**Trabalho prático - Parte 2**

**Aplicações Mobile**

1 a)

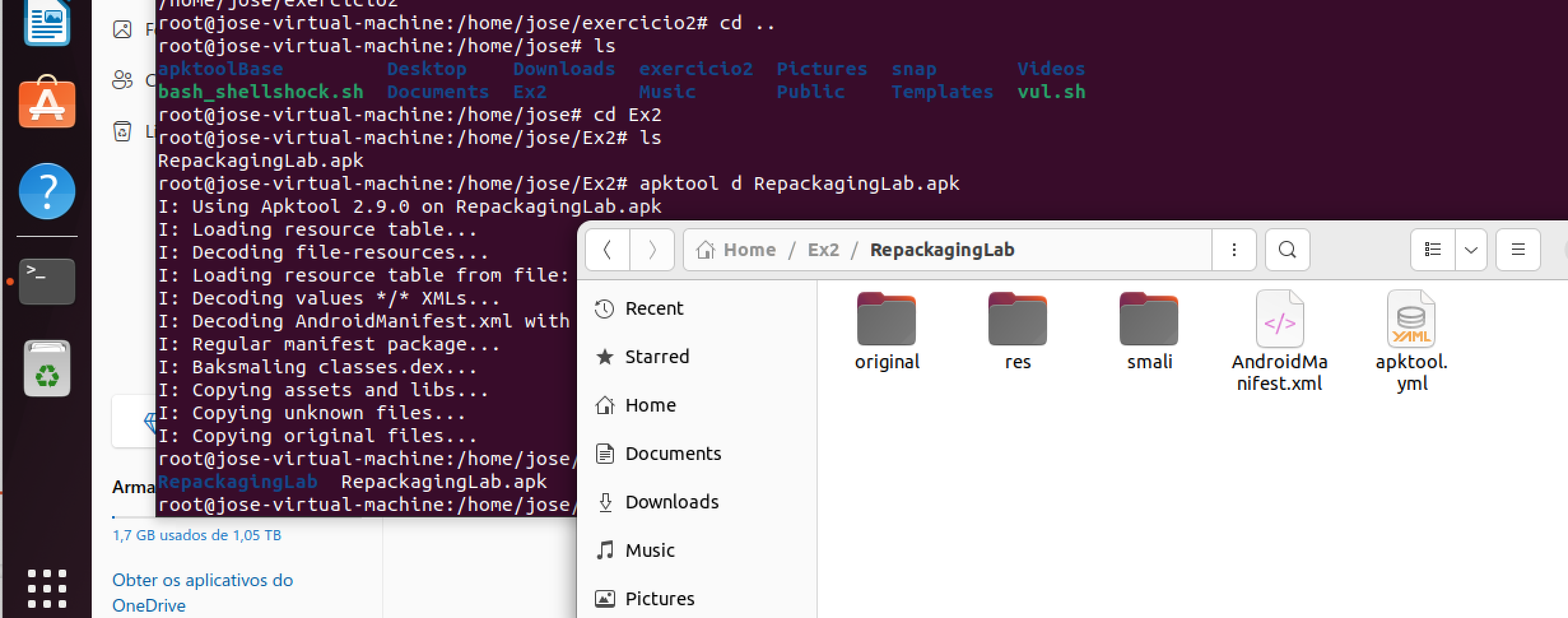
Instalando o APK no AVD a imagem que aparece é a do logotipo do android em azul sobre um fundo branco

1 b)

Vamos usar o APK Tool que é uma ferramenta para fazer o dissasembling dos APK android assim como para voltar a fazer o assembling depois de eventuais mudanças terem acontecido.

A instalação do APK Tool foi feita de acordo com o Git repository indicado no PDF de segurança em aplicações mobile na página 21.

Para o depackaging o comando é: apktool d RepackagingLab.apk



Depackaging do .apk

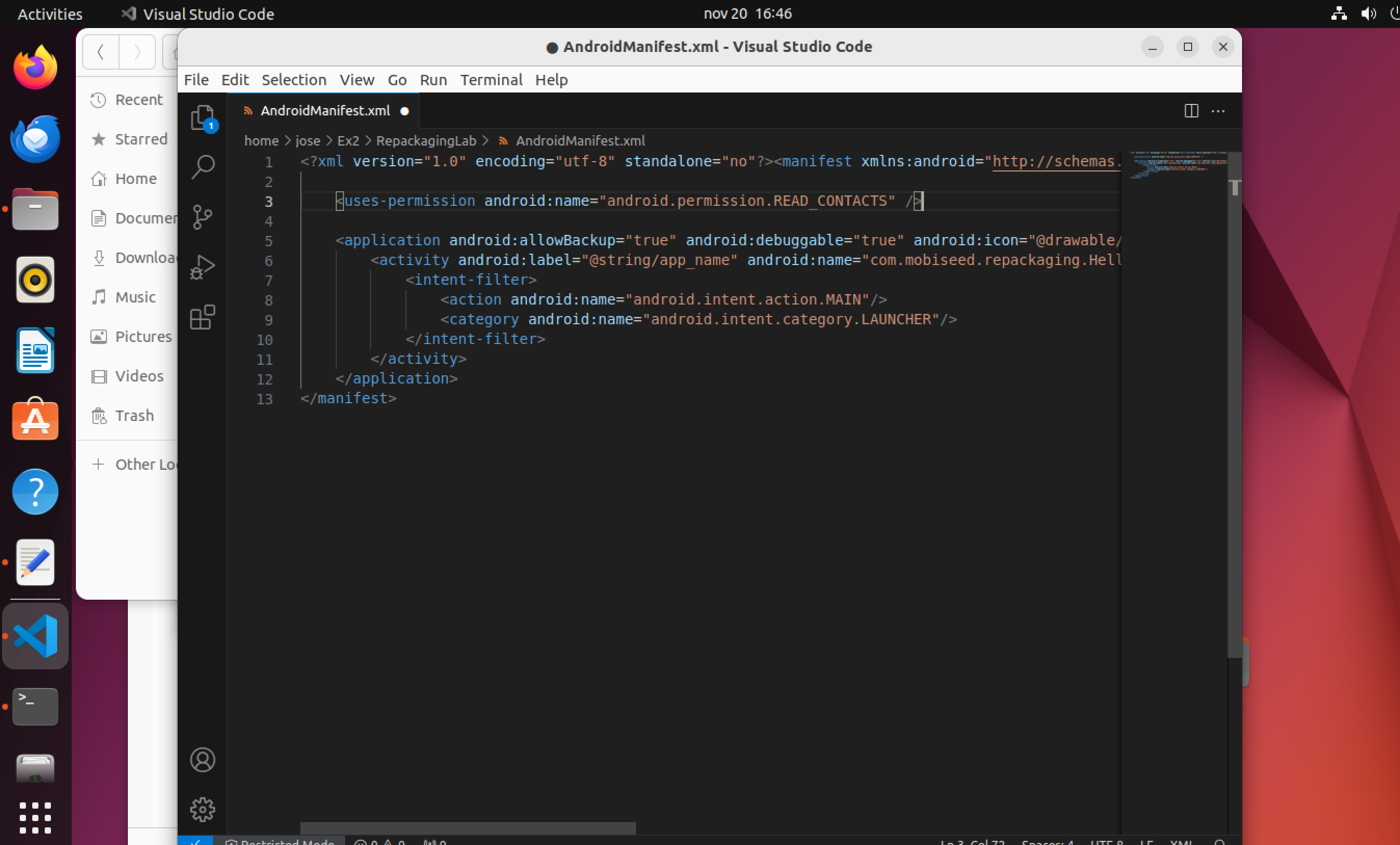
É criada uma estrutura de ficheiros onde a pasta res tem ficheiros de resources como imagens e a pasta smali tem uma versão human readable de código que resulta do processo de conversão de java bytecode em smali code durante o APK build process.

Smali é mais human readable que java bytecode, pode ser usado para estudar as aplicações ao nível de investigação, mas também pode ser usado para fazer reverse engineering e introduzir mudanças nas aplicações Android.

Hoje em dia na programação de multi propósitos usamos essencialmente as linguagens de objectos mas há poucas décadas atrás não era incomum programar em linguagens assembly. Assim sendo, para algumas pessoas ter este bytecode smali que é uma forma de assembly digamos assim pode não ser de todo dissuasor de alterar as aplicações.

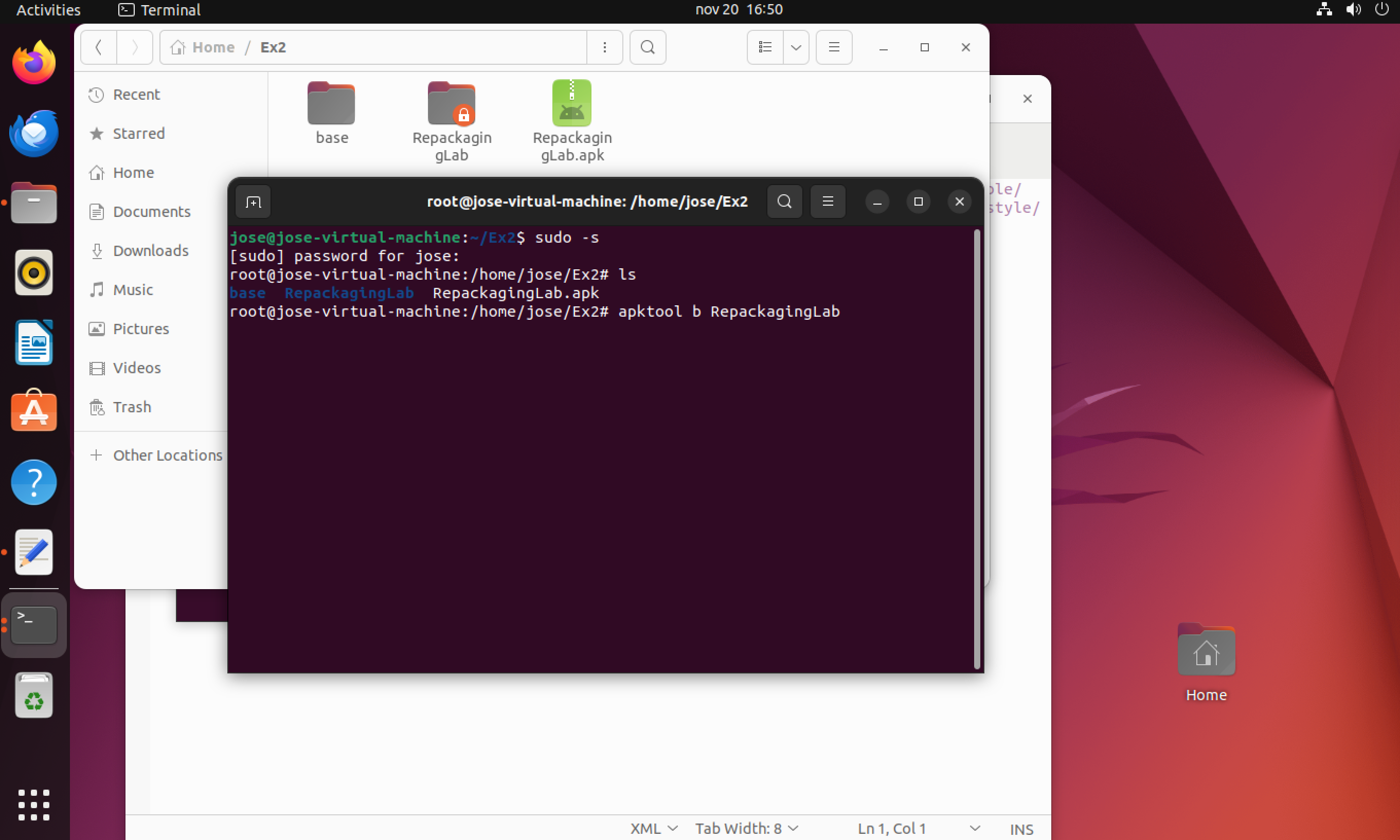
Para o exercício modificou-se a imagem indicada no enunciado (pasta res).

Tal como é indicado nos slides sobre segurança em aplicações mobile quando há ataques através de repackaging pode tirar-se partido de várias features do sistema operativo android, no exemplo na imagem abaixo mdificou-se o ficheiro AndroidManifest.xml de modo a permitir à aplicação aceder em modo leitura à lista de contactos. Fez-se meramente como exemplo uma vez que apenas é pedido que se altere uma imagem neste exercício.



Acrescentou-se permissão read\_contacts

E depois voltamos a fazer o repackaging desta vez com a aplicação já alterada



repackaging da app

Chegamos ao ponto em que para podermos prosseguir devemos dar uma assinatura digital ao APK.

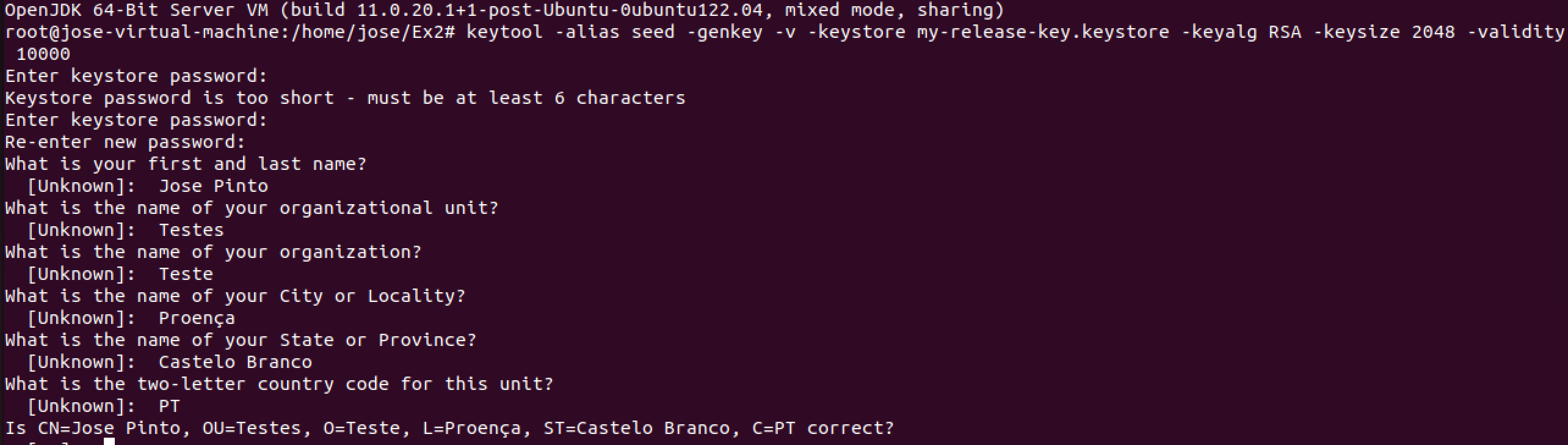
O facto da assinatura digital ser self certified levanta problemas a nível de segurança.

**1 c) Assinatura digital**

Primeiro devemos gerar um par de chaves pública e privada

Vamos tirar partido da biblioteca do Keytool do Java para gerar o par de chaves assimétricas pública/privada ( podemos ver o algoritmo a ser definido por -keyalg RSA).

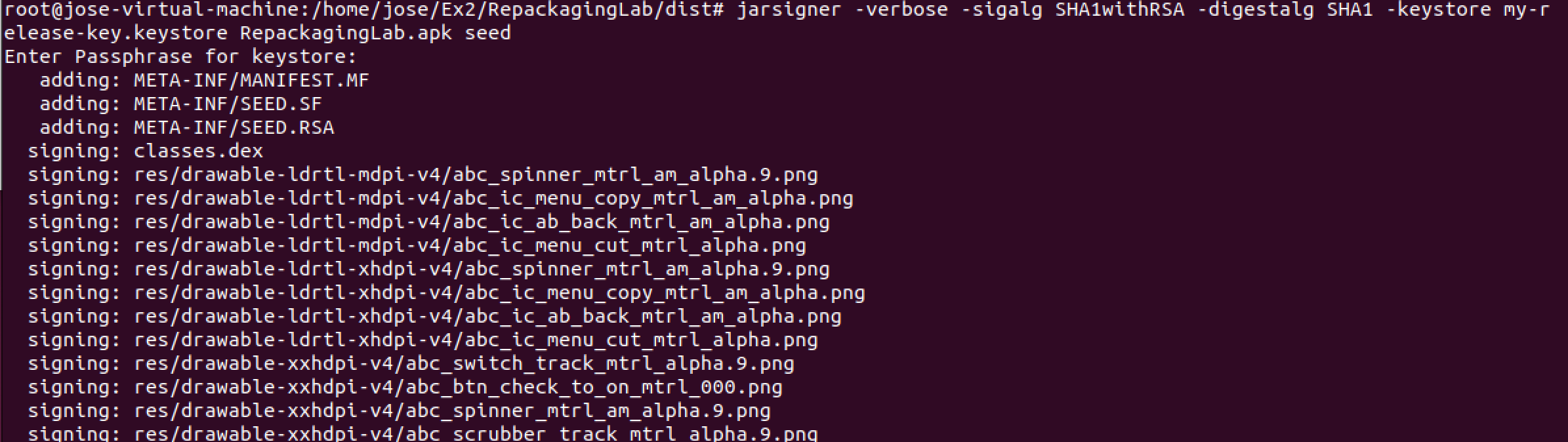
As chaves são armazenadas na Keystore para uso na assinatura da aplicação.



obter o par de chaves para a assinatura digital

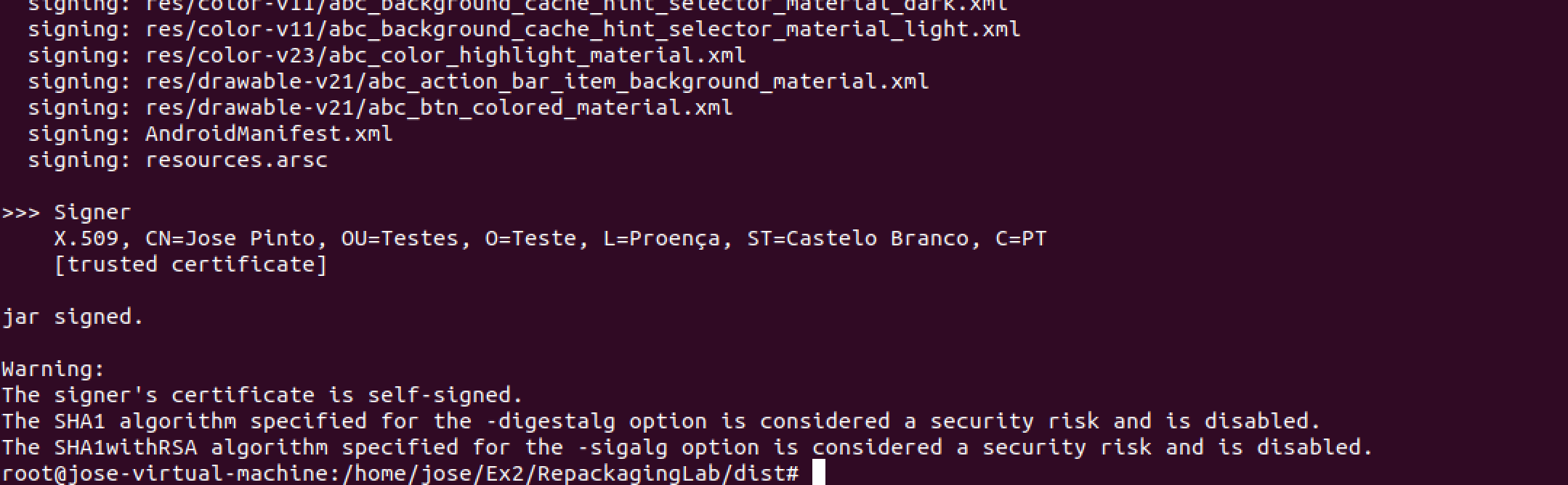
A Keystore do Java, que é um repositório para chaves criptográficas e certificados. O certificado usado para assinar é associado com uma chave privada e ambos são armazenados na keystore.

Vamos assinar digitalmente o novo APK usando



assinatura digital feita pela kestore do java

Depois de assinado podemos ver o aviso que o certificado é self-signed.



aviso de serf-signed certificate

Quer este aviso dizer que não há uma autoridade mediadora para garantir que estamos perante a empresa que efetivamente providencia uma determinada aplicação ao invés de poder ser um hacker como é objectivo deste exercício.

Para listar os detalhes do certificado digital guardado na keystore usamos novamente o comando keytool

root@jose-virtual-machine:/home/jose/Ex2/RepackagingLab/dist# keytool -list -v -keystore my-release-key.keystore

Enter keystore password:

Keystore type: PKCS12

Keystore provider: SUN

Your keystore contains 1 entry

Alias name: seed

Creation date: 20/11/2023

Entry type: PrivateKeyEntry

Certificate chain length: 1

Certificate[1]:

Owner: CN=Jose Pinto, OU=Testes, O=Teste, L=Proença, ST=Castelo Branco, C=PT

Issuer: CN=Jose Pinto, OU=Testes, O=Teste, L=Proença, ST=Castelo Branco, C=PT

Serial number: 358d9b97

Valid from: Mon Nov 20 17:02:41 WET 2023 until: Fri Apr 07 18:02:41 WEST 2051

Certificate fingerprints:

SHA1: FA:1F:59:7D:29:13:96:52:70:4D:5F:73:58:8B:B5:4F:70:67:D6:1E

SHA256: 6C:73:4D:D5:CE:50:0C:FF:29:F0:F4:1A:29:29:BC:A6:7E:45:91:A2:35:EE:8A:02:9A:86:67:8D:CC:36:62:33

Signature algorithm name: SHA256withRSA

Subject Public Key Algorithm: 2048-bit RSA key

Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false

SubjectKeyIdentifier [

KeyIdentifier [

0000: 5C 30 4E E8 80 0E 94 FF DD 60 8E 85 9D 82 14 17 \0N......`......

0010: C3 3F 42 D3 .?B.

]

]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Conclusão do ponto 1

Se fornecer o APK alterado a um utilizador que não use os canais de distibuição seguros como a Google Play ou a App Store ele vai estar perante uma aplicação Android que efetivamente está digitalmente assinada, no entanto o que não estará a coincidir é o hash de controlo.

Podemos ver na seguinte imagem o hasf sha256 do novo Apk por repackaging obtido pelo comando

root@jose-virtual-machine:/home/jose/Ex2/RepackagingLab/dist# keytool -printcert -jarfile RepackagingLab.apk

Signer #1:

Certificate #1:

Owner: CN=Jose Pinto, OU=Testes, O=Teste, L=Proença, ST=Castelo Branco, C=PT

Issuer: CN=Jose Pinto, OU=Testes, O=Teste, L=Proença, ST=Castelo Branco, C=PT

Serial number: 358d9b97

Valid from: Mon Nov 20 17:02:41 WET 2023 until: Fri Apr 07 18:02:41 WEST 2051

Certificate fingerprints:

SHA1: FA:1F:59:7D:29:13:96:52:70:4D:5F:73:58:8B:B5:4F:70:67:D6:1E

SHA256: 6C:73:4D:D5:CE:50:0C:FF:29:F0:F4:1A:29:29:BC:A6:7E:45:91:A2:35:EE:8A:02:9A:86:67:8D:CC:36:62:33

Signature algorithm name: SHA256withRSA

Subject Public Key Algorithm: 2048-bit RSA key

Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false

SubjectKeyIdentifier [

KeyIdentifier [

0000: 5C 30 4E E8 80 0E 94 FF DD 60 8E 85 9D 82 14 17 \0N......`......

0010: C3 3F 42 D3 .?B.

]

]

Se estivermos a comparar com o hash guardado no distribuidor 'oficial' como a Google Play vamos notar diferenças pois as alterações feitas refletem-se no hash.

Diríamos que no mundo empresarial um hack envolvendo a Google Play nã será fácil logo este tipo de ataques será mais provável num site de downloads paralelo por assim dizer, aí o facto do APK estar assinado digitalmente vai fazer com que possa ser instalado no S.O. Android e o utilizador num cenário mais comum não irá comparar a integridade por via do hash, dando origem ao possível hack.

**Exercício 2**

2 a) testou OK

2b) Correndo o comando

java -jar Bytecode-Viewer-2.11.2.jar Lab2-2.apk

Considerando o código obtido ver as alíneas em comentário

public static boolean a(String var0) {

byte[] var1 = Base64.getDecoder().decode("5UJiFctbmgbDoLXmpL12mkno8HT4Lv8dlat8FxR2GOc=");

var1 = a1.a(b("8d127684cbc37c17616d806cf50473cc"), var1); //nota: b é método que converte em byte[]

String DecodedText=new String(var1);

return var0.equals(new String(var1));

sendo a1.a

public static byte[] a(byte[] var0, byte[] var1) throws Exception {

// alinea ii

//chave dada pelo byte array de var0="8d127684cbc37c17616d806cf50473cc"

//alinea iii

//que vai ser usado no Algoritmo AES

Key key = new SecretKeySpec(var0, "AES");

Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");

//alinea i

//vai desencriptar var1="5UJiFctbmgbDoLXmpL12mkno8HT4Lv8dlat8FxR2GOc=" que é a frase cifrada

cipher.init(Cipher.DECRYPT\_MODE, key);

return cipher.doFinal(var1);

}

O código foi obtido por dois passos de reverse engineering ao bytecode Smali do APK pela aplicação Bytecode-Viewer-2.11.2.jar , o download da mesma feito a partir do site indicado pelo enunciado.

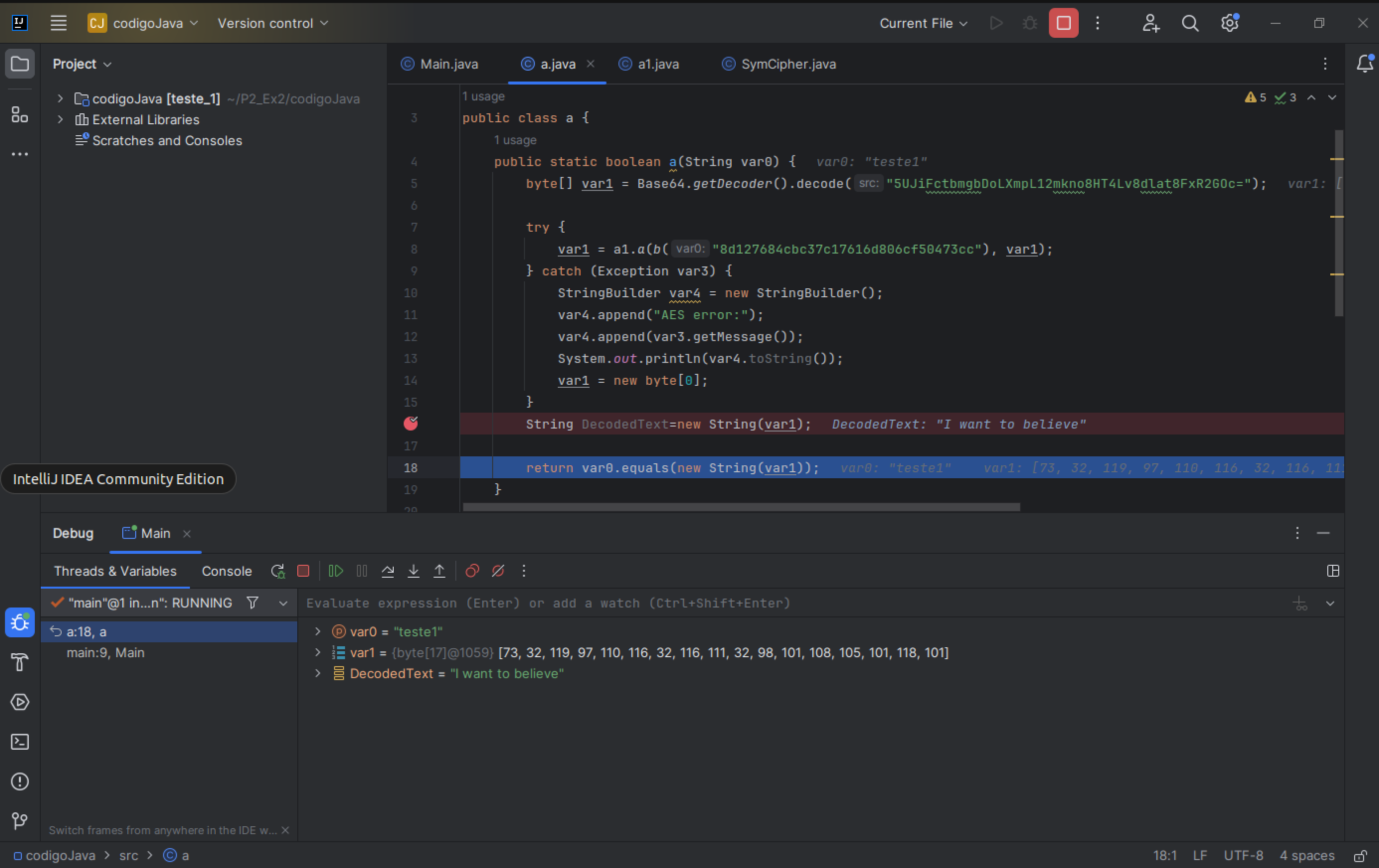
Em primeira fase foi convertido em Java

Em segunda fase extrairam-se as classes relevantes para fazer a simulaçã de um input e comparação com password.

Afalha é que ao invés de encriptra e comparar os valores encriptados o programador deixou a chave de encriptação no código estando a usá-la para fazer desencriptação.

Depois de ter desencriptado a password faz a comparação, ora neste cenário e numa primeira abordagem basta intersectar a linha de comparação de valores desencriptados como se mostra na imagem seguinte.

O código está nas classes Main.java, a.java, a1.java, SymCipher.java (consulta)



intersectar comparação de valores desencriptados

Password = I want to beleive

Mas mesmo que o programador estivesse a comparar valores encriptados o facto de a chave estar exposta deste modo permite desencriptar a password que está também em código.

A falha de segurança acontece por estar a ser usada uma encriptação simétrica (Advanced Encription Standars Algorithm AES) logo a mesma chave é usada para encriptar e desencriptar, nas classes usadas o cypher object tanto serve para encriptar como desencriptar cipher.init(Cipher.DECRYPT\_MODE, key);

Alternativas mais seguras seriam a utilização de chave pública e envio da cifra para sistema externo (ao APK ) de autenticação, ou por exemplo se for feito uma query de comparação de Hash de password em runtime.