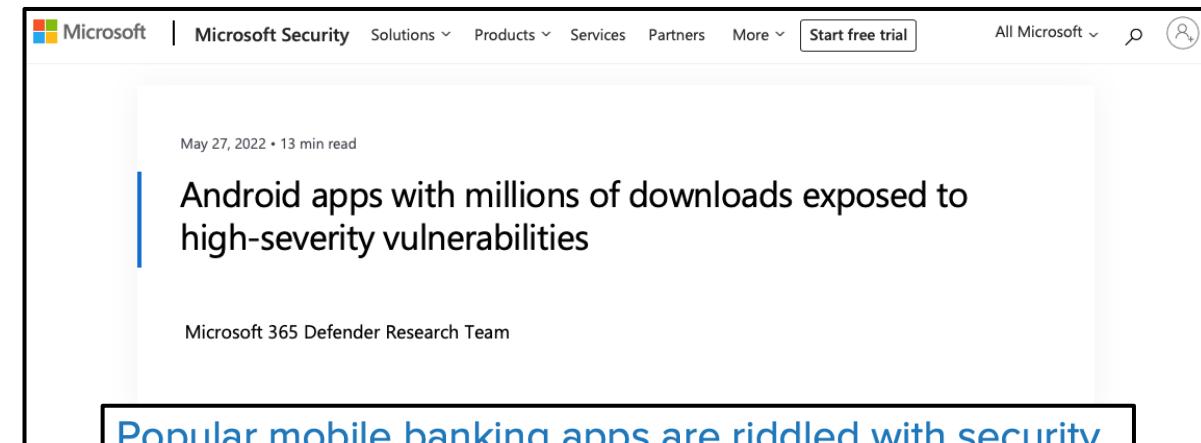
Microsoft 365 Defender Research Team

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

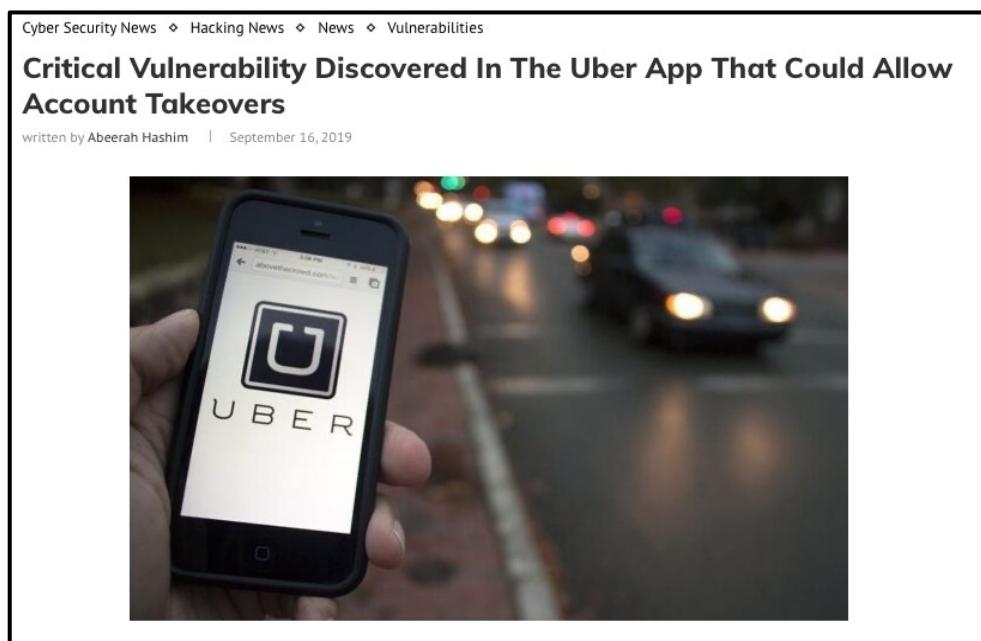
written by Abeerah Hashim | September 16, 2019



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

f t g



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.

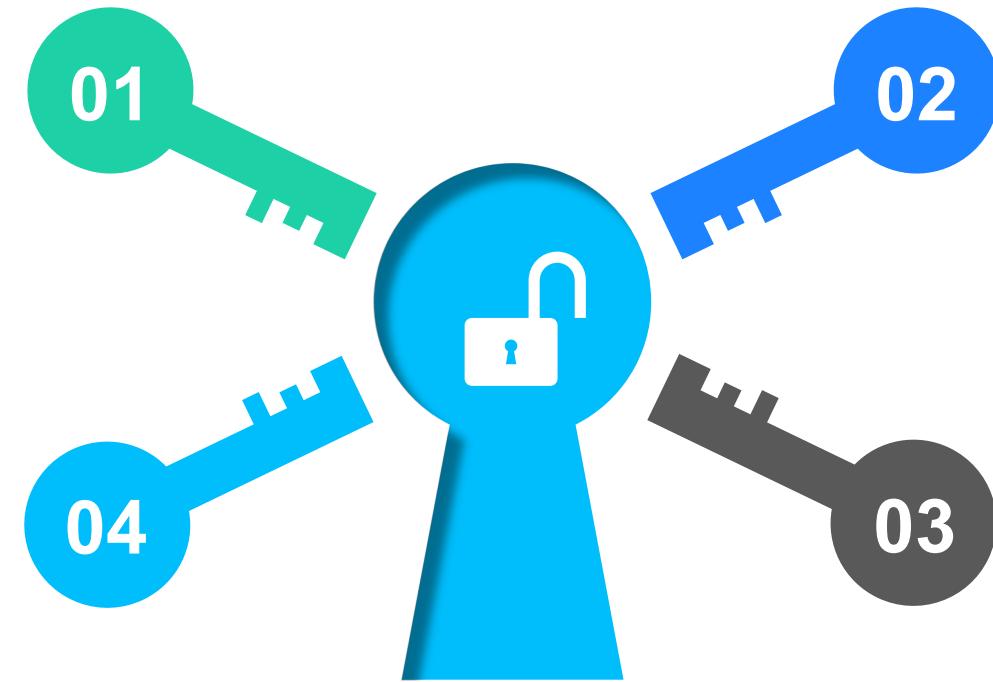
By: Tech Desk | New Delhi | Updated: November 25, 2019 5:49:16 pm

f t g

ADVERTISEMENT



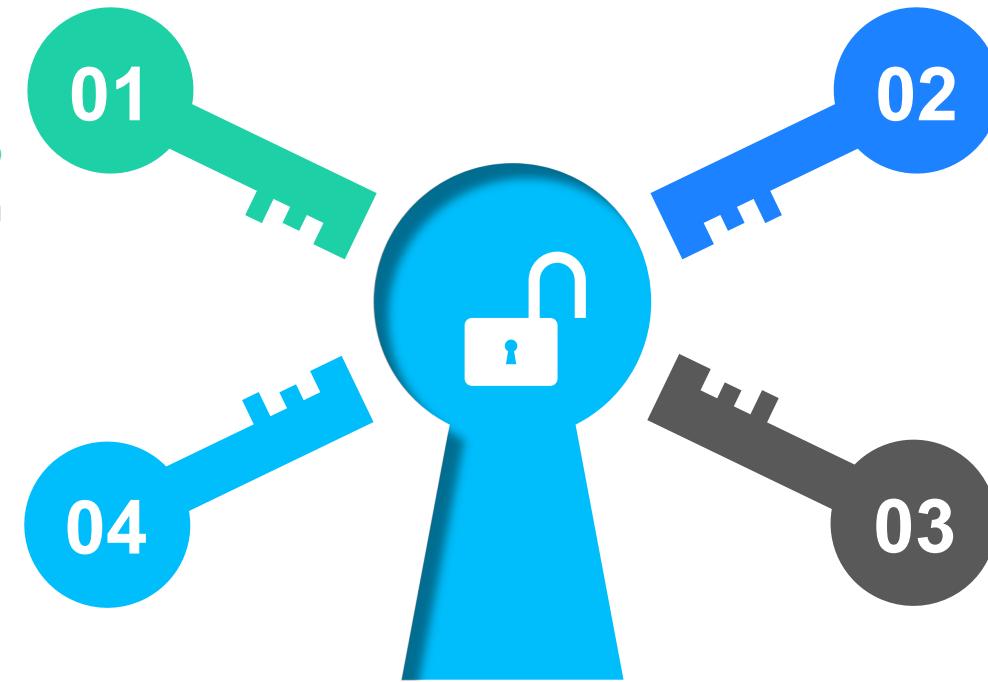
Por que vulnerabilidades ainda são constantes?



Por que vulnerabilidades ainda são constantes?

Falta de conhecimento

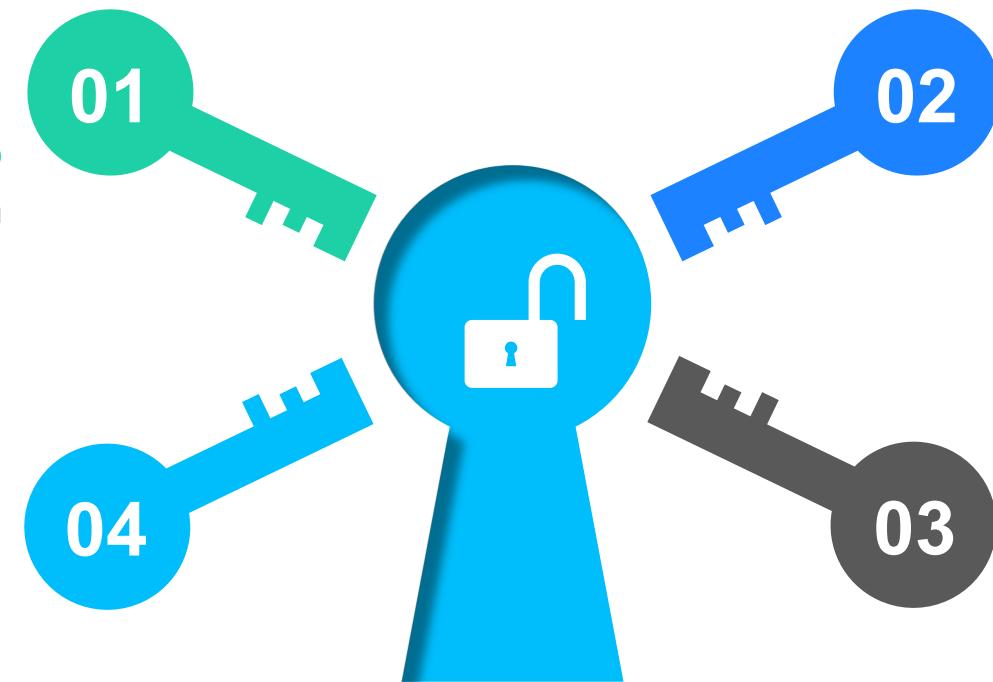
Desenvolvedores não têm conhecimento sobre segurança.



Por que vulnerabilidades ainda são constantes?

Falta de conhecimento

Desenvolvedores não têm conhecimento sobre segurança.



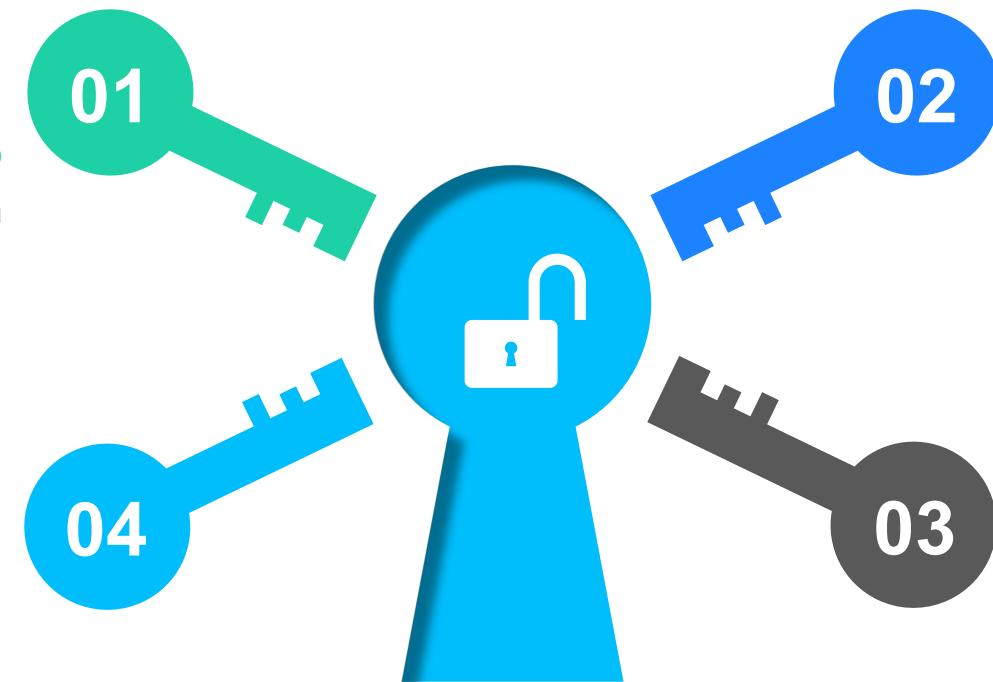
Segurança é complexo

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

Por que vulnerabilidades ainda são constantes?

Falta de conhecimento

Desenvolvedores não têm conhecimento sobre segurança.



Segurança é complexo

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

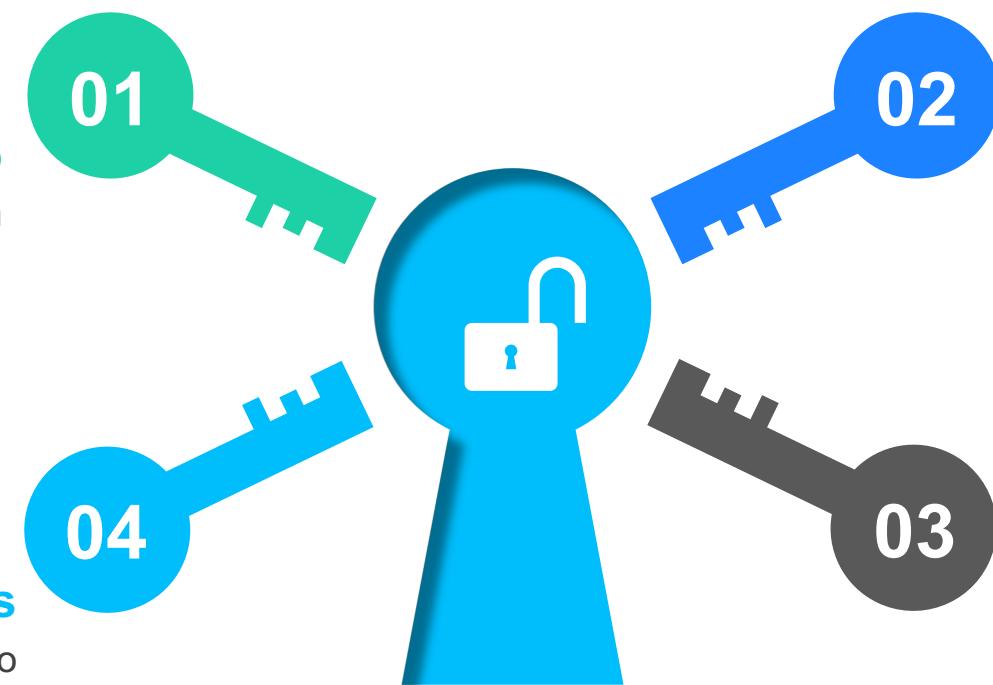
Segurança é negligenciada

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

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

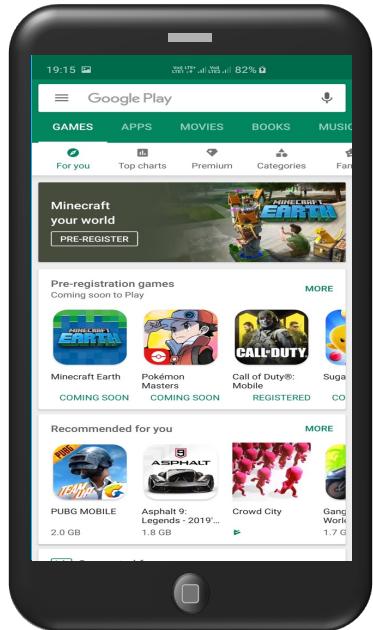
O teste de segurança é negligenciado com relação a outros tipos de testes



Android

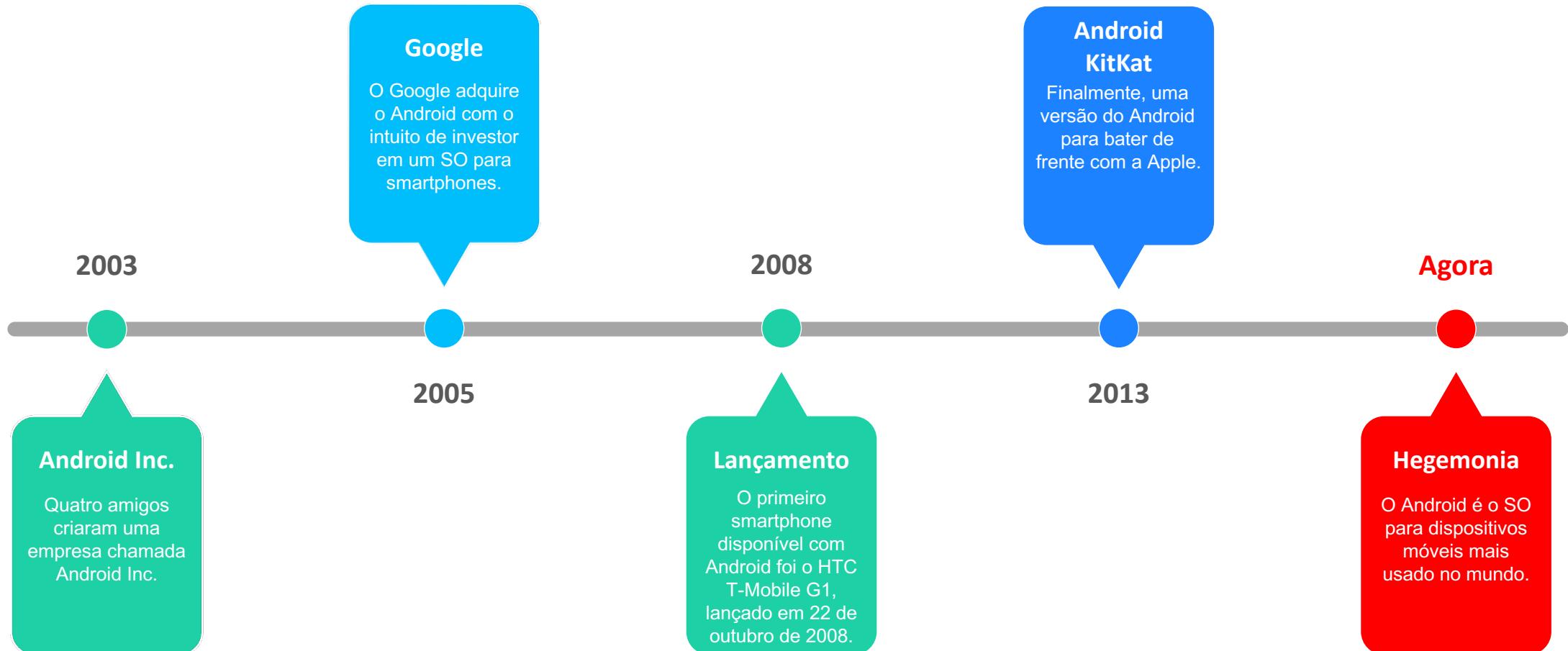
Conhecendo o Android, arquitetura e afins

Conhecendo o Android

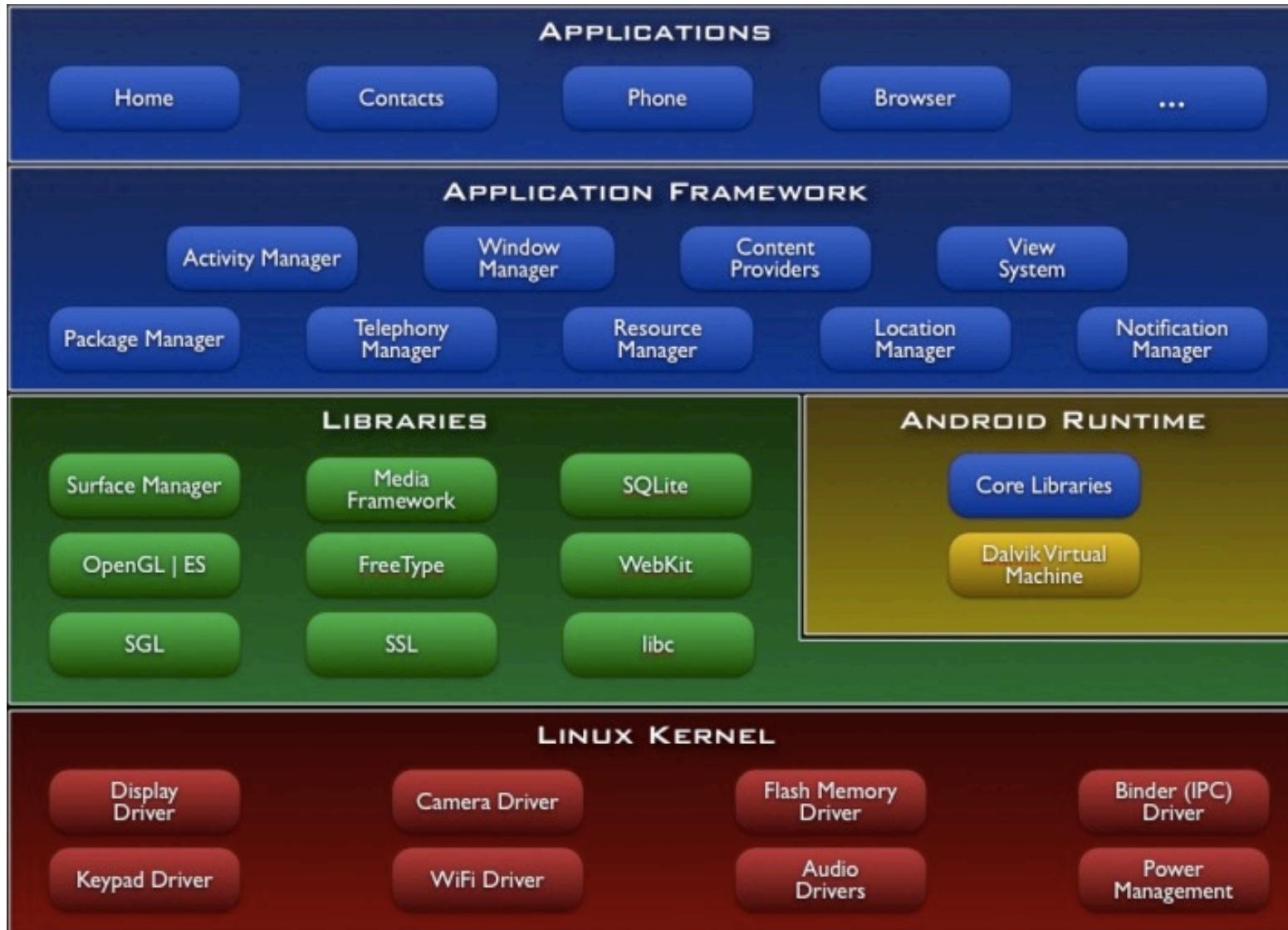


- Sistema Operacional mais utilizado em dispositivos móveis, tais como *smartphones* e *tablets*;
- Originalmente desenvolvido pela Android Inc em 2003 e adquirido pelo Google em 2005;
- Sistema Operacional aberto – *open-source* (<https://source.android.com>);
- Usa o Kernel Linux como base.

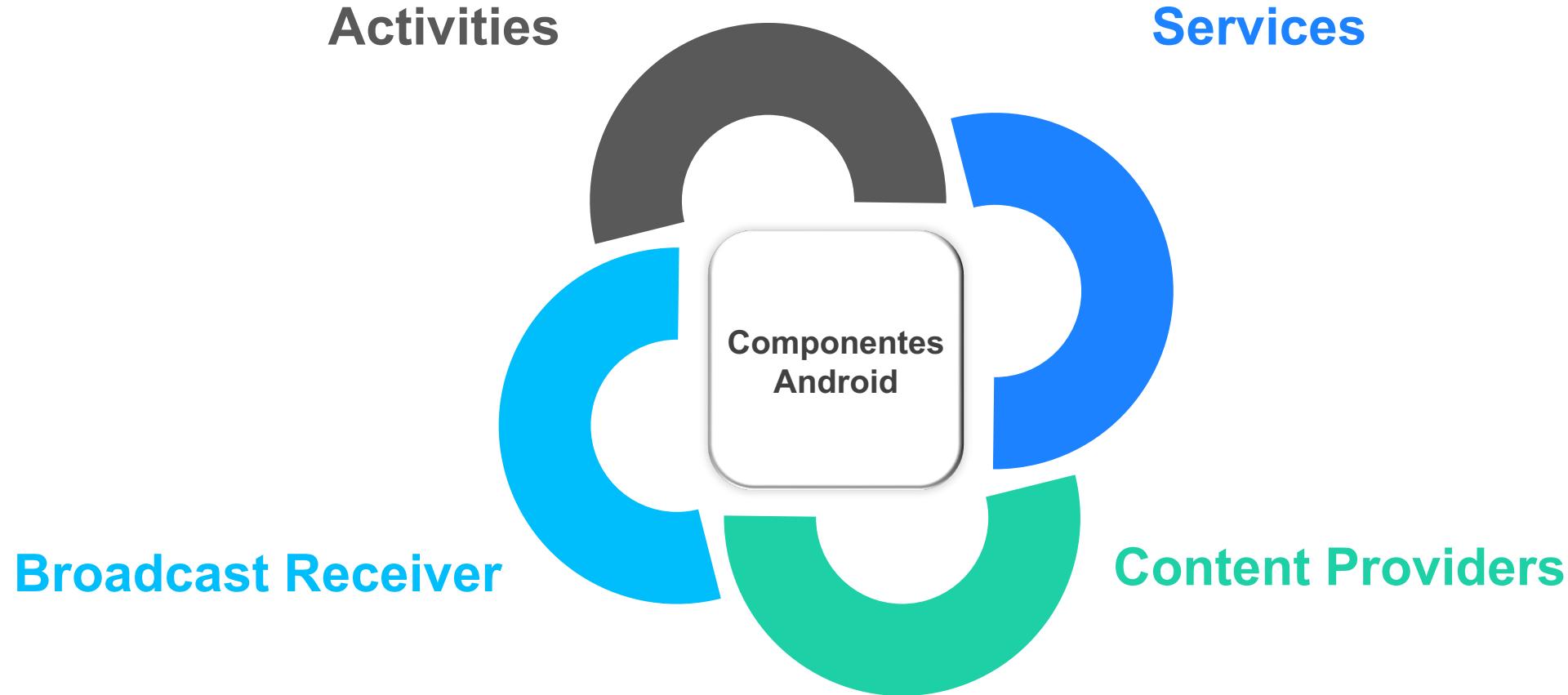
A História do Android



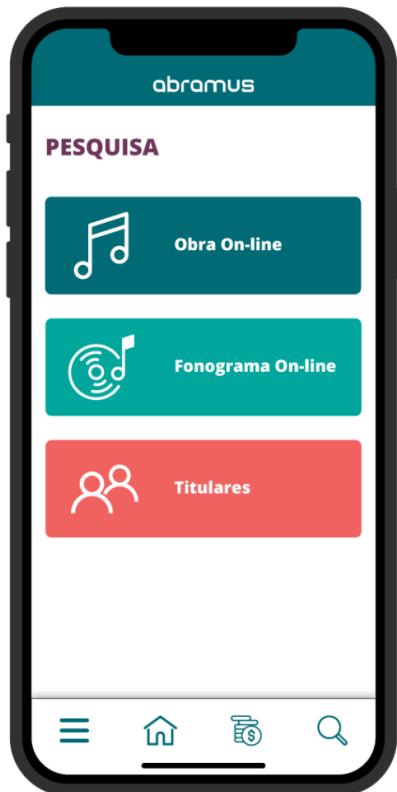
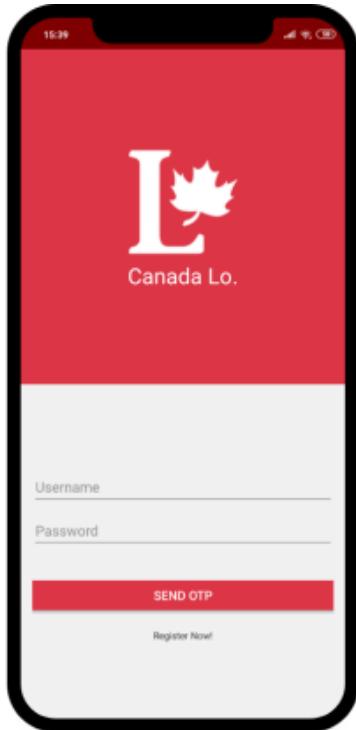
Arquitetura do Android



Principais componentes do Android



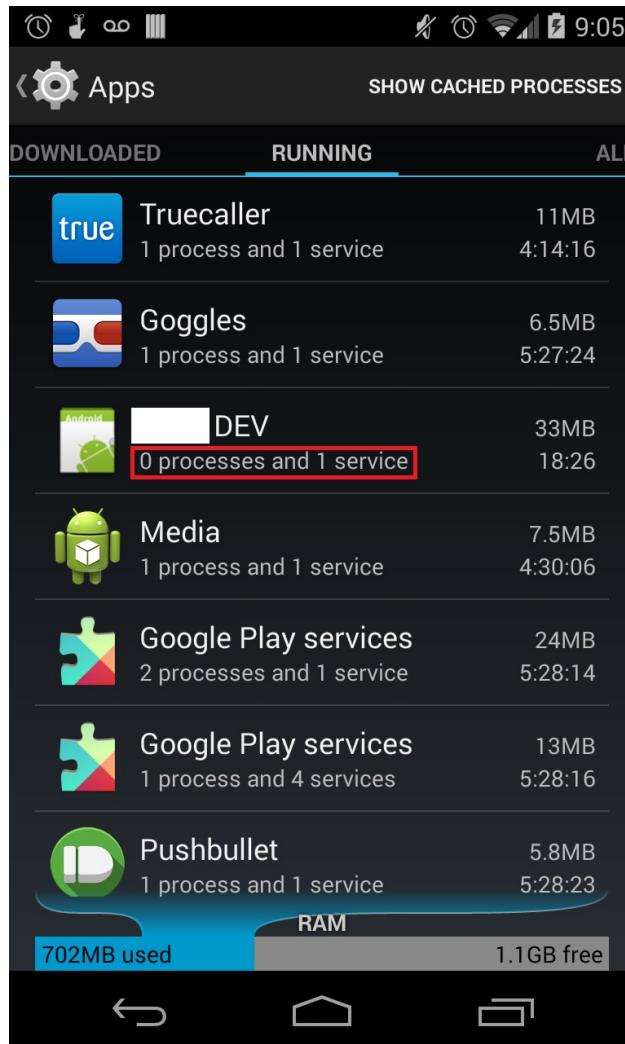
Activities



- 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*:

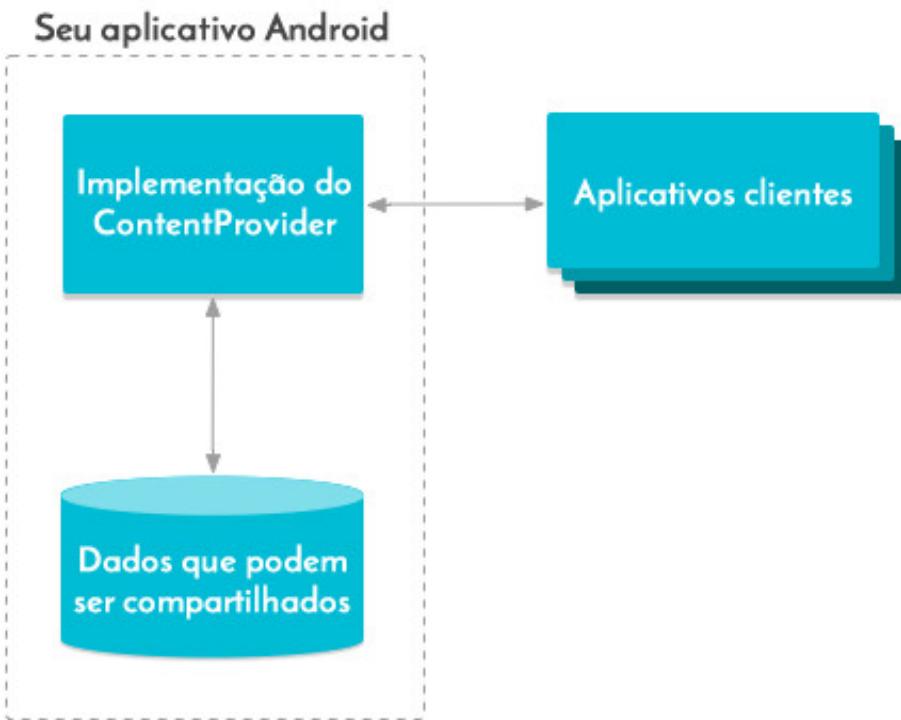
```
Intent intent = new Intent(this, SegundaActivity.class);
startActivity(intent);
```

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.

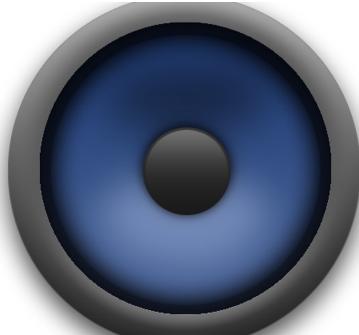
Broadcast Receiver



- 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

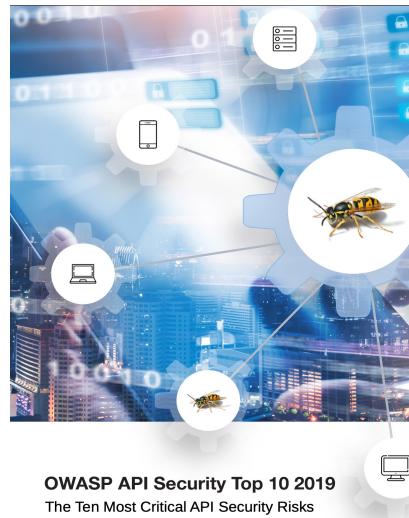
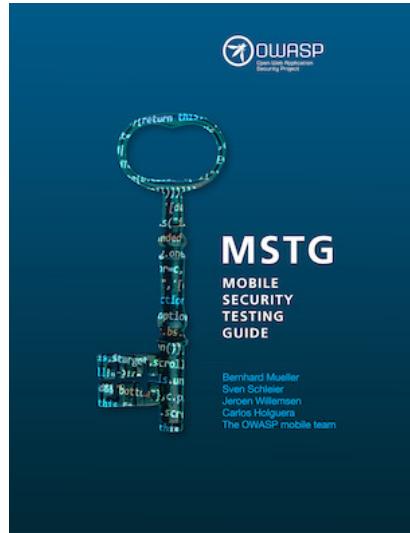


Problemas e riscos de segurança

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 e APIs;
- Disponibiliza gratuitamente artigos, metodologias, diretrizes, documentações, ferramentas e afins sobre segurança;
- Projetos Top 10 Mobile, Top 10 Web e Top 10 API.

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

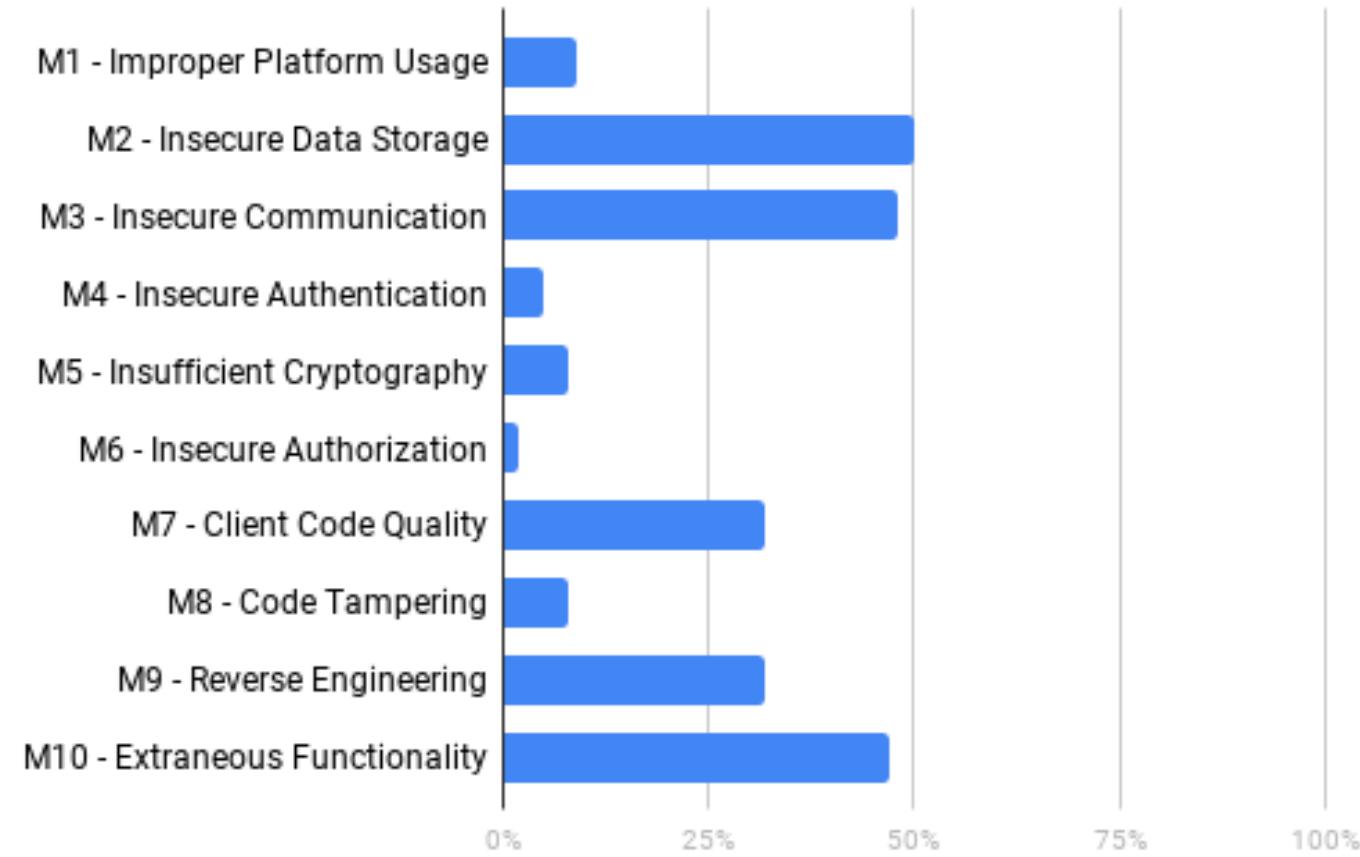


Top 10 Mobile

Comparison Between 2016-2023		
OWASP-2016	OWASP-2023-Initial Release	Comparison Between 2016-2023
M1: Improper Platform Usage	M1: Improper Credential Usage	New
M2: Insecure Data Storage	M2: Inadequate Supply Chain Security	New
M3: Insecure Communication	M3: Insecure Authentication / Authorization	Merged M4&M6 to M3
M4: Insecure Authentication	M4: Insufficient Input/Output Validation	New
M5: Insufficient Cryptography	M5: Insecure Communication	Moved from M3 to M5
M6: Insecure Authorization	M6: Inadequate Privacy Controls	New
M7: Client Code Quality	M7: Insufficient Binary Protections	Merged M8&M9 to M7
M8: Code Tampering	M8: Security Misconfiguration	Rewording [M10]
M9: Reverse Engineering	M9: Insecure Data Storage	Moved from M2 to M9
M10: Extraneous Functionality	M10: Insufficient Cryptography	Moved from M5 to M10

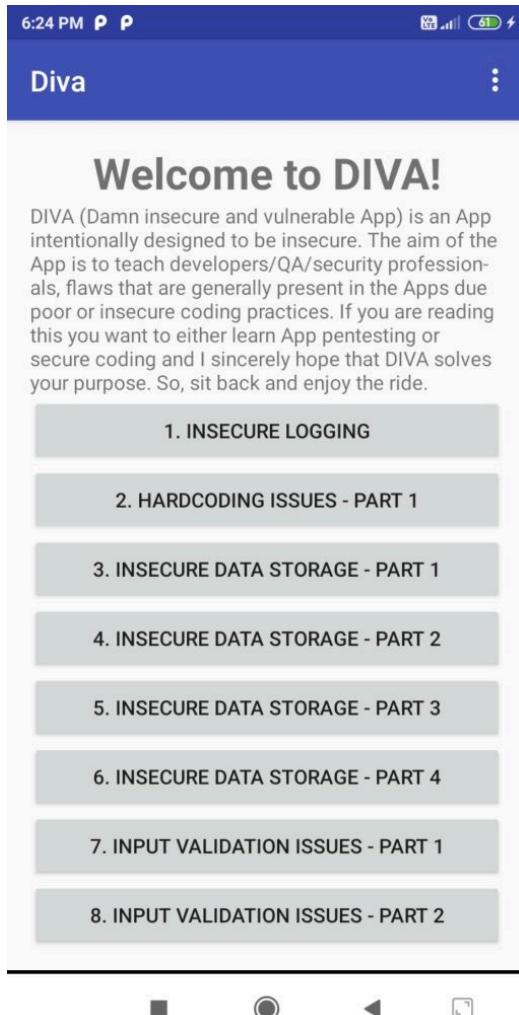
Top 10 Mobile

OWASP MOBILE TOP 10 VIOLATION RATES





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.

Ferramentas

JD-GUI



JD-GUI

eSoftner



APK tool

Android SDK
Manager



Genymotion



Dex2jar



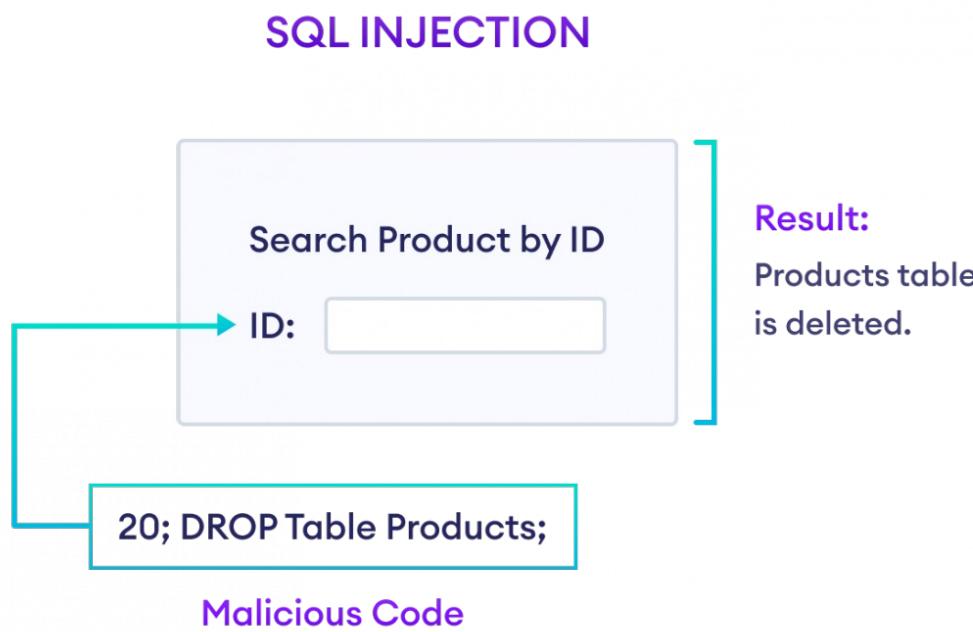
BurpSuite



Algumas Vulnerabilidades

- 1 – SQL Injection;
- 2 – Armazenamento Inseguro de Dados;
- 3 – Log Inseguro;
- 4 – Comunicação Insegura;
- 5 – Problemas de Controle de Acesso.

SQL Injection



1. Explora falhas relacionadas a validação de entrada de dados;
2. Permite o acesso não autorizado a dados confidenciais;
3. Possibilita a execução de comandos maliciosos no banco de dados, tais como editar, criar e deletar dados;
4. **Método de mitigação: sempre validar campos de entrada de dados do usuário.**

SQL Injection

SQLInjectionActivity.class - Java Decomplier

File Edit Navigation Search Help

diva.jar

InputValidation3Activity.class InsecureDataStorage1Activity.class SQLInjectionActivity.class

```
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

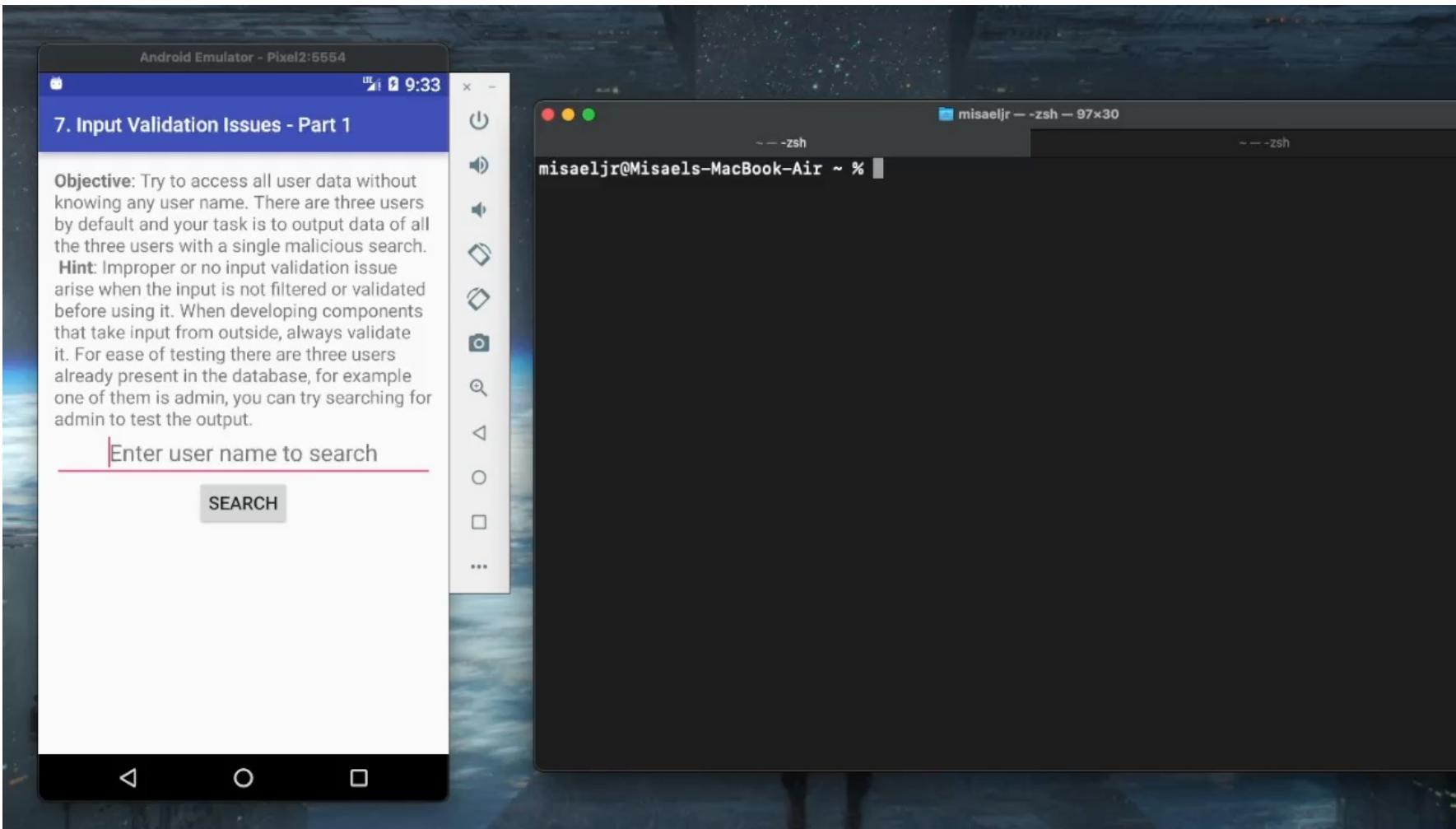
public class SQLInjectionActivity extends AppCompatActivity {
    private SQLiteOpenHelper mDB;

    protected void onCreate(Bundle paramBundle) {
        super.onCreate(paramBundle);
        try {
            this.mDB = openOrCreateDatabase("sqli", 0, null);
            this.mDB.execSQL("DROP TABLE IF EXISTS sqliuser");
            this.mDB.execSQL("CREATE TABLE IF NOT EXISTS sqliuser(user VARCHAR, password VARCHAR, credit_card VARCHAR);");
            this.mDB.execSQL("INSERT INTO sqliuser VALUES ('admin', 'passwd123', '1234567812345678');");
            this.mDB.execSQL("INSERT INTO sqliuser VALUES ('diva', 'p@ssword', '11112223334444');");
            this.mDB.execSQL("INSERT INTO sqliuser VALUES ('john', 'password123', '555566667778888');");
        } catch (Exception exception) {
            Log.d("Diva-sqli", "Error occurred while creating database for SQLI: " + exception.getMessage());
        }
        setContentView(2130968617);
    }

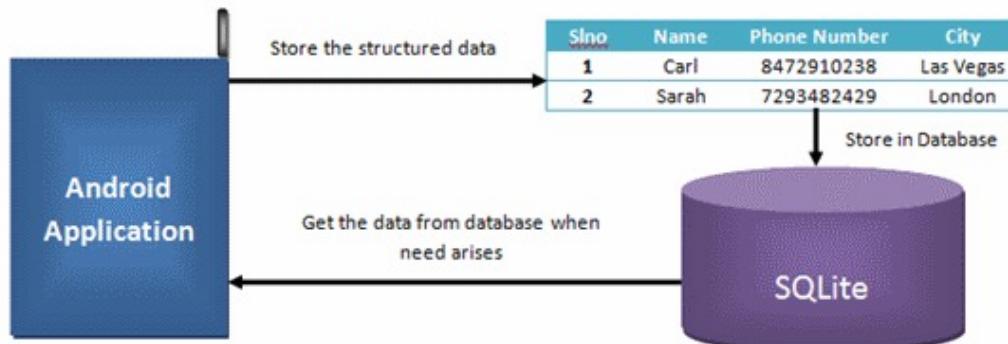
    public void search(View paramView) {
        EditText editText = (EditText) findViewById(2131493017);
        try {
            Cursor cursor = this.mDB.rawQuery("SELECT * FROM sqliuser WHERE user = '" + editText.getText().toString() + "'", null);
            StringBuilder stringBuilder = new StringBuilder("");
            if (cursor != null && cursor.getCount() > 0) {
                cursor.moveToFirst();
                do {
                    stringBuilder.append("User: (" + cursor.getString(0) + ") pass: (" + cursor.getString(1) + ") Credit card: (" + cursor.getString(2) + ")");
                } while (cursor.moveToNext());
            } else {
                stringBuilder.append("User: (" + editText.getText().toString() + ") not found");
            }
            Toast.makeText((Context)this, stringBuilder.toString(), 0).show();
            return;
        } catch (Exception exception) {
            Log.d("Diva-sqli", "Error occurred while searching in database: " + exception.getMessage());
            return;
        }
    }
}
```

The code shows a Java application structure with several activity classes. The main focus is on the `SQLInjectionActivity`. It contains logic to create a SQLite database named `sqli` and insert three test users: `admin`, `diva`, and `john`. The `search` method demonstrates a SQL injection vulnerability by concatenating user input directly into an `SQLiteDatabase.rawQuery` query without proper sanitization. A red box highlights the problematic code segment where the user input is appended to the query string.

SQL Injection

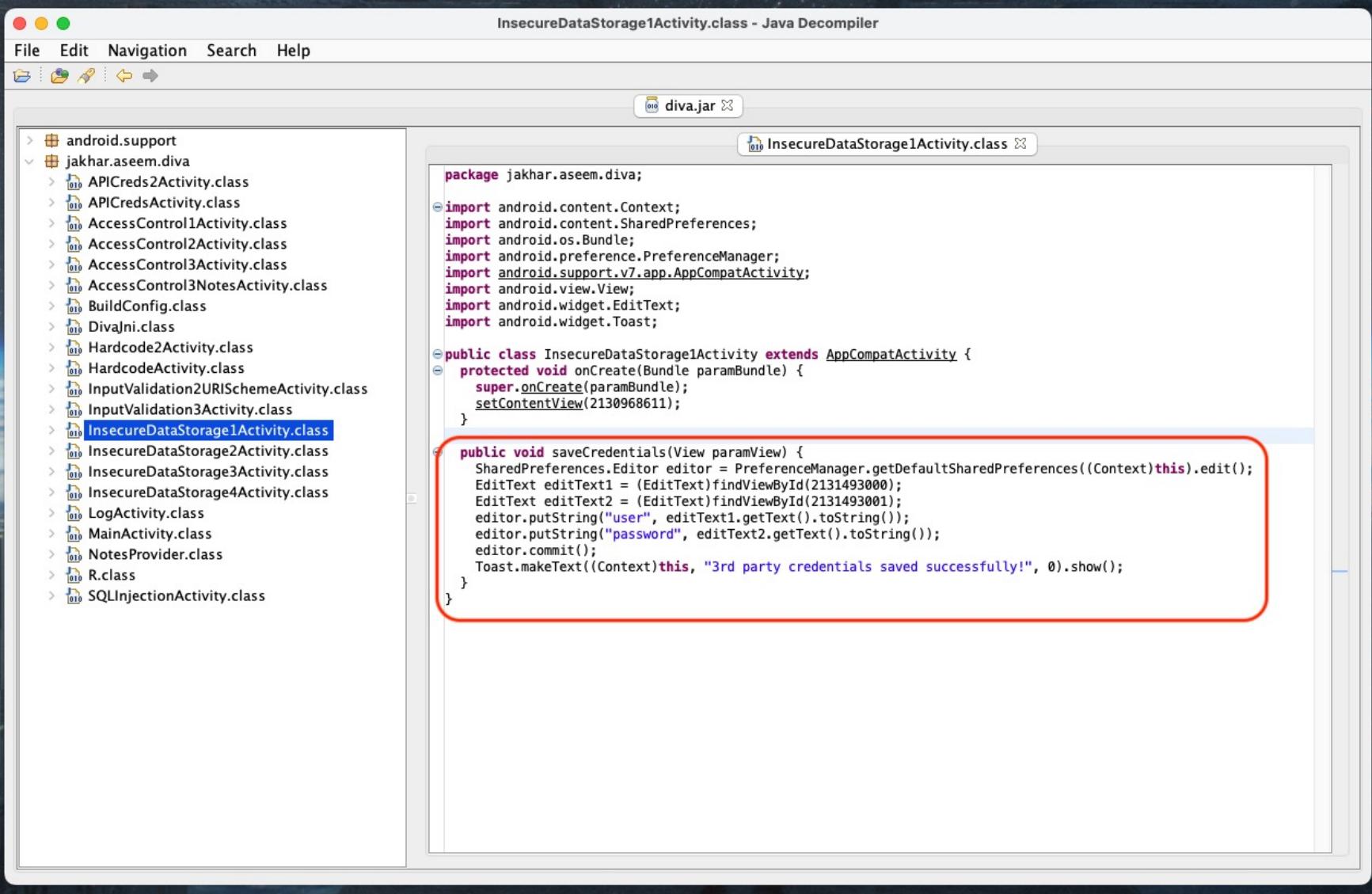


Armazenamento Inseguro de Dados



1. O armazenamento local de dados sensíveis sem criptografia torna possível tais dados serem explorados;
2. Permite o acesso indevido a informações sensíveis armazenadas no banco de dados;
3. Dados excluídos também podem ser acessados caso eles não sejam excluídos após a desinstalação do aplicativo;
4. **Método de mitigação: criptografar os dados armazenados localmente.**

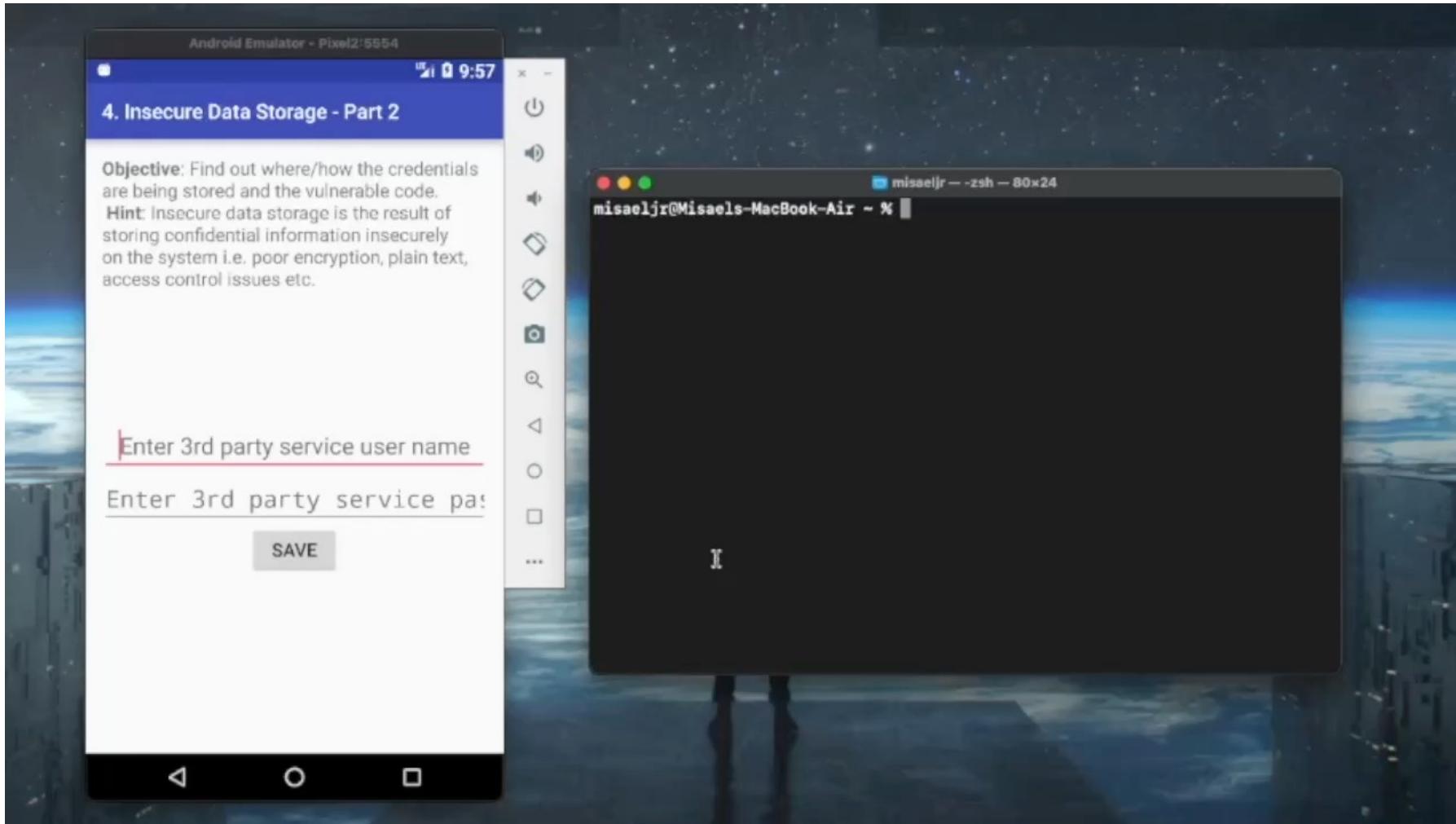
Armazenamento Inseguro de Dados



The screenshot shows the Java Decomiler interface with the title "InsecureDataStorage1Activity.class - Java Decomiler". The left pane displays a class hierarchy tree for the package "jakhar.aseem.diva", including classes like APIcreds2Activity.class, APIcredsActivity.class, AccessControl1Activity.class, etc., and the target class "InsecureDataStorage1Activity.class" which is highlighted with a blue selection bar. The right pane shows the decompiled Java code for "InsecureDataStorage1Activity.class". A red rectangular box highlights the following code segment:

```
public void saveCredentials(View paramView) {
    SharedPreferences.Editor editor = PreferenceManager.getDefaultSharedPreferences((Context)this).edit();
    EditText editText1 = (EditText)findViewById(2131493000);
    EditText editText2 = (EditText)findViewById(2131493001);
    editor.putString("user", editText1.getText().toString());
    editor.putString("password", editText2.getText().toString());
    editor.commit();
    Toast.makeText((Context)this, "3rd party credentials saved successfully!", 0).show();
}
```

Armazenamento Inseguro de Dados

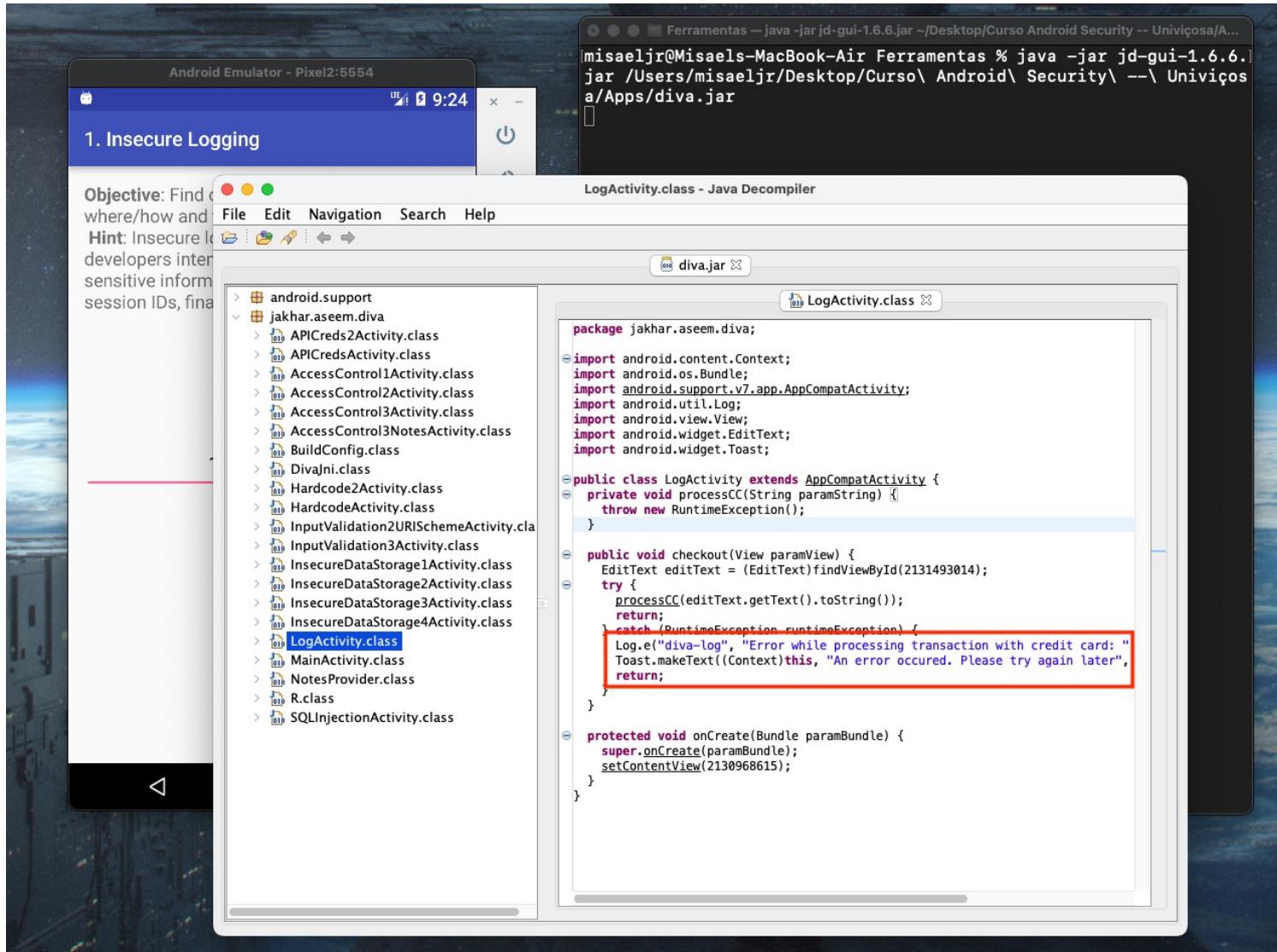


Log Inseguro

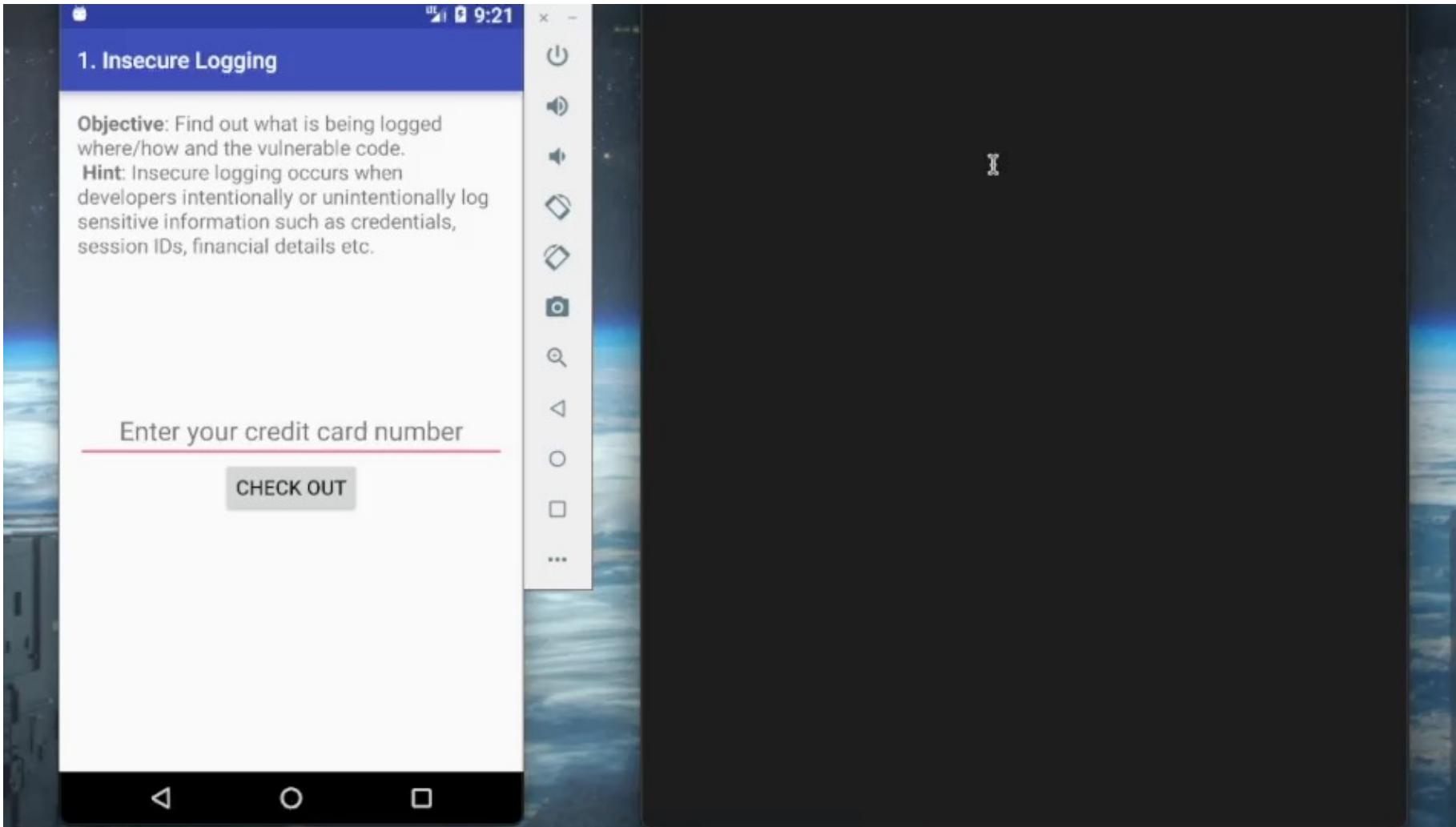
```
rewanth@rewanth-ThinkPad-T450s:~/Downloads/Dann-Vulnerable-Bank-Blog$ adb logcat | grep 5021
06-09 00:34:09.031 539 2014 I ActivityManager: Start proc 5021:com.app.dannvulnerablebank/u0a71 for activity com.app.dannvulnerablebank/.SplashScreen
06-09 00:45:58.840 5021 5021 I Thread-8: type=1400 audit(0.0:17462): avc: denied { read } for name="volley" dev="sdb3" ino=81873 scontext=u:r:untrusted_app:s0:c512,c7
08 tcontext=u:object_r:system_data_file:s0:c512,c768 tclass=dir permissive=1
06-09 00:45:58.840 5021 5021 I Thread-8: type=1400 audit(0.0:17463): avc: denied { open } for path="/data/data/com.app.dannvulnerablebank/cache/volley" dev="sdb3" ino
=81873 scontext=u:r:untrusted_app:s0:c512,c768 tcontext=u:object_r:system_data_file:s0:c512,c768 tclass=dir permissive=1
06-09 00:45:58.844 5021 5021 I Thread-8: type=1400 audit(0.0:17464): avc: denied { open } for path="/data/data/com.app.dannvulnerablebank/cache/volley/147482923313729
32246" dev="sdb3" ino=81774 scontext=u:r:untrusted_app:s0:c512,c768 tcontext=u:object_r:system_data_file:s0:c512,c768 tclass=file permissive=1
06-09 00:45:58.852 5021 5021 I nvulnerablebank: type=1400 audit(0.0:17465): avc: denied { write } for name="volley" dev="sdb3" ino=81873 scontext=u:r:untrusted_app:s0
:c512,c768 tcontext=u:object_r:system_data_file:s0:c512,c768 tclass=dir permissive=1
06-09 00:45:58.854 5021 5021 D Volley : [2] d.a: Cache cleared.
06-09 00:45:58.852 5021 5021 I nvulnerablebank: type=1400 audit(0.0:17466): avc: denied { remove_name } for name="14748292331372932246" dev="sdb3" ino=81774 scontext=
u:r:untrusted_app:s0:c512,c768 tcontext=u:object_r:system_data_file:s0:c512,c768 tclass=dir permissive=1
06-09 00:45:58.852 5021 5021 I nvulnerablebank: type=1400 audit(0.0:17467): avc: denied { unlink } for name="14748292331372932246" dev="sdb3" ino=81774 scontext=u:r:u
ntrusted_app:s0:c512,c768 tcontext=u:object_r:system_data_file:s0:c512,c768 tclass=dir permissive=1
06-09 00:45:58.880 5021 5021 I Thread-9: type=1400 audit(0.0:17468): avc: denied { add_name } for name="-2134775796-1733553926" scontext=u:r:untrusted_app:s0:c512,c7
8 tcontext=u:object_r:system_data_file:s0:c512,c768 tclass=dir permissive=1
06-09 00:45:58.880 5021 5021 I Thread-9: type=1400 audit(0.0:17469): avc: denied { create } for name="-2134775796-1733553926" scontext=u:r:untrusted_app:s0:c512,c768
tcontext=u:object_r:system_data_file:s0:c512,c768 tclass=file permissive=1
06-09 00:45:58.880 5021 5021 I Thread-9: type=1400 audit(0.0:17470): avc: denied { write } for path="/data/data/com.app.dannvulnerablebank/cache/volley/-2134775796-17
33553926" dev="sdb3" ino=81774 scontext=u:r:untrusted_app:s0:c512,c768 tcontext=u:object_r:system_data_file:s0:c512,c768 tclass=file permissive=1
06-09 00:46:07.797 5021 5021 D Volley : [2] d.a: Cache cleared.
06-09 00:46:07.827 5021 5021 E RecyclerView: No adapter attached; skipping layout
06-09 00:46:12.811 5021 5021 D Volley : [2] d.a: Cache cleared.
06-09 00:46:18.247 5021 5021 D Volley : [2] d.a: Cache cleared.
06-09 00:46:18.276 5021 5021 E RecyclerView: No adapter attached; skipping layout
06-09 00:46:47:89.335 5021 5021 D Volley : [2] d.a: Cache cleared.
06-09 01:08:23.531 5021 5021 D FRIDA CHECK: Frida Server NOT RUNNING
06-09 01:08:23.584 5021 5021 I queued-work-loo: type=1400 audit(0.0:30600): avc: denied { rename } for name="jwt.xml" dev="sdb3" ino=82181 scontext=u:r:untrusted_app:
s0:c512,c768 tcontext=u:object_r:system_data_file:s0:c512,c768 tclass=file permissive=1
06-09 01:08:23.588 5021 5021 I queued-work-loo: type=1400 audit(0.0:30601): avc: denied { setattr } for name="jwt.xml" dev="sdb3" ino=81780 scontext=u:r:untrusted_app:
s0:c512,c768 tcontext=u:object_r:system_data_file:s0:c512,c768 tclass=file permissive=1
06-09 01:08:26.865 5021 5021 I zygote : Deoptimizing android.graphics.Typeface a.a.a.a.a.V(android.content.Context, int, android.util.TypedValue, int, b.t.e.b.g, and
rotd.os.Handler, boolean) due to JIT same target
06-09 01:08:26.878 5021 5021 I zygote : Do full code cache collection, code=121KB, data=95KB
06-09 01:08:26.879 5021 5021 I zygote : After code cache collection, code=92KB, data=62KB
06-09 01:08:29.215 5021 5021 I Choreographer: Skipped 31 frames! The application may be doing too much work on its main thread.
06-09 01:08:34.724 5021 5021 I zygote : Do partial code cache collection, code=116KB, data=81KB
06-09 01:08:34.724 5021 5021 I zygote : After code cache collection, code=115KB, data=81KB
06-09 01:08:34.724 5021 5021 I zygote : Increasing code cache capacity to 512KB
06-09 01:08:44.653 5021 5021 D accessstoken: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlci2VybmtZSI6InVzZXIxIiwiaXNfYWRtaW4iOmZhbnLLCjpyXQloE2MjMzMzU4NDV9.lHjB
BagQ2OYR
EBBD10LXUNLNn1D6nI4KMXgFlfxRNU
06-09 01:08:46.553 5021 5021 I Choreographer: Skipped 39 frames! The application may be doing too much work on its main thread.
```

1. Dados sensíveis são gravados em logs do sistema que podem ser acessados por usuários não autorizados;
2. Os logs podem ser armazenados em *smartphones* ou em servidores de aplicativos;
3. A exibição dos dados sensíveis em texto claro no *logcat*, torna a exploração da vulnerabilidade mais simples;
4. Método de mitigação: não incluir dados sensíveis em logs da aplicação.

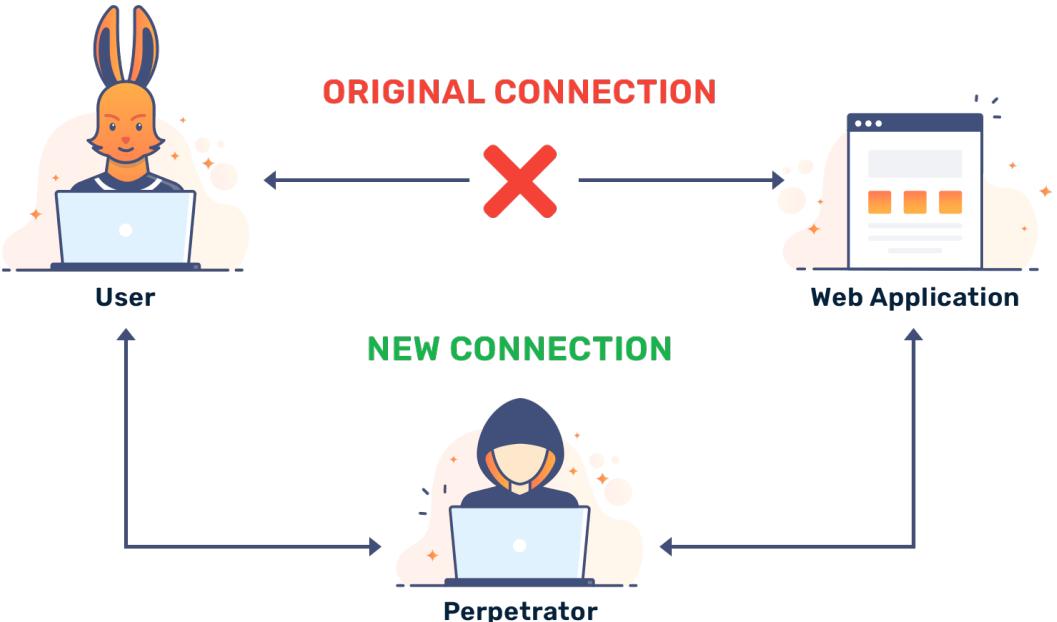
Log Inseguro



Log Inseguro

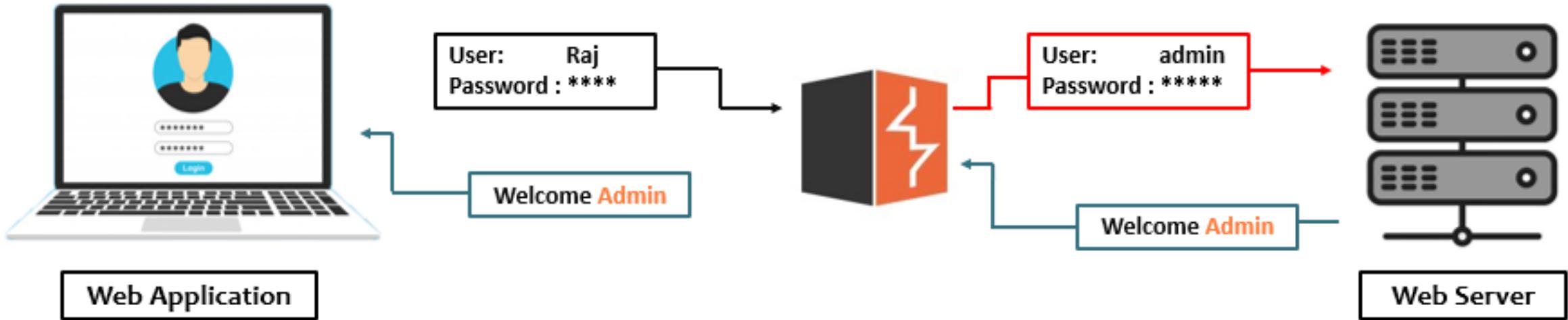


Comunicação Insegura

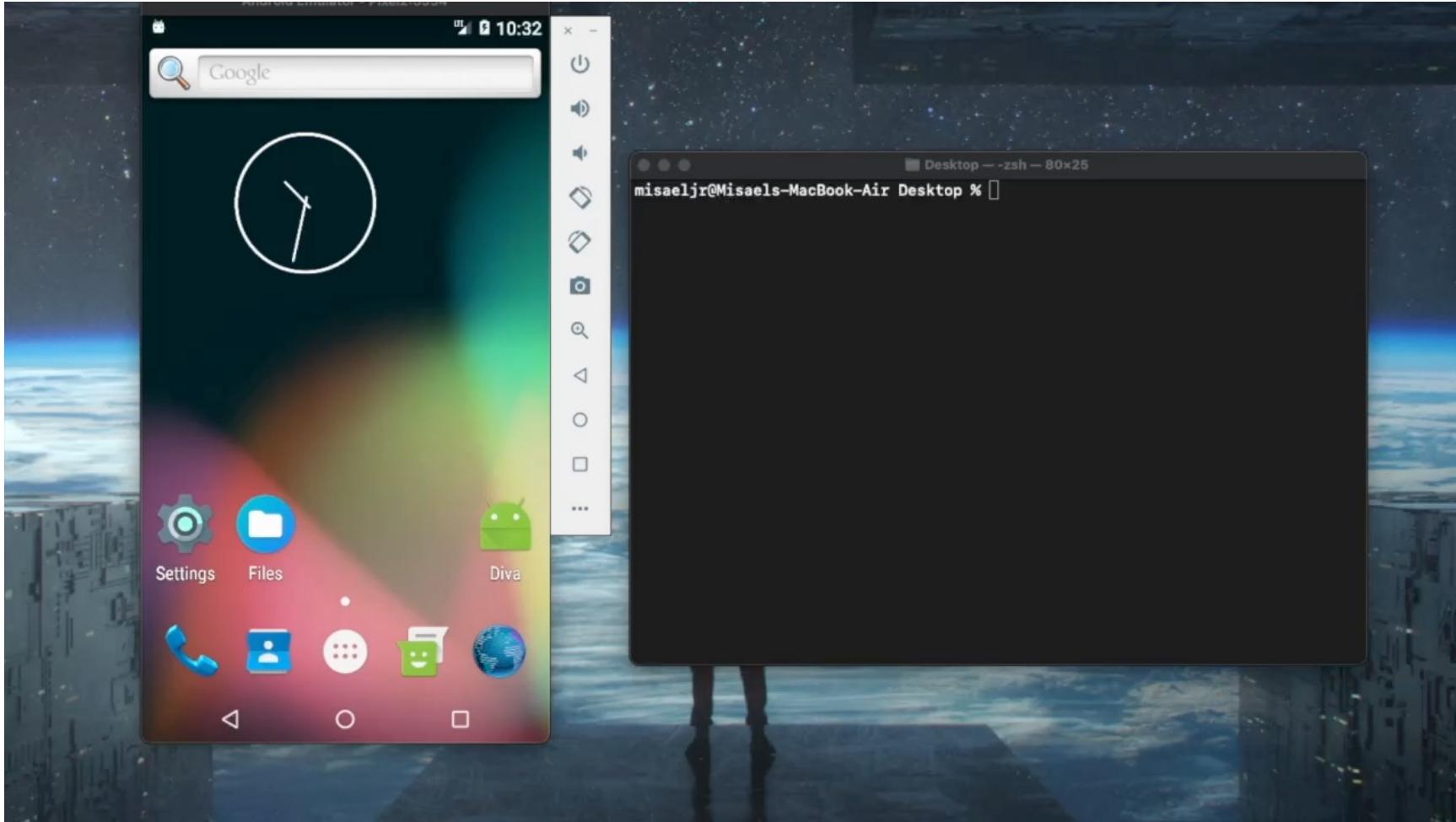


1. Vulnerabilidades associadas com cenários em que ocorre comunicação entre a aplicação e um serviço back-end;
2. Tais vulnerabilidades ocorrem quando não há proteção no tráfego de rede.
3. Motivo comum: falta de criptografia.
4. **Método de mitigação:** assegurar que os dados trafegados entre a aplicação e o servidor *back-end* estejam criptografados.

Comunicação Insegura



Comunicação Insegura

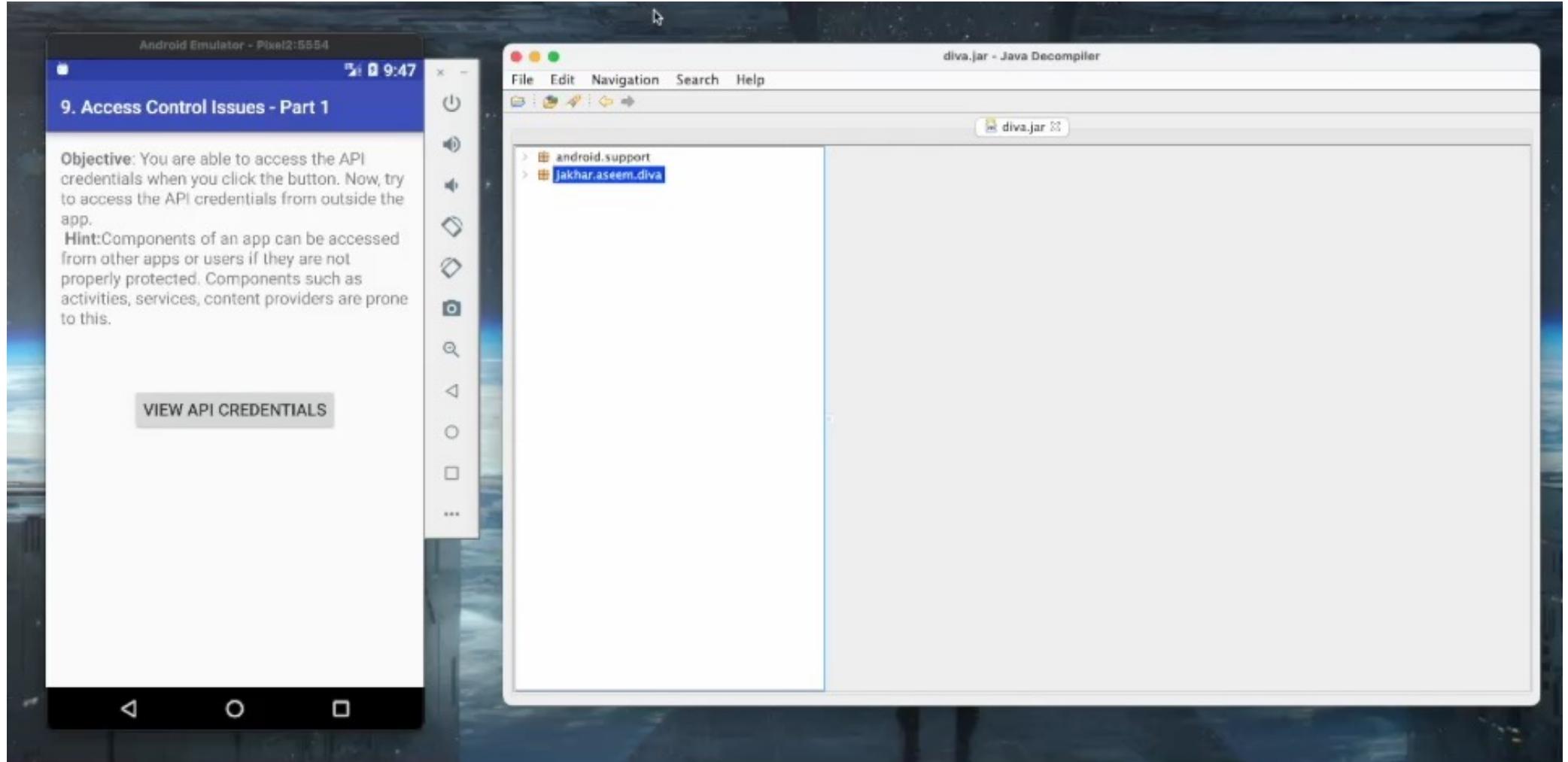


Problemas de Controle de Acesso

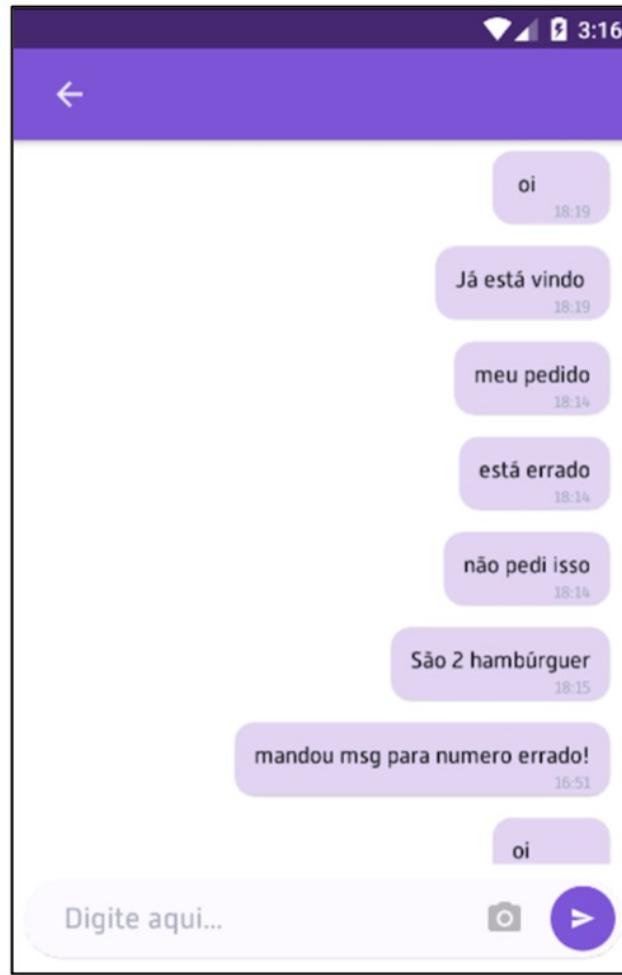


1. Essas vulnerabilidades ocorrem em casos em que a aplicação fornece funcionalidades específicas para usuários com autorização (Login -> Home);
2. Desse modo, o aplicativo deve estar corretamente configurado para não permitir acesso a usuários que não tenham real autorização;
3. **Método de mitigação:** assegurar que a tela/feature não pode ser acessada externamente e por usuários não autorizados.

Problemas de Controle de Acesso



Problemas de Controle de Acesso



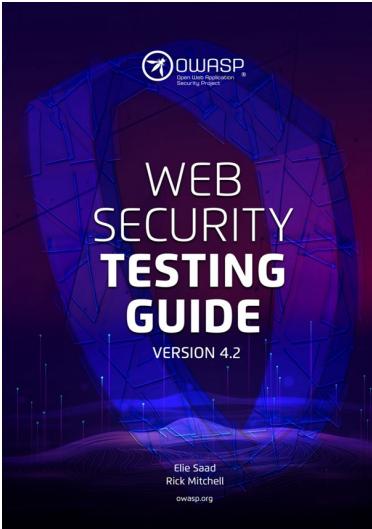
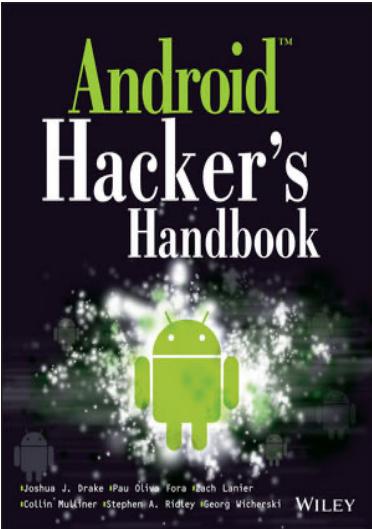
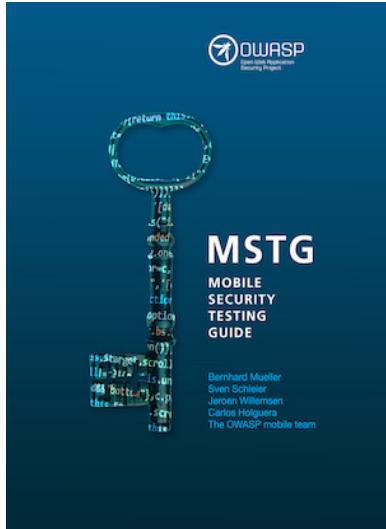


Indicações

Leituras, ferramentas, cursos e afins

Livros e Ferramentas

Livros



Ferramentas



Sites, Cursos e afins

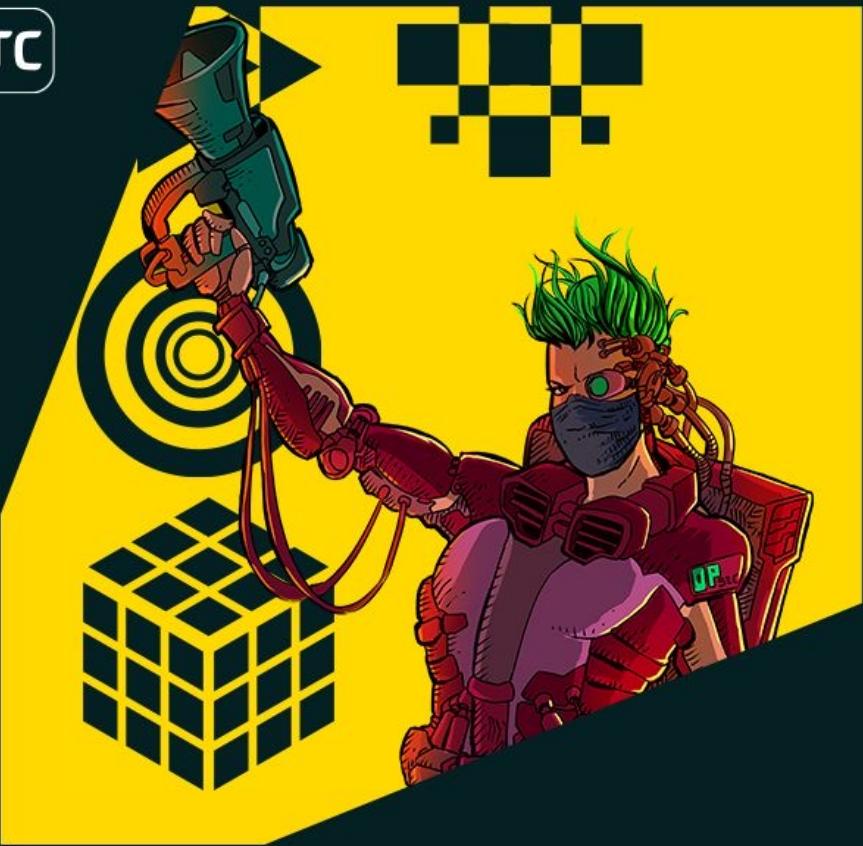
Sites



Cursos/Treinamentos



Roadsec

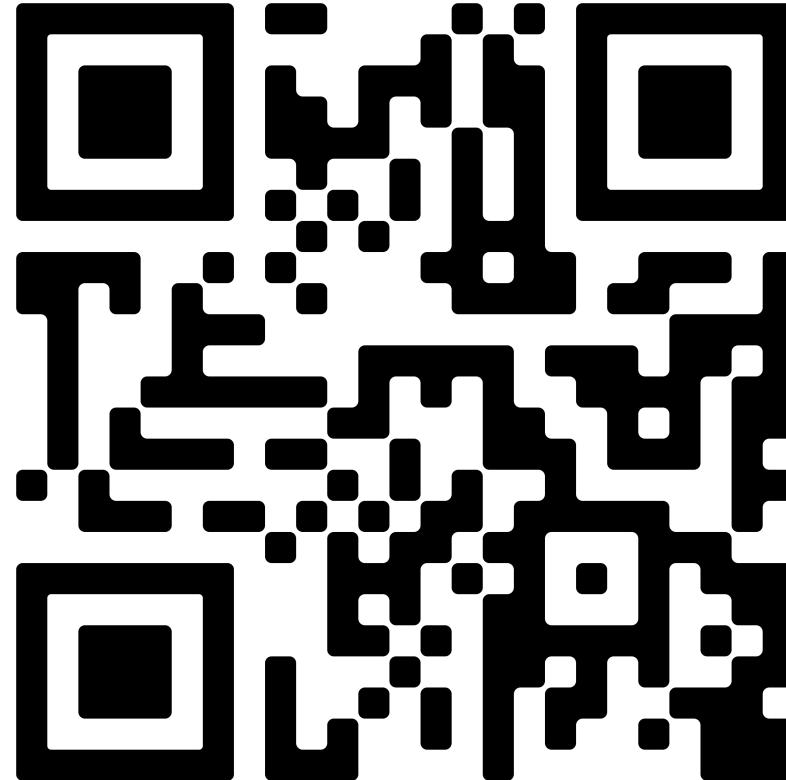


O TC investe em Tech

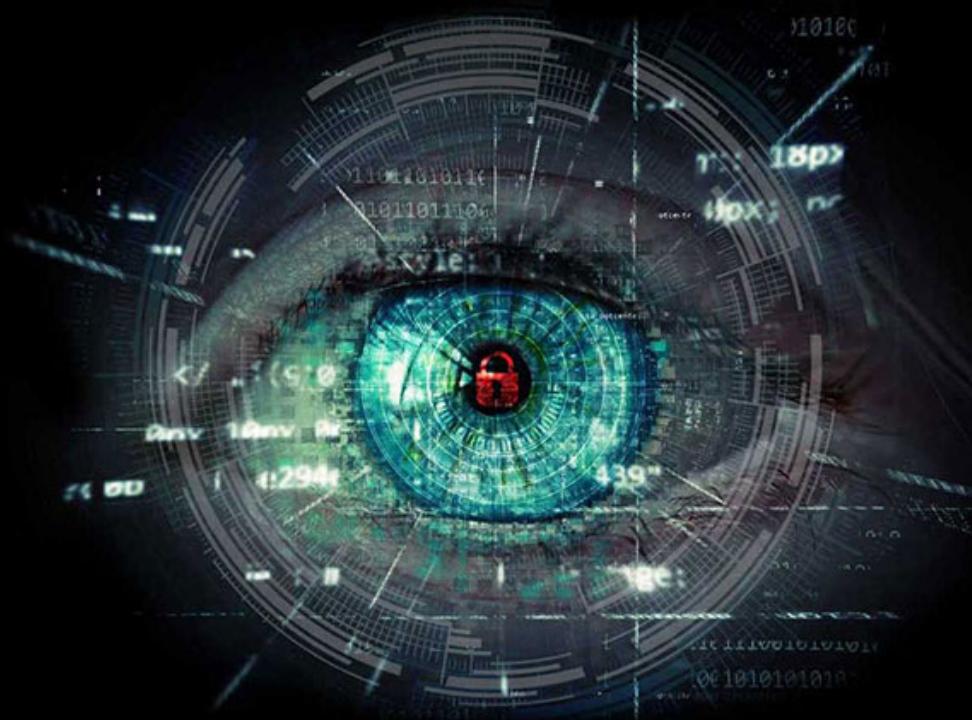
Somos patrocinadores oficiais do

ROADSEC
2022 黑客文化
ROADSEC®

Material da palestra



```
$ git clone https://github.com/misaelljr/Palestra__AndroidSec__Estacio.git
```



Muito obrigado!

Dúvidas?