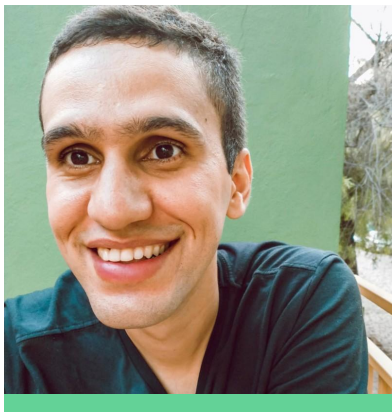


Introdução à Segurança de Aplicações Android

Eduardo Vasconcelos

*Arquiteto de Segurança de Aplicações MS @ SiDi
Egresso da EngComp (turma de 2017)*

\$ echo “\$USER”



vasconcedu



vasconcedu



vasconcedu.github.io

Eduardo Vasconcelos

Experiência

Arquiteto de Segurança de Aplicações MS @ **SiDi** (current)
Engenheiro de Segurança de Aplicações Móveis SR @ **iFood**
Analista de Segurança de Software SR @ **SiDi**
Analista de Segurança da Informação PL @ **Hacker Rangers**
Estagiário de TI @ **Embraer**
Estagiário de Engenharia @ **Sigma**

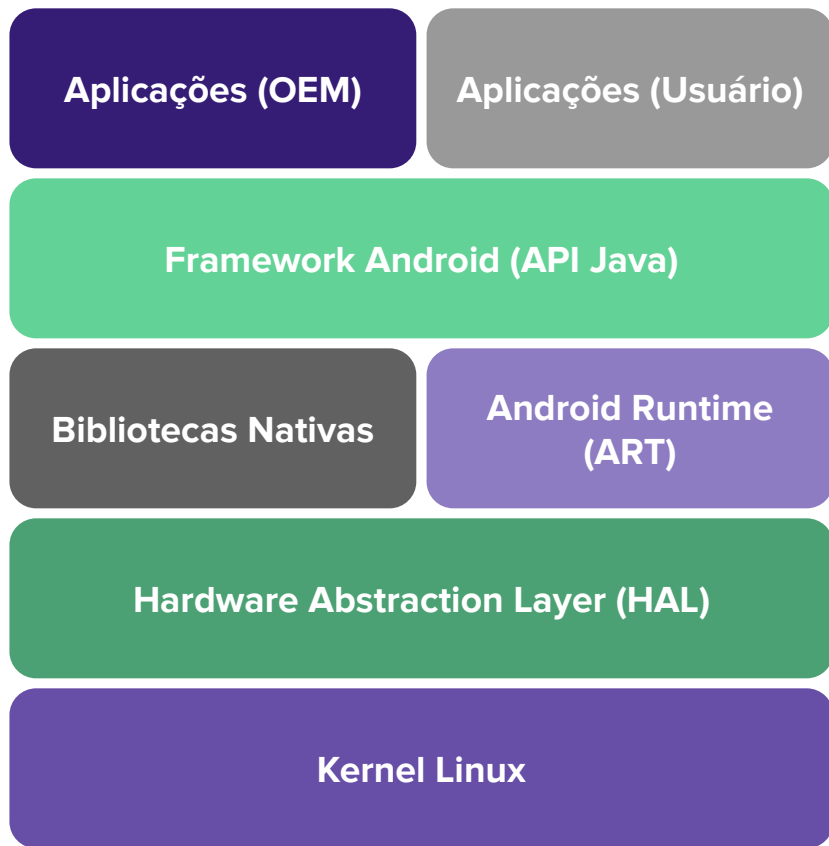
Formação

Mestrado em Ciências de Computação @ **USP** (ongoing)
Especialização em Eng. de Software @ **Unicamp**
Eng. de Computação @ **USP**
Eng. Eletrônica (visitante) @ **Trinity College Dublin**

Agenda

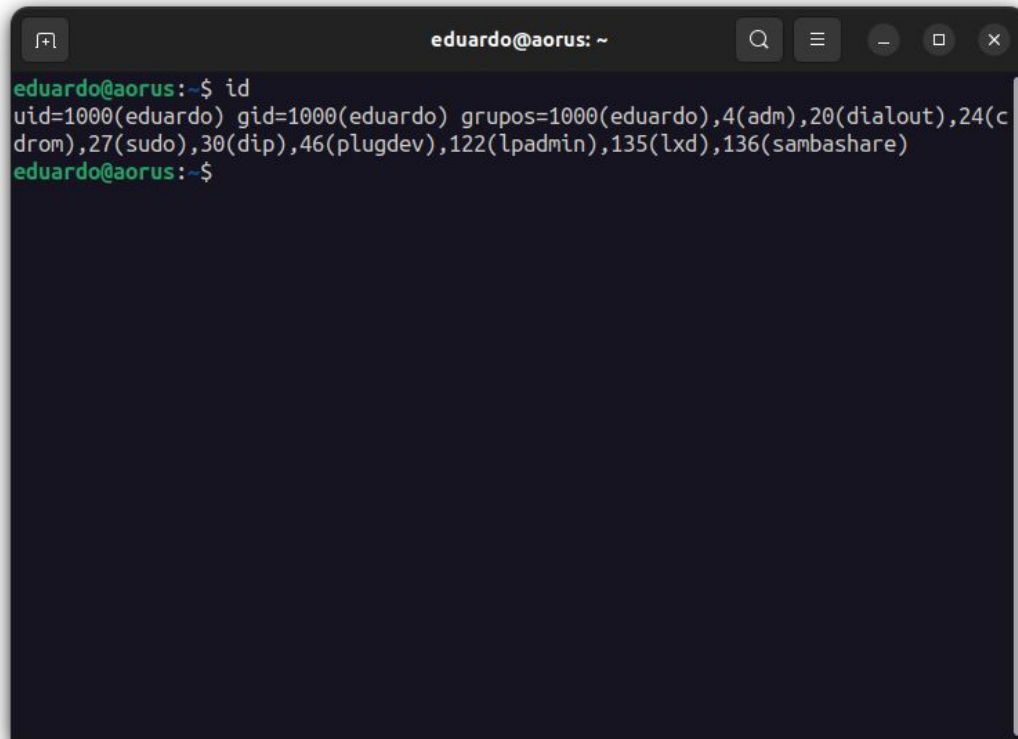
- **Parte 1. *Pré-security***
 - A Arquitetura do Android
 - *Sandboxing* de aplicações
 - Outros mecanismos de segurança
 - Anatomia básica de um app Android
- **Parte 2. *Security***
 - Vulnerabilidades em Android
 - Task Hijacking
 - Engenharia reversa e RASP
- **Sobre o SiDi**
- **Bônus: sobre trabalhar com segurança**

Parte 1. Pré-*security*



A Arquitetura do Android

Sandbox (I)



```
eduardo@aorus: ~  
eduardo@aorus:~$ id  
uid=1000(eduardo) gid=1000(eduardo) grupos=1000(eduardo),4(adm),20(dialout),24(c  
drom),27(sudo),30(dip),46(plugdev),122(lpadmin),135(lxd),136(sambashare)  
eduardo@aorus:~$
```

Saída do comando
`id` no Linux.

Sandbox (II)

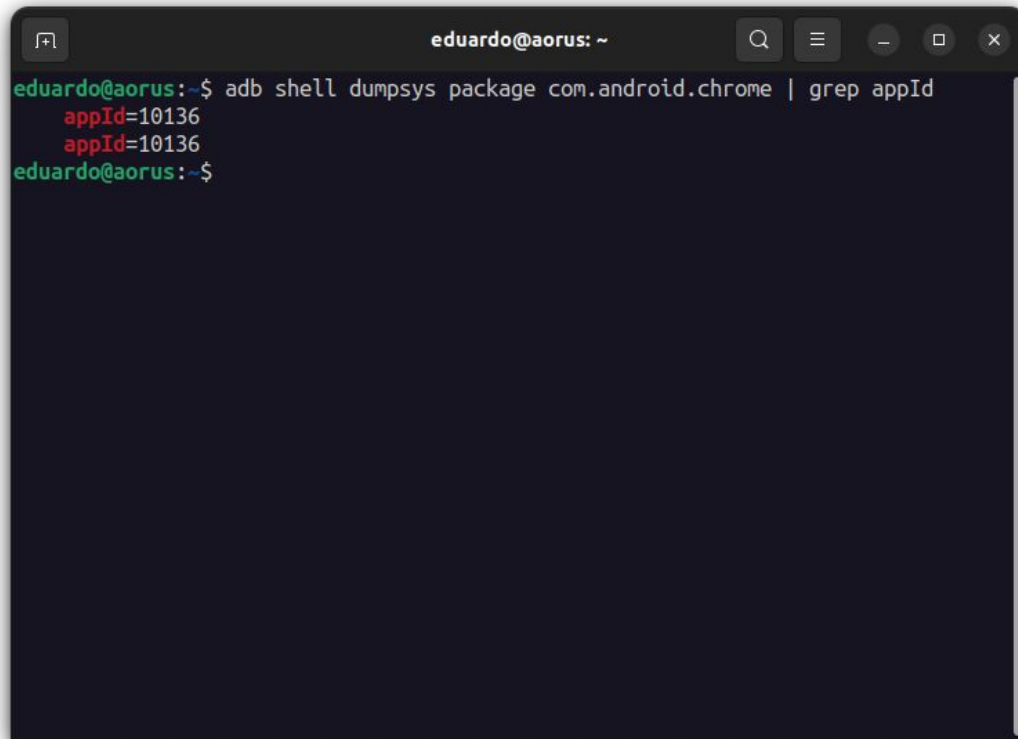
- O Android aproveita o conceito de *user ID* (UID) e *group ID* (GID) do Linux;
- UID e GID são mapeados genericamente para *Android ID* (AID);
- Cada app possui um AID específico, atribuído pelo sistema no momento da instalação.

Trecho de definição de AIDs de usuários críticos do sistema. Disponível em:

https://android.googlesource.com/platform/system/core/+master/libcutils/include/private/android_file_system_config.h

```
39  /* This is the main Users and Groups config for the platform.
40   * DO NOT EVER RENUMBER
41   */
42
43  #define AID_ROOT 0 /* traditional unix root user */
44
45  /* The following are for tests like LTP and should only be used for testing. */
46  #define AID_DAEMON 1 /* Traditional unix daemon owner. */
47  #define AID_BIN 2 /* Traditional unix binaries owner. */
48  #define AID_SYS 3 /* A group with the same gid on Linux/macOS/Android. */
49
50  #define AID_SYSTEM 1000 /* system server */
51
52  #define AID_RADIO 1001 /* telephony subsystem, RIL */
53  #define AID_BLUETOOTH 1002 /* bluetooth subsystem */
54  #define AID_GRAPHICS 1003 /* graphics devices */
55  #define AID_INPUT 1004 /* input devices */
56  #define AID_AUDIO 1005 /* audio devices */
57  #define AID_CAMERA 1006 /* camera devices */
58  #define AID_LOG 1007 /* log devices */
59  #define AID_COMPASS 1008 /* compass device */
60  #define AID_MOUNT 1009 /* mountd socket */
61  #define AID_WIFI 1010 /* wifi subsystem */
```

Sandbox (III)



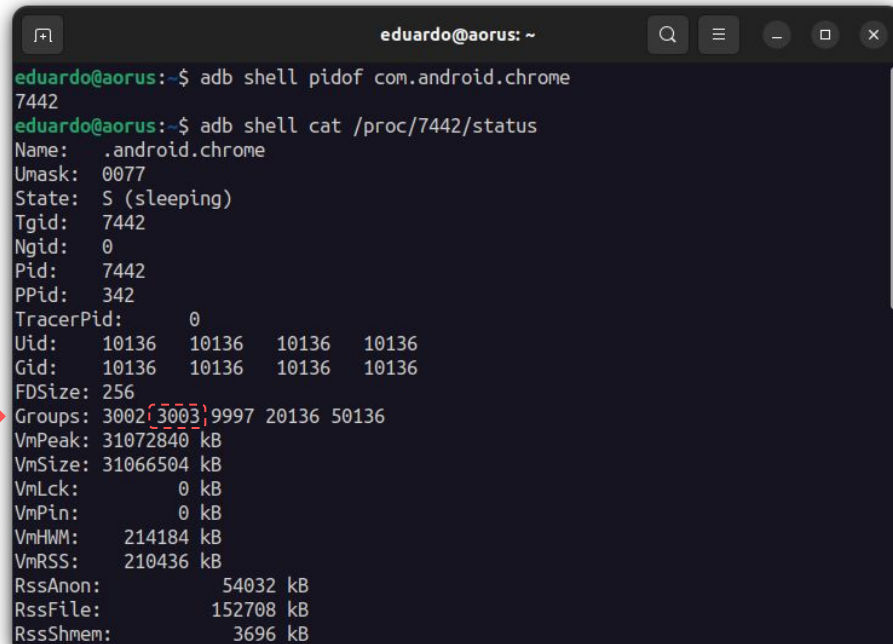
```
eduardo@aorus: ~  
eduardo@aorus:~$ adb shell dumpsys package com.android.chrome | grep appId  
  appId=10136  
  appId=10136  
eduardo@aorus:~$
```

Localizando o AID
do Chrome em um
dispositivo Android.

Sandbox (IV)


- Quando o usuário inicia um app, o sistema atribui os AIDs do app ao *process ID* (PID) correspondente;
- Assim, o app consegue acessar o sistema de arquivos e os recursos de sistema necessários ao seu funcionamento.

“Dumpando” o status do processo do Chrome em um dispositivo Android.



```
eduardo@aorus: ~  
eduardo@aorus:~$ adb shell pidof com.android.chrome  
7442  
eduardo@aorus:~$ adb shell cat /proc/7442/status  
Name: .android.chrome  
Umask: 0077  
State: S (sleeping)  
Tgid: 7442  
Ngid: 0  
Pid: 7442  
PPid: 342  
TracerPid: 0  
Uid: 10136 10136 10136 10136  
Gid: 10136 10136 10136 10136  
FDSize: 256  
Groups: 3002(3003)9997 20136 50136  
VmPeak: 31072840 kB  
VmSize: 31066504 kB  
VmLck: 0 kB  
VmPin: 0 kB  
VmHWM: 214184 kB  
VmRSS: 210436 kB  
RssAnon: 54032 kB  
RssFile: 152708 kB  
RssShmem: 3696 kB
```

Sandbox (V)



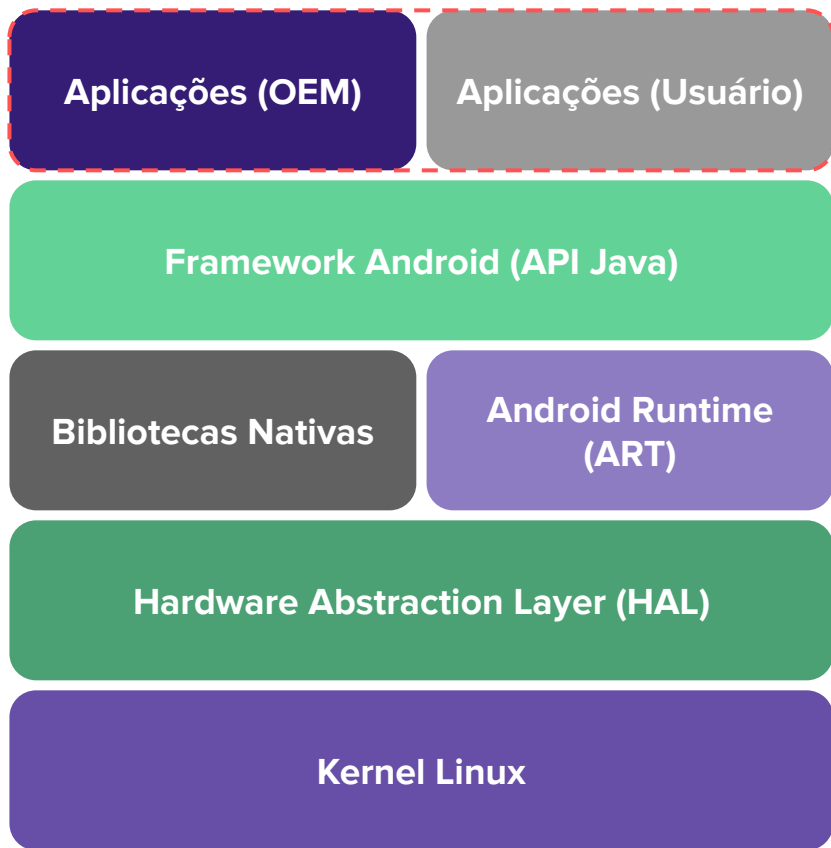
```
160  /* The 3000 series are intended for use as supplemental group id's only.
161   * They indicate special Android capabilities that the kernel is aware of. */
162  #define AID_NET_BT_ADMIN 3001 /* bluetooth: create any socket */
163  #define AID_NET_BT 3002      /* bluetooth: create sco, rfcomm or l2cap sockets */
164  #define AID_INET 3003        /* can create AF_INET and AF_INET6 sockets */
165  #define AID_NET_RAW 3004     /* can create raw INET sockets */
166  #define AID_NET_ADMIN 3005   /* can configure interfaces and routing tables. */
167  #define AID_NET_BW_STATS 3006 /* read bandwidth statistics */
168  #define AID_NET_BW_ACCT 3007 /* change bandwidth statistics accounting */
169  #define AID_READPROC 3009    /* Allow /proc read access */
170  #define AID_WAKELOCK 3010    /* Allow system wakelock read/write access */
171  #define AID_UHID 3011        /* Allow read/write to /dev/uhid node */
172  #define AID_READTRACEFS 3012 /* Allow tracefs read */
173  #define AID_VIRTUALMACHINE 3013 /* Allows VMs to tune for performance*/
```

Sandbox (VI)

- Resumindo...
 - O **kernel** faz cumprir o isolamento entre os apps e o acesso a recursos de sistema usando **recursos padrão do Linux**;
 - Se um app tenta acessar recursos que extrapolam os privilégios que lhe foram concedidos, ele será **impedido** de fazer isso.

Outros mecanismos de segurança do Android

- Binder;
- Sistema de permissões;
- SELinux;
- Keystore;
- TEE;
- Etc.



← Nós estamos aqui!

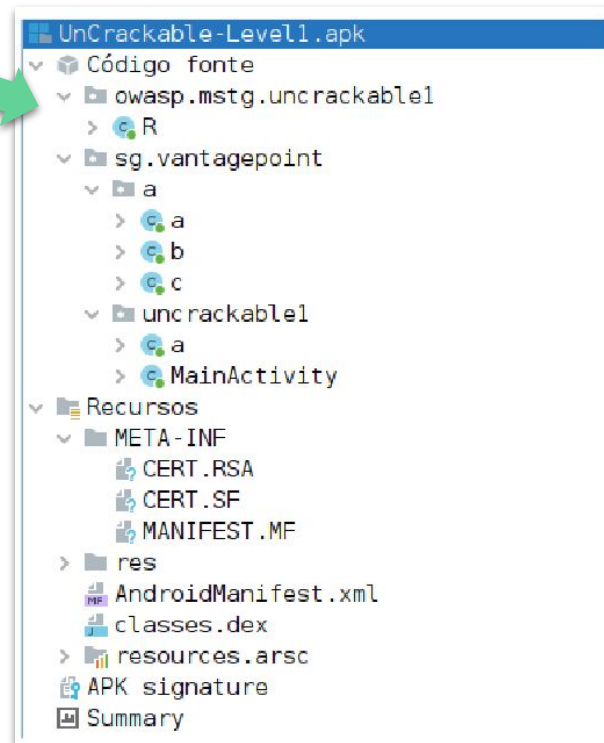
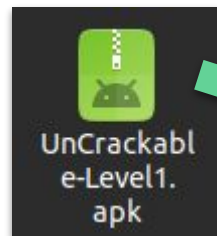
A Arquitetura do Android (recap)

Anatomia básica de um app Android

- O formato **mais básico*** de app Android é o **APK (Android Package Kit)**;
- O APK é um *archive* que reúne todos os arquivos que a aplicação precisa. Ele é comprimido para um formato similar a ZIP;
- O Android requer que APKs sejam **assinados digitalmente**** previamente à sua instalação.

*Vide AAB (Android Application Bundle)

**Vide criptografia de chave pública (criptografia assimétrica)



Um APK revertido: Android UnCrackable L1 (OWASP). Disponível em:

<https://mas.owasp.org/crackmes/Android/#android-uncrackable-l1>

O manifest (I)

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <manifest package = "com.example.test">
3
4      <!-- NAO COMPILA -->
5      <!-- Manifest ficticio elaborado para fins meramente didaticos -->
6
7      <!-- Definicao da aplicacao e seus componentes -->
8      <application android:label = "@string/app_name">
9
10         <!-- Primeiro componente: MainActivity da aplicacao -->
11         <activity android:label = "@string/app_name" android:name="com.example.test.MainActivity">
12             <intent-filter>
13                 <action android:name="android.intent.action.MAIN" />
14                 <category android:name="android.intent.category.LAUNCHER" />
15             </intent-filter>
16         </activity>
17
18         <!-- Segundo componente: outra activity -->
19         <activity android:name="com.example.test.AnotherActivity" />
20
21     </application>
22
23 </manifest>
```

Manifest de um app Android fictício.

O manifest (II)

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <manifest package = "com.example.test">
3
4     <!-- NAO COMPILA -->
5     <!-- Manifest ficticio elaborado para fins meramente didaticos -->
6
7     <!-- Definicao da aplicacao e seus componentes -->
8     <application android:label = "@string/app_name">
9
10         <!-- Primeiro componente: MainActivity da aplicacao -->
11         <activity android:label = "@string/app_name" android:name="com.example.test.MainActivity">
12             <intent-filter>
13                 <action android:name="android.intent.action.MAIN" />
14                 <category android:name="android.intent.category.LAUNCHER" />
15             </intent-filter>
16         </activity>
17
18         <!-- Segundo componente: outra activity -->
19         <activity android:name="com.example.test.AnotherActivity" />
20
21     </application>
22
23 </manifest>
```

Definição de componentes de aplicação

Manifest de um app Android fictício.

Componentes de uma aplicação Android

Activity

Interage diretamente com o usuário, através de uma interface gráfica. Tag de manifest:

`<activity>`

Receiver

É um componente através do qual a aplicação pode receber *intents*. Tag de manifest:

`<receiver>`

Service

Não possui interface gráfica e a execução ocorre em background, sem a necessidade de interação do usuário. Tag de manifest:

`<service>`

Provider

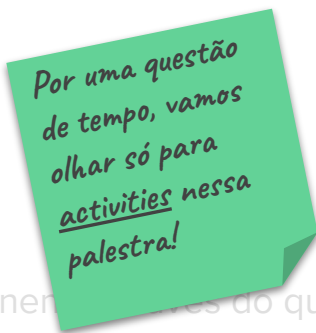
É um componente através do qual a aplicação, utilizando uma interface estruturada, é capaz de prover dados a outras aplicações (análogo a uma base de dados). Tag de manifest: `<provider>`

Componentes de uma aplicação Android

Activity

Interage diretamente com o usuário, através de uma interface gráfica. Tag de manifest:

`<activity>`



Receiver

É um componente através do qual a aplicação pode receber *intents*. Tag de manifest:

`<receiver>`

Service

Não possui interface gráfica e a execução ocorre em background, sem a necessidade de interação do usuário. Tag de manifest:

`<service>`

Provider

É um componente através do qual a aplicação, utilizando uma interface estruturada, é capaz de prover dados a outras aplicações (análogo a uma base de dados). Tag de manifest: `<provider>`

Intents

- Um intent é um objeto que descreve uma "mensagem" que uma aplicação deseja trocar com outra;
- Ele contém uma **operação** que a aplicação remetente deseja executar e **informação adicional** sobre a operação em questão.

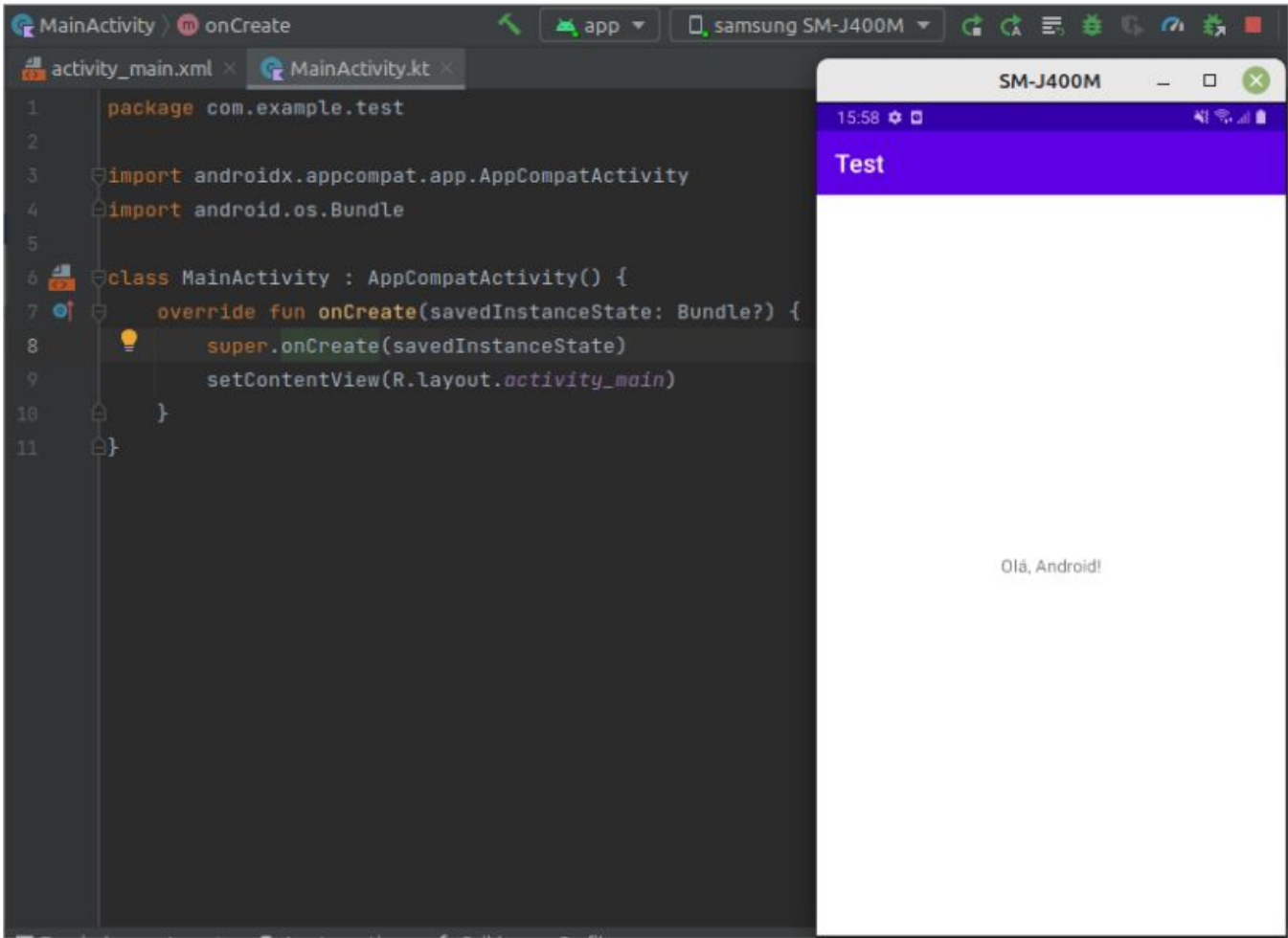
Ex.: app tem uma activity que gera um comprovante de pagamento em formato de imagem. O app pode lançar um **intent solicitando ao sistema que encontre uma aplicação externa capaz de abrir o comprovante.**

Esse intent contém:

- A ação desejada (*view*);
- O *path* da imagem.

Um intent pode ser endereçado a uma aplicação específica ou, ainda, a nenhuma aplicação em específico, cabendo ao sistema encontrar uma aplicação capaz de resolvê-lo.*

*Vide intent resolution



MainActivity de
com.example.test
(Kotlin)

Takeaways dessa primeira parte...

Aplicações Android...

- São essencialmente aplicações **escritas em linguagens de alto nível** (XML, Java, Kotlin e até C/C++);
- São **compiladas para bytecode DEX**, sendo que uma máquina virtual específica, chamada Android Runtime (ART), é capaz de interpretá-lo;
- **É fácil reverter o código de uma aplicação Android compilada** (APK) para uma representação de alto nível;
- Mecanismos de comunicação da API do Android permitem que aplicações **tentem interagir livremente** com outras aplicações instaladas no mesmo dispositivo Android.

Parte 2. *Security*

Vulnerabilidades em Android

Android Developers > Design & Plan > App quality > Privacy & Security


Isso foi útil?  

Mitigate security risks in your app

By making your app more secure, you help preserve user trust and device integrity.

This page presents a set of common security issues that Android app developers face. You can use this content in the following ways:

- Learn more about how to proactively secure your apps.
- Understand how to react in the event that one of these issues is discovered in your app.

The following list contains links to dedicated pages for each individual issue, sorted into categories based on [OWASP MASVS](#)  controls. Each page includes a summary, impact statement, and tips for mitigating the risk to your app.

MASVS-STORAGE: Storage

OWASP category description

- Backup Leaks
- Improperly Exposed Directories to FileProvider
- Log Info Disclosure
- Path traversal
- Sensitive Data Stored in External Storage

Nesta página

[MASVS-STORAGE: Storage](#)

[MASVS-CRYPTO: Cryptography](#)

[MASVS-NETWORK: Network Communication](#)

[MASVS-PLATFORM: Platform Interaction](#)

[MASVS-CODE: Code Quality](#)

Recurso recomendado:
Android Developers
Common risks.
Disponível em:

<https://developer.android.com/privacy-and-security/risks>

Vamos olhar para
uma vulnerabilidade
em específico

Task Hijacking (API level < 28) [Ren et al., 2015]* (I)

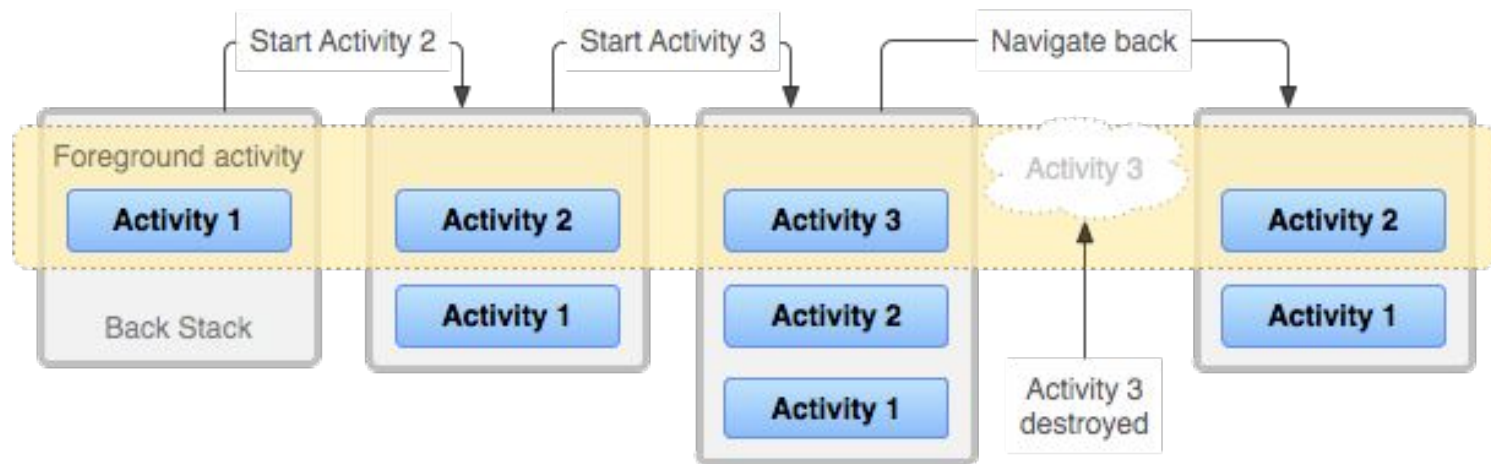
- No Android, existe o conceito de **back stack**;
- Uma **back stack** é um conjunto de activities que o usuário utiliza para cumprir uma tarefa (*task*);
- A activity mais recente aparece no **topo** da back stack, e recebe o nome de **foreground activity**;
- A back stack cuja foreground activity está aparecendo na tela recebe o nome de **foreground task**.

*Vide “Towards Discovering and Understanding Task Hijacking in Android”. Disponível em:
<https://www.usenix.org/system/files/conference/useenixsecurity15/sec15-paper-ren-chuangang.pdf>

Figura extraída de
<https://youtu.be/ZCZC-TqJQTU>



Task Hijacking (API level < 28) [Ren et al., 2015] (II)



Transições em uma back stack. Disponível em: https://developer.android.com/images/fundamentals/diagram_backstack.png

Task Hijacking (API level < 28) [Ren et al., 2015] (III)

- A API do Android define o conceito de **task affinity**, que é uma maneira de definir a *task preferencial* de um app. Ex.: app expressa *afinity* pela task do PayPal (com.paypal.android.p2pmobile);
- Existe, ainda, o conceito de **task reparenting**, uma maneira de especificar que uma activity pode ser movida para a task do app com o qual ela tem *task affinity*, na próxima vez que a task do app com o qual ela tem *task affinity* for para foreground.

<activity>

Sintaxe:

```
<activity android:allowEmbedded=["true" | "false"]  
          android:allowTaskReparenting=["true" | "false"]  
          android:alwaysRetainTaskState=["true" | "false"]
```

```
          android:supportsPictureInPicture=["true" | "false"]  
          android:taskAffinity="string"  
          android:theme="resource or theme"  
          android:uiOptions=["none" | "splitActionBarWhenNarrow"]  
          android:windowSoftInputMode=["stateUnspecified",  
                                         "stateUnchanged", "stateHidden",  
                                         "stateAlwaysHidden", "stateVisible",  
                                         "stateAlwaysVisible", "adjustResize", "adjustPan"]
```

```
...  
</activity>
```

Trechos da documentação da API do Android para <activity>. Disponível em: <https://developer.android.com/guide/topics/manifest/activity-element>

Task Hijacking (API level < 28) [Ren et al., 2015] (IV)

Code Blame 29 lines (21 loc) · 978 Bytes

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:tools="http://schemas.android.com/tools"
4     package="com.example.dummy1">
5
6     <application
7         android:allowBackup="true"
8         android:icon="@mipmap/ic_launcher"
9         android:label="@string/app_name"
10        android:roundIcon="@mipmap/ic_launcher_round"
11        android:supportsRtl="true"
12        android:theme="@style/AppTheme"
13        tools:ignore="GoogleAppIndexingWarning">
14
15
16     <activity android:exported="true" android:allowTaskReparenting="true" android:taskAffinity="##VULNERABLE_APP##" android:name=".Main" />
17
```

Exemplo de activity maliciosa que ataca ##VULNERABLE_APP## usando task hijacking. Autor da prova de conceito: Ivan Marković. Disponível em: <https://github.com/Ivan-Markovic/Android-Task-Injection>



A phishing UI from the malware shows up instead of the authentic PayPal UI !



Presidência da República
Casa Civil
Subchefia para Assuntos Jurídicos



LEI Nº 12.737, DE 30 DE NOVEMBRO DE 2012.

Vigência

Dispõe sobre a tipificação criminal de delitos informáticos; altera o Decreto-Lei nº 2.848, de 7 de dezembro de 1940 - Código Penal; e dá outras providências.

A PRESIDENTA DA REPÚBLICA Faço saber que o Congresso Nacional decreta e eu sanciono a seguinte Lei:

Art. 1º Esta Lei dispõe sobre a tipificação criminal de delitos informáticos e dá outras providências.

Art. 2º O Decreto-Lei nº 2.848, de 7 de dezembro de 1940 - Código Penal, fica acrescido dos seguintes arts. 154-A e 154-B:

“Invasão de dispositivo informático

Art. 154-A. Invadir dispositivo informático alheio, conectado ou não à rede de computadores, mediante violação indevida de mecanismo de segurança e com o fim de obter, adulterar ou destruir dados ou informações sem autorização expressa ou tácita do titular do dispositivo ou instalar vulnerabilidades para obter vantagem ilícita:

Pena - detenção, de 3 (três) meses a 1 (um) ano, e multa.

Vale lembrar...

- Aplicações Android podem ser **revertidas** para código Java;
- O processo de reverter uma aplicação com o intuito de compreender o seu funcionamento recebe o nome de **engenharia reversa** (*reversing*);
- Em certos contextos, adversários podem tentar fazer engenharia reversa de apps com o intuito de **subjugar proteções de segurança**;
- Nesses contextos, cabe analisar/reduzir a vulnerabilidade de apps a engenharia reversa.*

*Vide RASP (Runtime Application Self-Protection)

Sobre o SiDi @segueosidi

Sobre o SiDi

Olá, somos **um dos maiores** institutos de ciência e tecnologia do Brasil.

Já estamos em **Campinas, Manaus e Recife** e já somos mais de **800 colaboradores**.

Temos quase **20 anos** de história e trazemos em nossa bagagem mais de **1000 projetos** realizados.

Criamos soluções **inovadoras** que **transformam** a vida de pessoas e organizações. **@segueosidi**



#VemSerSiDier
sidi.gupy.io



Bônus: sobre trabalhar com segurança

Por onde eu começo?

Aproveite a universidade



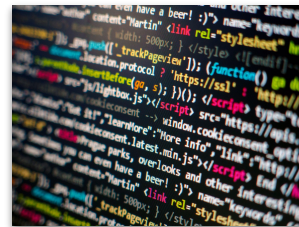
Preocupe-se com aprender, leve a sério, envolva-se.

Comece a aprender sobre segurança



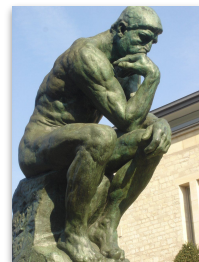
Existem diversos recursos gratuitos. Preocupe-se com aprender direito, não com volume.

Aprenda a escrever software



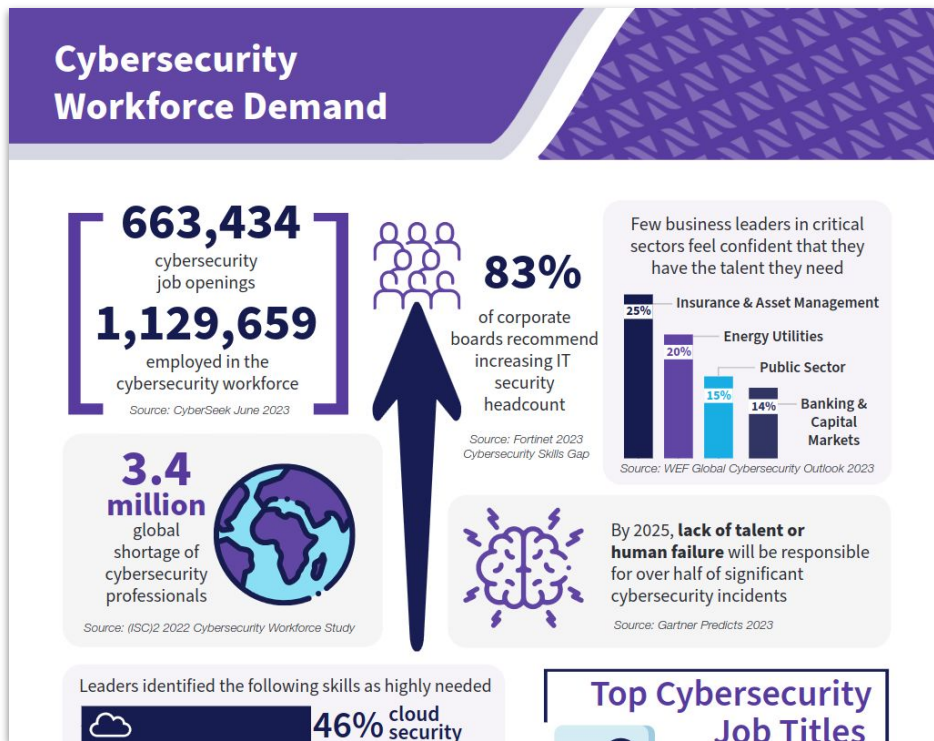
Você não precisa ser um engenheiro de software excepcional, mas muito ajuda.

Especialize-se



Encontre um nicho de segurança que você goste e procure aprender mais sobre ele. Existem vários.

Procura-se: bons profissionais de segurança



Cybersecurity Workforce Demand (NIST). Disponível em:
https://www.nist.gov/system/files/documents/2023/06/05/NICE%20FactSheet_Workforce%20Demand_Final_20211202.pdf

Perguntas?
