Análise de segurança à aplicação Autenticação.gov

Mestrado em Cibersegurança e Informática Forense

Tiago João Rodrigues da Silva

Leiria, novembro de 2020

Análise de segurança à aplicação Autenticação.gov

Mestrado em Cibersegurança e Informática Forense

Tiago João Rodrigues da Silva

Trabalho de projeto realizado sob a orientação do Professor Doutor Miguel Monteiro de Sousa Frade (Miguel.frade@ipleiria.pt).

Leiria, novembro de 2020

Originalidade e Direitos de Autor

O presente relatório de projeto é original, elaborado unicamente para este fim, tendo sido devidamente citados todos os autores cujos estudos e publicações contribuíram para o elaborar.

Agradecimentos

No início deste projeto de mestrado, não poderia deixar de agradecer a todas as pessoas que, de alguma forma, colaboraram para a realização deste trabalho.

Primeiramente agradeço à minha entidade patronal, Santuário de Fátima por ter-me proporcionado as condições necessárias para a realização deste mestrado e pelo desenvolvimento deste projeto.

Um agradecimento especial à minha colega Ana Vazão pela motivação e apoio diário que me deu ao longo do desenvolvimento deste projeto.

Um agradecimento especial ao Professor Doutor Miguel Monteiro de Sousa Frade, pelos ensinamentos como professor no decorrer dos ciclos de estudos e por todo o apoio ao longo deste projeto, e em particular pela orientação científica do mesmo.

Por último, mas não menos importante, um enorme agradecimento à minha família, em especial à Catarina, Carlota e ao Bernardo pelo apoio, paciência e pelas limitações ao longo destes últimos anos.

Resumo

Nos últimos anos verificou-se um rápido crescimento da transformação digital dos processos das organizações. Atualmente é um facto admitido que os sistemas informáticos estão cada vez mais presentes no dia-a-dia dos cidadãos, tendo assumido um papel fundamental nas suas vidas. O seu constante e rápido crescimento tem-se generalizado através do desenvolvimento de múltiplas aplicações disponíveis na Web ou destinadas a dipositivos móveis, que facilitam as suas tarefas diárias e estabelecem inclusivamente novos meios de interação social.

A Administração Pública não podia ficar fora desta evolução das tecnologias de informação e dos sistemas de comunicação, sobretudo num período em que se visa a modernização e a simplificação administrativa. Tornou-se, assim, imprescindível para a Administração Pública evoluir nos serviços prestados ao cidadão adequando-os ao uso das tecnologias, promovendo uma Sociedade de Informação, baseada no desenvolvimento tecnológico.

O presente projeto tem como objetivo o estudo da aplicação disponibilizada pela Administração Pública, Autenticação.gov para dispositivos móveis. Através do presente estudo foi possível perceber o comportamento da aplicação Auenticação.gov e efetuar um levantamento das suas vulnerabilidades. O sistema operativo móvel Android foi escolhido para este projeto, pelo domínio que representa no mercado dos dispositivos móveis, sendo o seu estudo importante para determinar a forma de efetuar uma análise às vulnerabilidades numa aplicação móvel.

Por fim efetuou-se um estudo sobre a Chave Móvel Digital de modo a perceber o seu funcionamento comparando as suas funcionalidades com as do Cartão de Cidadão.

Podemos assim concluir que a aplicação Autenticação.gov apresenta vulnerabilidades. Através da comparação das assinaturas eletrónicas efetuadas com o Cartão de Cidadão e a Chave Móvel Digital constatou-se que estas assinam os documentos de modo diferente.

Palavras-chave: Aplicações Móveis, Android, *Malware*, Autenticação.gov, Chave Móvel Digital.

Abstract

In recent years there has been a rapid growth of the digital transformation of organizations' processes. Nowadays, it is an admitted fact that computer systems are increasingly present in the citizens everyday lives, taking a fundamental role in their lives. Its constant and rapid growth has become widespread through the development of multiple applications available on the Web or designed for mobile devices, which facilitate their daily tasks and even establish new means of social interaction.

Public Administration could not be left out of this information technologies and communication systems evolution, especially at a time when modernization and administrative simplification are being pursued. It has become, therefore, essential for the Public Administration to evolve in the services provided to the citizen, adjusting them to the use of technologies, promoting an Information Society, based on technological development.

The purpose of this dissertation is to study the application Autenticação.gov for mobile devices, made available by the Public Administration. Through this study it was possible to understand the behavior of the Auenticação.gov application and carry out a survey of its vulnerabilities. The Android mobile operating system was chosen for this project, due to its dominance in the mobile devices market, and its study is important to determine how to perform a vulnerability analysis in a mobile application.

Finally, a study was carried out on the Digital Mobile Key in order to understand its operation by comparing its features with those of the Citizen Card.

We can therefore conclude that the Autenticação.gov application presents vulnerabilities. By comparing the electronic signatures made with the Citizen Card and the Digital Mobile Key, it was found that they sign the documents differently.

Keywords: Mobile application, Android, *Malware*, Autenticação.gov, Digital Mobile Key.

Índice

Lista de Figuras

Lista de tabelas

Lista de siglas e acrónimos

ABI	Application Binary Interface
ADB	Android Debug Bridge
API	Application Programming interface
APK	Android Package
ART	Android Runtime
BYOD	Bring Your Own Device
CA	Certification Authority
CC	Cartão de Cidadão
CMD	Chave Móvel Digital
DEX	Dalvik Executables
DSS	Document Security Store
EMVCAP	Europay MasterCard and Visa Chip Authentication Program
ETSI	European Telecommunications Standards Institute
EU	European Union
HOTP	HMAC-based One-Time Password
HTOP	HMAC One Time Password
IDM	Identity Management
IOT	Internet of Things
JNI	Java Native interface
LXDE	Lightweight X11 Desktop Environment
OTP	One time password
PADES	PDF Advanced Electronic Signature
PID	Process Identifier
PIN	Personal Identification Number
PKI	Public Key Infrastructure
SCEE	Sistema de Certificação Eletrónica do Estado
SCMD	Serviço Chave Móvel digital
SHA	Sashing Secure Hash Algorithm
SIM	Subscriber identity module
SO	Sistema Operativo

TOTP	Time-based One-time Password algorithm
VM	Virtual Machine
XML	Extensible Markup Language

1. Introdução

A criação do Cartão de Cidadão, documento de Identificação do Cidadão Português, desenvolvido na sequência do projeto de modernização da Administração Pública, unificou num só cartão informação que anteriormente se encontrava difundida em vários documentos, proporcionando novas funcionalidades e informações relativas ao cidadão. O primeiro passo para a criação do Cartão de Cidadão foi dado no ano 1999, com a aprovação do Decreto-Lei n. o 290-D/99, de 2 de Agosto, "dá-se, em Portugal, o primeiro passo no sentido da consagração legal das assinaturas eletrónicas, acolhendo-se, designadamente, as soluções avançadas no quadro da União Europeia, na proposta de diretiva do Parlamento Europeu e do Conselho, relativa a um quadro legal comunitário para as assinaturas eletrónicas."(Anacom, 1999; DRE, 2019). A 5 de fevereiro de 2007 foi publicada a lei Nº 7/2007 que rege a emissão, substituição, utilização e cancelamento do Cartão de Cidadão.

Esta medida fez com que fosse possível incluir a Assinatura Digital como uma política de modernização administrativa, capaz de desenvolver vários pensamentos e doutrinas que ajudam no seu entendimento e levantam questões importantes como a certificação, a segurança e a autenticidade da mesma. Evidentemente que, acompanhada com todas estas novas ideologias, e com a implementação da Assinatura Digital, foi publicada legislação necessária para a regular, a fim de conceder autenticidade e validade legal aos documentos assinados por via eletrónica (DRE, 2003).

Desde o lançamento do Cartão de Cidadão que as assinaturas eletrónicas qualificadas eram possíveis, no entanto, estas eram usadas por um número reduzido de pessoas. Um dos problemas para uma utilização tão reduzida, era a necessidade de ser requerido um leitor de *smartcards* para introduzir o cartão e poucas pessoas estavam devidamente prevenidas.

A 1 de janeiro de 2009 entra em vigor o Decreto-Lei n.º43-A/2008, de 25 de julho e a Portaria n.º701-G/2008 de 29 de julho que consagram a utilização de assinaturas eletrónicas qualificadas e selos temporais, para uso em plataformas eletrónicas de contratação.

É a Lei nº 37/2014, de 26 de Junho "estabelece um sistema alternativo e voluntário de autenticação dos cidadãos nos portais e sítios na Internet da Administração Pública denominado Chave Móvel Digital" com o objetivo de massificar as assinaturas eletrónicas qualificadas.

Em 2015 surgiu um novo meio de autenticação dominado de Chave Móvel Digital[1] como um meio simples e seguro de autenticação dos cidadãos em portais e websites da Administração Pública na Internet, com dois fatores de segurança: uma palavra-chave e um código recebido por SMS.

Com a crescente utilização das Assinaturas Digitais, realizadas através do Cartão de Cidadão e posteriormente através da aplicação chave móvel digital é fundamental garantir a segurança durante o processo de autenticação. Pretende-se com o presente trabalho investigar possíveis vulnerabilidades na aplicação de autenticação do Cartão de Cidadão através da aplicação móvel Autenticação.gov. Pretende-se também efetuar uma análise comparativa à assinatura efetuada através do Cartão de Cidadão comparativamente com a assinatura efetuada através da Chave Móvel Digital.

1.1. Motivação

Assistimos diariamente ao aparecimento de novos modelos de *smartphones* com novas funcionalidades, e cada vez mais avançados.

A massificação dos *smartphones*, a facilidade de estar sempre conectado à Internet, em qualquer momento e lugar levou como já se referiu, ao surgimento de aplicações especificas como a Chave Móvel Digital, com o propósito de generalizar as assinaturas eletrónicas.

É intuito deste projeto estudar e perceber o funcionamento da aplicação Chave Móvel Digital, e acima de tudo verificar se a aplicação é segura estudando as suas vulnerabilidades.

Este estudo incide na aplicação Autenticação.gov para dispositivos móveis com o sistema operativo Android, uma vez que este domina o mercado com percentagens superiores a 73%, como podemos visualizar na figura seguinte.

[1] Chave Móvel Digital -- Lei n.º 37/2014 - https://tinyurl.com/y3qde5wr

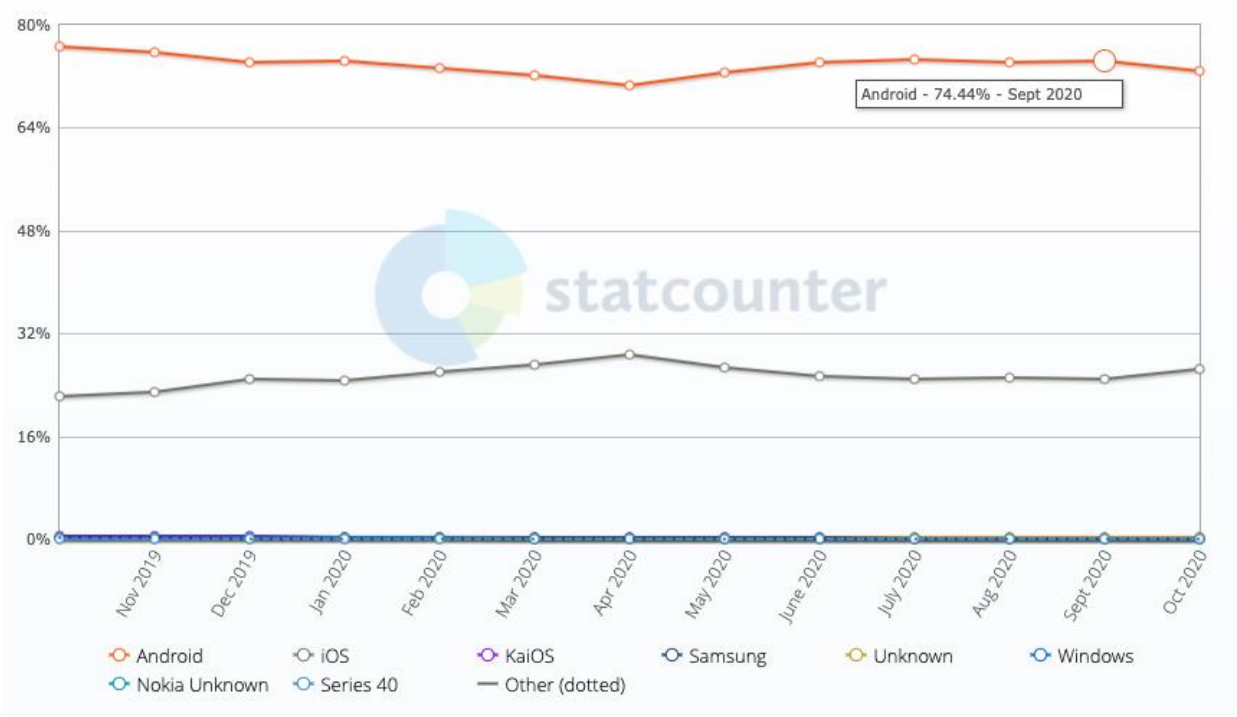


Figura 1-1 - Quota de mercado do SO Android (GlobalStats, 2020)

O valor apresentado na Figura 1-1 pode variar dependo da empresa que efetua o estudo e da metodologia usada, no entanto, em qualquer estudo visualizado o Android domina o mercado com percentagens sempre superiores a 73%.

1.2. Enquadramento

A crescente distribuição de aplicações em áreas sensíveis como a banca, saúde ou serviços públicos não passou despercebida entre os utilizadores que nos seus dispositivos móveis com características computacionais cada vez mais avançadas, juntamente com a facilidade de estar permanentemente conectado à Internet, leva a que estes colecionem as mais diversas aplicações muitas por vezes instaladas e esquecidas.

Perante este cenário e com a necessidade de massificar as assinaturas digitais, e sob a égide da desburocratização e das medidas simplex, o governo disponibilizou uma aplicação para dispositivos móveis chamada Autenticação.gov, regulada pela Portaria[2] n.º 189/2014, de 23 de setembro, publicada no Diário da República, 1º Série, daquela data. Tal Portaria procedeu à "regulamentação necessária ao desenvolvimento da Chave Móvel Digital, aprovada pela Lei

[2] Portaria nº 189/2004 - https://tinyurl.com/y6tklo8c

n.º 37/2014, de 26 de junho, enquanto meio alternativo e voluntário de autenticação dos cidadãos nos portais e sítios na Internet da Administração Pública".

1.3. Objetivos

Com a evolução das Tecnologias o surgimento de novas formas de comunicar, tornou-se imprescindível para a Administração Pública evoluir nos serviços prestados ao Cidadão adequando-os ao uso das tecnologias, promovendo uma Sociedade de Informação, baseada no desenvolvimento tecnológico. Este cenário levou ao surgimento da Chave Móvel Digital com intuito de propagar as assinaturas eletrónicas qualificadas, devido à sua simplicidade de utilização.

Perante o cenário é objetivo deste projeto efetuar um estudo do funcionamento da Chave Móvel Digital, como um meio simples e seguro de autenticação dos cidadãos em portais e websites da Administração Pública.

Pretende-se também realizar uma análise às vulnerabilidades da aplicação móvel autenticação.gov, recorrendo a métodos de análise estática e dinâmica de aplicações, a fim de validar eventuais fragilidades da aplicação. A aplicação autenticação.gov é uma aplicação para dispositivos móveis que permite receber o código de segurança da Chave Móvel Digital por meio de uma notificação, em alternativa ao SMS.

Por último este projeto irá efetuar a comparação da assinatura eletrónica realizada através da Chave Móvel Digital com a assinatura efetuada através do Cartão de Cidadão. Irão ser comparadas as comunicações e alterações registadas no computador no momento da realização da assinatura eletrónica efetuada através do Cartão de Cidadão e da Chave Móvel Digital.

Em Suma os objetivos são:

- Estudar o funcionamento da Chave Móvel Digital;
- Realizar uma análise estática e dinâmica à aplicação móvel para o sistema Android Autenticação.gov;
- Comparar a assinatura efetuada através do Cartão de Cidadão com a assinatura praticada com a Chave Móvel Digital.

1.4. Estrutura do Documento

Este documento encontra-se divido por cinco capítulos, por forma a possibilitar um melhor manuseamento/ compreensão do mesmo.

No presente capítulo são apresentadas as motivações, desafios e abordagem ao presente relatório.

No Capítulo 2 explanamos os métodos de análise de aplicações em dispositivos móveis, assinaturas digitais eletrónicas e por fim autenticação.

No Capítulo 3 abordamos as diversas formas de Autenticação. Abordamos a autenticação de dois fatores e assinaturas digitais.

No Capítulo 4 abordamos tecnologia. Começámos por falar do sistema operativo Android, passando depois a abordar as ferramentas de deteção: sejam de *malware* ou de vulnerabilidades no sistema operativo Android.

O Capítulo 5 terá um teor mais prático, onde se irá utilizar o conhecimento adquirido neste trabalho para efetuar uma análise da aplicação autenticação.gov, assim como efetuar testes sobre a mesma.

Por último, no capítulo 6 são apesentadas as respetivas conclusões da presente dissertação.

2. Trabalho relacionado

Este capítulo descreve os conceitos necessários e essenciais no desenvolvimento deste trabalho enumerando métodos e ferramentas para efetuar uma análise de vulnerabilidades numa aplicação tendo como base o sistema operativo móvel Android.

Nos últimos anos, o mercado de *smartphones* foi alvo de uma grande expansão, de tal forma que estes são utilizados no dia a dia, seja em negócios nas empresas, no governo, hospitais, hotéis ou em família, no espaço de trabalho, como meio de comunicação ou até extensão do mesmo como ferramenta de trabalho (Zoe, 2017). Em 2007 foi lançado o primeiro iPhone pela Apple que foi um marco importante devido às funcionalidades inovadoras que a partir desse momento impulsionaram o mercado dos *smartphones* de forma significativa com novas funcionalidades e com este a possibilidade de instalar as mais diversas aplicações em *smartphones* cada vez mais evoluídos (Montgomery & Mingis, 2020) .

Os *smartphones* passaram a ser o alvo apetecível para os mais variados tipos de ataque. Desde espionagem, vigilância, obtenção de dados sigilosos ou mesmo invasão à rede interna representando inúmeros riscos para as organizações, quando os dispositivo utilizados no trabalho são perdidos ou a sua segurança comprometida, especialmente quando falamos de *smartphones* com o sistema operativo Android, que representam mais de 73% ou mesmo 85% da quota do mercado dependendo da metodologia de medição usada nos diferentes estudos realizados (GlobalStats, 2020; IDC, 2020).

A possibilidade de instalação das mais variadas aplicações levou a que fossem desenvolvidas aplicações em áreas tão especificas como a banca ou mesmo aplicações governamentais. A Figura 2-1 mostra o número de aplicações disponíveis nas principais *stories* em setembro de 2020.

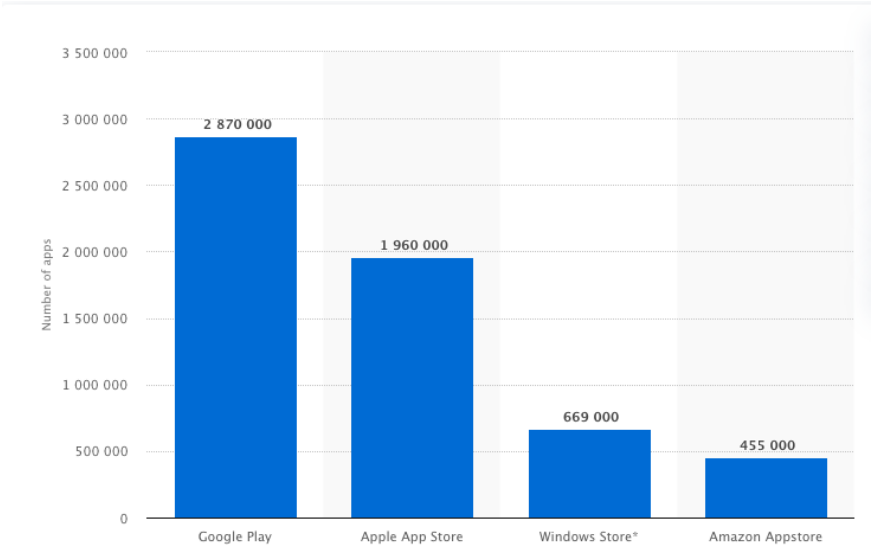


Figura 2-1 - Número de aplicações disponíveis (Statista, 2020)

De acordo com Bilić (2020) o sistema operativo móvel Android é o mais utilizado no mundo, sendo responsável por quotas de mercado superiores 73%. Bilić (2020) também refere que as deteções de código malicioso no Android representam 99% de todo o *malware* em dispositivos móveis, consequência de uma enorme fragmentação do sistema operativo levando a que muitos utilizadores tenham versões antigas instaladas (Bilić, 2020).

Boas notícias são que o número de deteções de *malware* diminuiu 8% no primeiro semestre de 2018 e 10% no segundo semestre do ano de 2019, talvez como resultado dos esforços da Google e dos responsáveis de segurança na deteção de ameaças e evitar sua propagação como podemos visualizar na figura seguinte.

Figura 2-2 - Deteções da *malware* Android (Bilić, 2019)

A análise de uma aplicação recorre a técnicas de estudo que coincidem com as técnicas de análise de *malware*. As duas recorrem a técnicas de análise estática e dinâmica, têm também o mesmo objetivo, ou seja, procuram vulnerabilidades numa aplicação

Para melhor compreender as funcionalidades, os pontos fortes e fracos de cada ferramenta para investigar uma aplicação, quer na procura de vulnerabilidades, quer na análise de *malware* nos dispositivos móveis realizou-se um estudo documental sobre esta temática. Os tipos de trabalho considerados para este estudo foram teses de mestrado, artigos e relatórios técnicos. A Tabela 2-1 apresenta os trabalhos que foram considerados relevantes para a presente dissertação, sendo que o ano de 2017 foi considerado como a data limite da pesquisa. Os trabalhos anteriores ao referente ano não foram considerados.

Para que seja possível escolher a ferramenta a usar para efetuar uma análise nos dispositivos com o sistema operativo Android considerou-se importante comparar várias soluções *open source*.

Como se pode visualizar na tabela seguinte, trabalhos que implementaram soluções *open source* (S. Chen et al., 2018; Fauskrud, 2019; Franek, 2017; López, 2018; Mendonça et al., 2017; Naway & Li, 2019; Ranganath, 2019; Shahriar, 2019; Suciu et al., 2019) que usam diversas ferramentas para efetuar uma análise dinâmica ou estática nos dispositivos moveis Android foram considerados significativos para este estudo.

No levantamento efetuado verificou-se que todos os trabalhos usam diversas ferramentas para elaborar um estudo. Franek, Bc. Peter, elabora mesmo uma tabela onde compara diversas ferramentas para efetuar uma análise estática e outra tabela para efetuar uma análise dinâmica (Franek, 2017).

Nesta investigação salientamos ainda os trabalhos de Chen e de López que efetuam uma análise as aplicações para dispositivos móveis das instituições bancárias dos respetivos países.(S. Chen et al., 2018; López, 2018). Este trabalho é em tudo semelhante ao do presente estudo que pretende implementar um cenário para efetuar uma análise a uma aplicação especifica, a aplicação do Cartão de Cidadão para dispositivos móveis Android.

Tabela 2-1 - Trabalhos relevantes

Ano	Tipo de trabalho	Universidade/ Instituição	Autor	Título	Descrição do trabalho
2019	Relatório		George Suciu, Cristiana Istrate, Ruxandra Raducanu,	Mobile devices forensic platform	Apresenta diversas ferramentas *open-source* e comerciais como por

			Maria Ditu, Octavian Fratu, Alexandru Vulpe,	for malware detection	exemplo: Santoku, MobSF, MobiSec entre outras, para efetuar uma análise forense a um dispositivo móvel.
2019	Artigo	North China Electric Power University, School of Control and Computer Engineering	Nuaway Abdelmonim, Li Yuancheng	Android Malware Detection Using Autoencoder	A implantação da ferramenta MobSF para deteção dinâmica de malware nos dispositivos Android.
2019	Artigo	Hossain Shahriar	Hossain Shahriar	An Exploratory Analysis of Mobile Security Tools	Implementa diversas ferramentas (Cuckoodroid, Flowdroid Droidbox e MobSF) para efetuar uma análise de malware numa aplicação num dispositivo móvel android.
2019	Artigo	Kansas State University	Venkatesh-Prasad Ranganath	Are Free Android App Security Analysis Tools Effective in Detecting Known Vulnerabilities?	Apresenta as vulnerabilidades nos dispositivos móveis Android ao logos das diversas Api e demostra através de diversas ferramentas tais como: QARK, MonSF, Fixdroid entre outras
2018	Artigo	East China Normal University, China New York University Shanghai, China	Sean Clean, Ting SU, Linglinfg Fan, Guozhu Meng, Minhui Xue, Yang Liu, Lihua XU,	Are Mobile Banking Apps Secure? What Can Be Improved?	Apresenta as vulnerabilidades existentes em diversas aplicações bancárias para dispositivos móveis Android. Foram entre outras usadas as ferramentas QARK, MobSF e AndroBugs.
2019	Artigo	Universidade Tiradentes (Unit) Aracaju - SE - Brasil	Mariano Florencio Mendonça, Layse Santos Souza, Isadora Lima do Nascimento, Fabio Gomes Rocha	Automatização de Teste de Segurança Móvel com MobiSec	Apresenta a ferramenta MobiSec fundamentos teóricos e faz pequeno comparativo com a ferramenta MobSF e a ferramenta ApkTool.
2019	Tese de Mestrado	Norwegian University of Science and Technology	Joakim Fauskrud	Hybrid analysis for Android malware family classification in a	Através de técnicas de *Machine learning*, identifica novos tipos de *malware* no sistema operativo Android

				time-aware setting	
2018	Tese de Mestrado	Escuela Politécnica Nacional Facultad de Ingeniería Elétrica y Electrónica	José Luis Palacios López	Análisis de Seguridad de las Aplicationes Móviles basadas en Android, utilizadas en la banca Electrónica del Equador.	Apresenta as vulnerabilidades existentes em diversas aplicações bancárias para dispositivos móveis Android. Foram estre outras usadas as ferramentas MobSF, Dexjar, Jd-Gui entre outras.
2017	Tese de Mestrado	Masaryk University Faculty of Informatics	Bc. Peter Franek	Secure software development process and tools for Android applications	Exibe a arquitetura Android, e os fundamentos básicos de segurança. Compara e avalia diversas ferramentas de análise dinâmica de *malware* nos dispositivos móveis. As ferramentas usadas foram entre outras: Mobsf, Drozer, AppUse

2.1. Análise de *Malware*

Malware é o termo utilizado para definir software cujo objetivo é perturbar ou danificar um computador, dispositivo móvel ou uma rede de computadores (como é o caso dos vírus informáticos) (Costa Marques, 2013; Infopedia, 2018). É uma das principais ameaças das novas tecnologias, sejam elas redes informáticas, de grande ou pequena dimensão, computadores pessoais, interfaces móveis ou mesmo pequenos dispositivos que fazem parte cada vez mais do nosso dia a dia, mais conhecidos por Internet of Things (IoT).

Com o crescente aumento de aplicações para aceder à banca online, muitas mesmo oferecendo estas aplicações com um único meio de acesso, em paralelo o desenvolvimento de *malware* projetado para o roubo de credenciais de acesso ou *pishing* também evoluiu. Exemplo disso é o desenvolvimento do *trojan* bancário Anubis distribuído através da loja online Google Play. Este *trojan* inclui táticas de *ramsomware*[3] podendo o utilizador perder toda a informação no dispositivo móvel(AV-TEST, 2019; Avira, 2020).

[3] Tipo de software nocivo que restringe o acesso ao sistema infetado encriptando os dados de um sistema e que solicita um resgate em cripto moedas para obtenção da informação.

O uso de dispositivo móveis está a aumentar e a empresa Tech Crauch prevê que até final do ano de 2020 existam mais de 6 biliões de utilizadores de *smartphones*. O aumento dos dispositivos móveis nas redes corporativas prende-se com o facto que os mesmos apresentam um nível de eficiência elevado levando a que as empresas adotem medidas de *Bring Your Own Device* (BYOD). A maior utilização dos *smartphones* leva a que se assista a um crescimento de *malware* criado especificamente para dispositivos móveis (ComsumerLab, 2020; kaspersky, 2020). Um relatório de cibersegurança da empresa Check Point menciona que durante o ano de 2019, 27% das empresas foram vitimas de um ciberataque, devido ao facto de a segurança de um dispositivo móvel ter sido comprometida (Inside, 2020).

Embora o *malware* para dispositivos móveis não ocorra em tanta quantidade em comparação com os computadores, tem-se verificado um crescimento acentuado. Os tipos de *malware* mais conhecidos para dispositivos móveis são (kaspersky, 2020):

- *Malware* direcionado a aplicações financeiras;
- *Ransomware* em dispositivos móveis;
- *Spyware* em dispositivos móveis;
- *Malware* por MMS;
- *Adware* em dispositivos móveis;
- Trojan enviado por SMS.

A análise de *malware* é o processo que permite identificar o comportamento de *malware*, o que ele faz, como faz e quais são os seus principais objetivos. A análise de *malware* é um processo complexo. A análise forense, a engenharia reversa, o *disassembling* e o *debugging* são processos morosos. O objetivo da análise de *malware* é o de determinar exatamente o que uma dada amostra de código suspeita pode fazer, como detetá-la na rede e como medir e conter os danos correspondentes (Oktavianto & Muhardianto, 2013).

Depois de se identificarem quais os ficheiros que requerem análise completa, são desenvolvidas assinaturas para detetar infeções de *malware* na rede. A análise de *malware* pode ser usada para desenvolver assinaturas baseadas em hospedeiro e assinaturas de rede.

As assinaturas baseadas em hospedeiro, ou indicadores, são utilizadas para detetar códigos maliciosos nos computadores ou dispositivos móveis vítimas. Estes indicadores geralmente identificam ficheiros criados ou modificados pelo *malware* ou alterações específicas que faz no registo. Ao contrário das assinaturas de antivírus, os indicadores de *malware* concentram-se no

que o *malware* faz num sistema e não nas características do *malware* em si, o que os torna mais eficazes na deteção de *malware* que muda de forma ou que foi excluído do disco rígido.

As assinaturas de rede são usadas para detetar códigos maliciosos, através da monitorização do tráfego de rede. As assinaturas de rede podem ser criadas sem análise de *malware*, mas as assinaturas criadas com a ajuda da análise de *malware* geralmente são muito mais efetivas, oferecendo uma maior taxa de deteção e menos falsos positivos. Após a obtenção das assinaturas, o objetivo final é descobrir exatamente como o *malware* funciona. Esta é muitas vezes a pergunta mais solicitada pela gestão de topo, que quer um relatório completo de uma grande intrusão (Sikorski & Honig, 2012).

São diversas as ferramentas para a análise de *malware*, ferramentas que permitem a análise estática e dinâmica, as quais aplicam diferentes técnicas para perceber as ações efetuadas pelo *malware*. As ferramentas estáticas envolvem o *disassembly* ou descompilação de código. Os antivírus, alguns deles gratuitos, disponibilizam um *scanning* online, no qual podemos fazer um teste em caso de suspeita de *malware* (Gregg, 2015).

2.2. Análise dinâmica de uma aplicação

Existem diversas ferramentas para a análise de uma aplicação, ferramentas estáticas e dinâmicas, as quais aplicam diferentes técnicas para perceber as ações efetuadas pela aplicação (Gregg, 2015). A análise dinâmica têm por base o estudo da aplicação durante a sua atuação, ou seja, analisam o seu comportamento dentro de um sistema controlado, que verifica passo a passo todos os comportamentos e alterações no ambiente provocados pela aplicação (Gregg, 2015).

A análise dinâmica pode ser realizada de forma manual ou automática como podemos verificar na Figura 2-3.

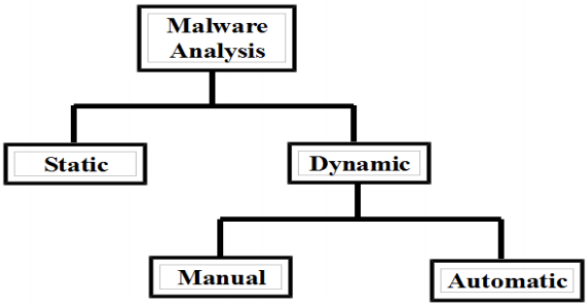


Figura 2-3 - Análise *Malware* (De Andrade et al., 2013)

A análise manual usa um *debugger* para examinar o estado interior de um executável mal-intencionado num ambiente controlado. Na análise dinâmica automática o ficheiro é executado em um ambiente que é dominado *sandbox* que consiste no ambiente isolado e controlado utilizado para execução e monitorização do comportamento de um ficheiro (Paulo & Pinheiro, 2004).

2.3. Análise estática de uma aplicação

A análise estática envolve a análise de uma aplicação sem a executar ao invés da análise dinâmica que envolve a execução da aplicação.

A análise estática básica consiste em examinar o arquivo executável sem visualizar as instruções reais. A análise estática básica pode confirmar se um ficheiro é mal-intencionado, fornecer informações sobre suas funcionalidades e às vezes fornecer informações que lhe permitirão produzir assinaturas de rede simples. A análise estática básica é direta e pode ser rápida, mas é ineficaz em grande parte contra o *malware* sofisticado, podendo perder comportamentos importantes.

A análise estática avançada consiste em engenharia reversa do *malware*, carregando o executável num *disassembler* e observando as instruções do programa para descobrir o que o programa faz. As instruções são executadas pela CPU, pelo que a análise estática avançada diz exatamente o que o programa faz. No entanto, a análise estática avançada possui uma curva de aprendizagem mais acentuada do que a análise estática básica e requer conhecimentos especializados de *disassembly*, elaboração de código e conceitos de sistemas operativos (Panda, 2018).

A análise híbrida é um outro tipo de análise de *malware* que resulta da combinação da análise estática e dinâmica (Fauskrud, 2019).

2.4. Desafio de análise de uma aplicação

Analisar uma aplicação irá permitir saber o seu comportamento, o que faz, como faz e quais os seus objetivos.

Para realizar análises estáticas, é necessário um grande conhecimento em programação, nomeadamente a linguagem *assembly x86*. Durante o processo de análise estática, não é

necessário executar a aplicação. Geralmente, o código-fonte não está disponível. Primeiro é preciso fazer o *disassembling* e descompilar do código. Depois de realizar a engenharia reversa, é possível analisar o código em linguagem de baixo nível.

A maioria dos analistas efetua uma análise estática numa primeira fase porque é mais segura do que a análise dinâmica. O desafio na análise estática é a complexidade no *malware* moderno, onde em alguns casos são implementados sistemas *anti-debugging* para dificultar ou impossibilitar a análise do código (Oktavianto & Muhardianto, 2013).

Por sua vez a análise dinâmica possui limitações, como o tempo de execução de um *sandbox* é limitado a alguns minutos, pode-se deixar passar ações de *malwares* que só seriam executadas após certo tempo ou apenas após a reinicialização do sistema.

3. Autenticação

Não muito longínquo vai o tempo que o telemóvel servia apenas para efetuar e receber chamadas e SMS. Nos nossos dias, para além da realização de tarefas básicas exigidas a um telemóvel, este permite instalar as mais diversas aplicações, andamos sempre conectados à Internet e aceder a informação privilegiada.

Embora o conceito de palavra passe tenha prevalecido em muitas disciplinas e domínios de tecnologia, o público em geral tinha pouco conhecimento sobre isso, com exceção dos PINs para seus cartões de bancários.

Foi a Internet que introduziu o conceito de palavra passe para maioria deles. Desde então as pessoas perceberam que precisavam se lembrar das suas palavras passe para aceder as suas contas de e-mail, sites de comércio eletrónico, sites bancários entre outros.

Se há uns anos atrás, uma palavra passe era suficiente para desbloquear uma conta e requisitos de complexidade de palavra passe era algo muito vago.

O cenário da cibersegurança não era nem de longe tão desafiador como é agora, especialmente quando se trata da segurança de contas de consumidor. Houve exceções para certos setores, como banca , onde os requisitos de palavra passe eram um pouco mais rígidos e de uma forma oculta de autenticação de dois fatores (Piqueras Roger, 2020).

Nos dias que decorrem o telemóvel passou a designar-se *smartphone*[4], com todas as vantagens que este engloba desde a facilidade em instalar aplicações ao facto de estar permanentemente conectados à Internet, os *smartphones* são considerados dispositivos perfeitos para complemento da autenticação (utilizador e palavra passe), quer através de um SMS quer através uma aplicação própria para o efeito.

3.1. Autenticação de dois fatores

A autenticação de dois fatores é um recurso oferecido por vários prestadores de serviços online que acrescentam uma camada adicional de segurança para o processo de login da conta, exigindo que o utilizador forneça duas formas de autenticação.

[4] Smartphone – palavra inglesa que significa telefone inteligente

A primeira forma – em geral – é a sua senha. O segundo fator pode ser qualquer coisa, dependendo do serviço. O caso mais comum, é um SMS, um código que é enviado para um e-mail ou um código gerado por uma aplicação instalada num *smartphone*.

A teoria geral por detrás da autenticação de dois fatores é que para efetuar login, será necessário saber e possuir algo mais.

A autenticação de dois fatores não é um método infalível, no entanto, é uma excelente barreira para prevenir a intrusão indesejada. É de conhecimento público que as palavras passe fracas são fáceis de lembrar, por outro lado, as fortes que podem ser difíceis de adivinhar, mas também difíceis de lembrar. Devido a isso, as pessoas utilizam as mesmas palavravas passe para todos os seus acessos.

Nesse sentido, a autenticação de dois fatores, pelo menos, faz com que um individuo mal intencionado não tenha apenas que descobrir sua autenticação, como também ter acesso a um segundo fator, muito mais difícil de conseguir (Donohue, 2014).

Ativar a autenticação de dois fatores para contas online é então fundamental para que exista um incremento substancial da segurança (Piqueras Roger, 2020).

O uso de *One Time Password* (OTP) enviados através de SMS como segundo fator de autenticação é um facto recorrente, no entanto, o uso deste deve ser ponderado apenas para algum tipo de contas. Contas que contenham dados sensíveis o uso de SMS como segundo fator de autenticação não deverá ser equacionado (Piqueras Roger, 2020; Sean, 2019).

Os OTP de autenticação baseados em SMS são opções populares para proteger o mais diverso tipo de contas, e estas encontram-se mais seguras do que apenas o uso de autenticação simples. Contudo, a segurança da rede móvel, indica que o SMS não é um método seguro de comunicação. Desde antenas instaladas de modo a capturar as comunicações a ataques mais sofisticados existem os mais diversificados métodos conhecidos para espionar e visualizar os SMS (Piqueras Roger, 2020).

Das deficiências de segurança da rede móvel ao método de SIM *swapping* o uso de *tokens* através de SMS deverá ser equacionado para contas que contenham dados sensíveis tais como dados pessoais ou financeiros (Piqueras Roger, 2020).

Como alternativa ao envio do *token* através de SMS, podemos configurar quando possível, a criação de um *token* através de uma aplicação instalada no *smartphone*. Este método é mais

seguro devendo ser usado para contas que contenham dados sensíveis. As vantagens e desvantagens do uso de SMS em comparação com aplicação instalada no *smartphone* que gera um *token* estão resumidos na Tabela 3-1.

Tabela 3-1 - *Tokens* Aplicação VS SMS adaptado de (Piqueras Roger, 2020)

Tipo Autenticação	Vantagens	Desvantagens
Aplicação Móvel	* Útil * Não requer ligação à rede * A mesma aplicação pode gerar tokens para várias contas	* Crítica para gerar e manter códigos de segurança * Perda ou roubo do telefone * Critico para gerar codigos de backup do *smatphone*
SMS	* Não está vinculado à criptografia * Mais fácil de recuperar em caso de perda ou roubo do dispositivo	* Requer ligação de rede * Geralmente inseguro

Apesar da sua conveniência e uso por um grande número de serviços, a autenticação de dois fatores com envio de *token* via SMS tem desafios de segurança significativos. Os desafios de segurança do uso de SMS para entrega de *token* de autenticação de dois fatores variam desde as ameaças um tanto sofisticadas aos protocolos da rede móvel, que exigem que um agente mal intencionado esteja na proximidade da vítima-alvo, a técnicas fáceis de usar que, apesar de ser muito menos complexo tecnicamente, não tem restrições de alcance e podem ser implementados com custos muito baixos como por exemplo a troca de SIM (Piqueras Roger, 2020).

Numa perspetiva global não sendo foco deste projeto detalhar especificamente as fragilidades da rede móvel, a Figura 3-1 apresenta os desafios de três métodos diferentes de interseção de uma mensagem de SMS.

METHOD OF SMS INTERCEPTION	ADVANTAGES	DIFFICULTY	COST
Over the air	• Fast. • Does not require any time-consuming preparation steps.	• Attacker must be in the vicinity of the victim. • Technical knowledge of GSM, SW-radios and SW-based network stack.	• Low.
SS7	• Can be done remotely. • No need to interact with the operator or leave any trail. • Fast. • Does not require any time-consuming preparation steps.	• Requires access to SS7 nodes and knowledge of SS7 protocols.	• High. • For-sale access to SS7 nodes on the dark web.
SIM swap	• Can be done remotely. • Low cost. • Low-hanging fruit.	• Requires attacker to interact with network operator's customer support, which can be a time-consuming preparation stage. • Leaves a trace. • Some operators in African nations have mitigated this method.	• Low.

Figura 3-1 - Métodos interseção SMS (Jover, 2020)

3.2. SIM swap scam

SIM Swap scam também conhecido como SIM *swapping*, em português fraude de troca de cartão *subscriber identity module* - SIM, é um golpe astuto de custo relativamente baixo em que os agentes maliciosos sequestram o número de telefone da vítima com a finalidade de obter a One *Time Password*. Este conclui a operação chamada de autenticação de dois fatores (H. Chen et al., 2010).

A autenticação de fatores adiciona um nível extra de segurança, através de dados biométricos, uma mensagem de texto, um email ou notificação numa aplicação em conjunto com as credenciais nome de utilizador e palavra passe. Assim é garantida uma autenticação adicional ao utilizador e dificultando a ação a pessoas com fins maliciosos (Russo, 2019).

Apesar da eficácia da interceção de SMS como já foi dito anteriormente a fraude da troca de SIM é indiscutivelmente uma ameaça de segurança contra comunicações SMS sendo esta responsável em entregar *tokens* únicos para autenticação.

A Figura 3-2 mostra o exemplo de um ataque de troca de SIM que consiste em enganar o operador da rede móvel, através de uma chamada para o atendimento ao cliente declarando ser o proprietário do telefone necessitando que a conta seja transferida para o novo cartão SIM, alegando a perda do telefone, mas que necessita de acesso assim que o recupere.

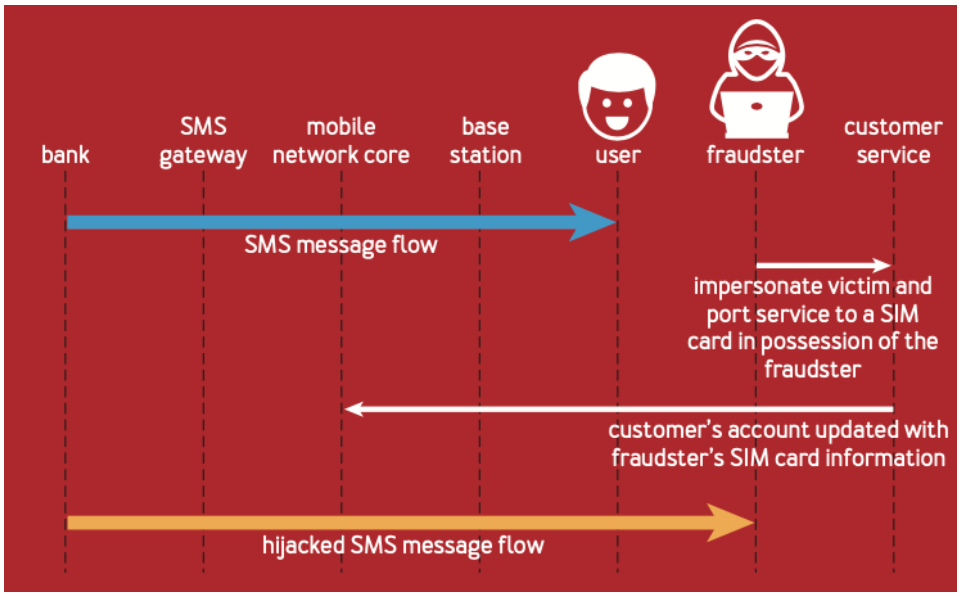


Figura 3-2 - SIM SWAP ATTACK (Piqueras Roger, 2020)

Tendo o invasor conseguido obter acesso à conta transferida para um novo SIM, o resto do ataque é bastante simples. A partir daquele momento, até que a vítima perceba da perda de cobertura de rede móvel e ligar para o atendimento ao cliente, o invasor irá ser o destino de qualquer chamada e mensagem de texto.

Este tipo fraude de troca SIM não são muito complexos de mitigar. Apesar de serem um risco de segurança amplamente reconhecido na América e Europa, esta é uma ameaça que já está a ser atenuada em África (Greenberg, 2019; Piqueras Roger, 2020).

No caso em estudo da aplicação Autenticação.gov o utilizador para iniciar tem que introduzir as suas credencias e de seguida introduzir um código que é enviado por SMS. A Figura 3-3 exemplifica o processo de autenticação na aplicação (número de telemóvel e PIN) mais código de segurança enviado através de SMS.

É na fase de receção da mensagem que se encontra a fragilidade da autenticação, pois esta pode ser vítima de falsificação através do método de fraude de troca de SIM.

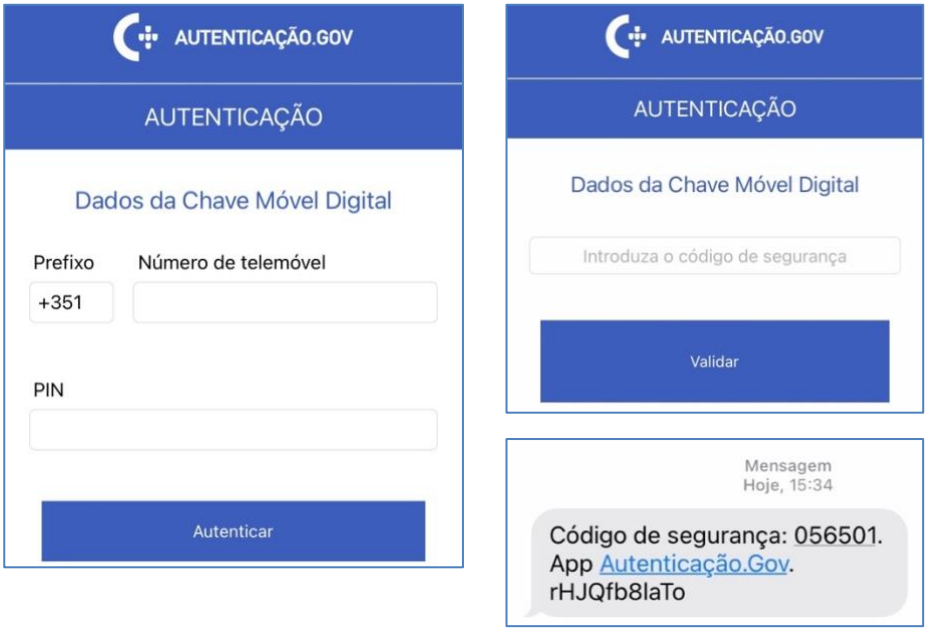


Figura 3-3 – Autenticação.GOV fazes de autenticação

3.3. Assinatura Digital Qualificada

O termo certificado digital qualificado surgiu devido à necessidade na *European Union* (EU) de se definir um padrão perante os sistemas de certificação. Este tipo de certificados são regidos pelas normativas Europeias definidas na diretiva relativa a *eSignatures*[5] no entanto a sua geração é geralmente adaptada às necessidades legais de cada país.

A assinatura digital qualificada de documentos é realizada recorrendo a certificados digitais qualificados. Em Portugal estes certificados são emitidos por entidades certificadas como o Sistema de Certificação Eletrónica do Estado (SCEE), existindo já diversas entidades do Estado que servem este propósito e também algumas do sector privado. Apenas a assinatura digital qualificada tem um efeito legal equivalente à assinatura manuscrita (Almeida, 2009).

O Decreto-Lei n. 290-D/99, de 2 de Agosto aprova o regime jurídico dos documentos eletrónicos e da assinatura digital tendo sido alterado e republicado pelo Decreto-Lei n.º 88/2009, de 9deAbril (que altera e república também o Decreto-Lei n.º 116-A/2006, de 16 de Junho) (Anacom, 1999).

[5] Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures
https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31999L0093:en:HTML

3.4. Assinaturas Digitais eletrónicas

A assinatura digital eletrónica consiste num software criptográfico utilizado para identificar de forma inequívoca o titular que assinou o documento. A mesma garante integridade do documento e autoria do titular.

As assinaturas digitais recorrem a algoritmos de cifra assimétrica também chamados de chave pública durante o processo de criação de códigos de integridade e autenticidade. Não sendo foco deste projeto detalhar cifras simétricas ou assimétricas é importante saber o seu uso na criação de uma assinatura digital.

Desta forma a assinatura digital resulta igualmente da encriptação no emissor, do resumo digital da mensagem a transmitir, mas recorrendo à chave privada do par de chaves utilizado pelo criador. Desta forma a assinatura digital permite para além de assegurar a integridade da mensagem, autenticar o emissor e garantir o não repudio do seu envio.

Quanto à confidencialidade caso seja desejável, as aplicações de segurança que recorrem às assinaturas digitais utilizam normalmente encriptação simétrica. O uso conjunto de criptografia simétrica e assimétrica fica a dever-se ao facto de a criptografia simétrica ser bastante mais eficiente (Granjal, 2017).

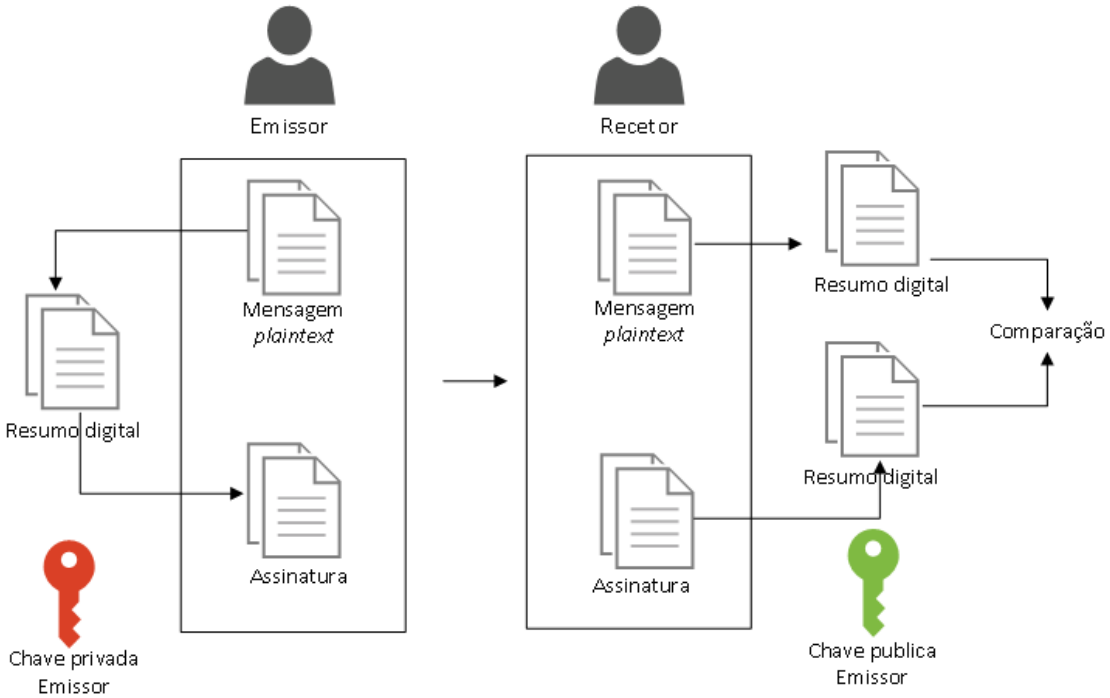


Figura 3-4 - Assinaturas digitais adaptado de (Granjal, 2017)

A Figura 3-4, exibe a utilização de chaves públicas na criação de assinaturas digitais, com o objetivo de proteger uma mensagem trocada entre duas entidades. A assinatura digital é criada pelo emissor recorrendo à sua chave privada do seu par de chaves. A mesma assinatura pode ser verificada pelo recetor recorrendo à chave pública do emissor. Essa chave permite desencriptar o código de integridade recebido, que o recetor compara com o resumo digital calculado a partir da mensagem recebida. Caso os dois resumos sejam iguais, a mensagem é considerada integra e o seu emissor autenticado.

3.5. PDF Advanced Electronic Signature (PadES)

Com a evolução dos ataques de *malware* foi necessário encontrar novos mecanismos para mitigar a fraude eletrónica fomentando o aparecimento de novos formatos de assinaturas eletrónicas tais como XML *Signatures*, XadES, CadES e PadES. Devido ao âmbito deste projeto iremos descrever o PadES, pois este integra as capacidades dos XadES e CadES, diferindo na medida em que se aplica apenas a documentos em formato PDF.

PDF Advanced Electronic Signature (PadES) é um formato de assinatura eletrónica Europeu criado pelo *European Telecommunications Standards Institute* (ETSI), com o objetivo de satisfazer os requisitos as diretivas impostas pela União Europeia, a Diretiva Europeia 1999/03/EC (Parlamento, 2000).

O formato PadES possibilita assinar um documento PDF com um carimbo validando a assinatura num período temporal específico dominado de "*timestamping"*(Molnar & Ko, 2016). A validação cronológica "*timestamping"* é um processo seguro que permite provar a temporalidade de um documento, isto é, a existência de certos dados antes de uma determinada data. Á medida que os certificados são revogados devido à perca ou perda de validade, as assinaturas efetuadas não podem de repente tornar-se invalidas. De forma a garantir a validade das assinaturas a longo prazo é necessário recorrer a um serviço de *timestamping* confiável (Molnar & Ko, 2016; Pohlmann et al., 2010).

A figura seguinte exibe um exemplo de uma assinatura PadES num documento PDF. Visualizamos a inclusão de dados adicionados no final do documento PDF um *Document Security Store* (DSS) que contem por exemplo certificados e informações sobre a validade do certificado. De seguida é adicionado um *timestamp* com uma data de validade 2009. Um segundo DSS e respetivo *timestamp* é adicionada ao PDF expandindo a vida útil do documento para 2015.

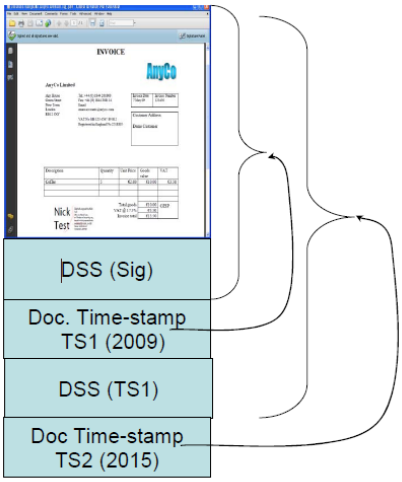


Figura 3-5 - Documento PDF com *timestamp* (ETSI EN 319 142-1, 2016; Pohlmann et al., 2010)

O ETSI define também 4 níveis base de assinatura de modo a abranger o ciclo de vida da assinatura PadES (ETSI EN 319 142-1, 2016; Turner, 2017):

1. Nível B-B - Define um perfil para assinaturas eletrónicas de curto prazo. Deve incluir uma assinatura eletrónica e o certificado de assinatura.
2. Nível B-T- adiciona um carimbo de data e hora, respetivamente, uma marca de tempo que prova que a assinatura existia em uma determinada data e hora.
3. Nível B-LT- Possibilita que a assinatura de um determinado documento possa ser validada, mesmo depois de um longo período de tempo, quando o ambiente de assinatura (por exemplo, CA) não está disponível. O nível LT é recomendado para assinaturas eletrónicas avançadas; no entanto, as leis nacionais devem ser verificadas caso a caso.
4. Nível B-LTA – Este nível fornece requisitos para a incorporação de carimbos de tempo eletrónicos que permitem a validação da assinatura durante muito tempo após esta ter sido concebida. Este nível visa lidar com a disponibilidade e integridade de longo prazo do material de validação.

3.6. Tokens

Provarmos a nossa identidade para que nos possamos autenticar e obter acesso a algum tipo de sistema é um desafio maior do que a maioria dos utilizadores imaginam.

Esse processo deve ser planeado de forma a que, por um lado, seja o mais fácil possível para o utilizador do sistema obter acesso, enquanto, por outro lado, seja o mais difícil possível para

alguém que não está autorizado ter acesso ao sistema. Neste paradigma os *tokens* encaixam assim como parte do processo de autenticação.

Um sistema simples de utilizador e palavra passe não é um sistema seguro para proteção de dados considerados sensíveis. Como alternativa temos a já descrita autenticação de dois fatores.

Existem três fatores independentes para uma autenticação (Rahav, 2020) como podemos visualizar na Figura 3-6.

- Algo que o utilizador sabe: uma password ou pin;
- Algo que o utilizador "tem": um *token* ou dispositivo móvel;
- Algo que utilizador de facto é: dados biométricos por ex.

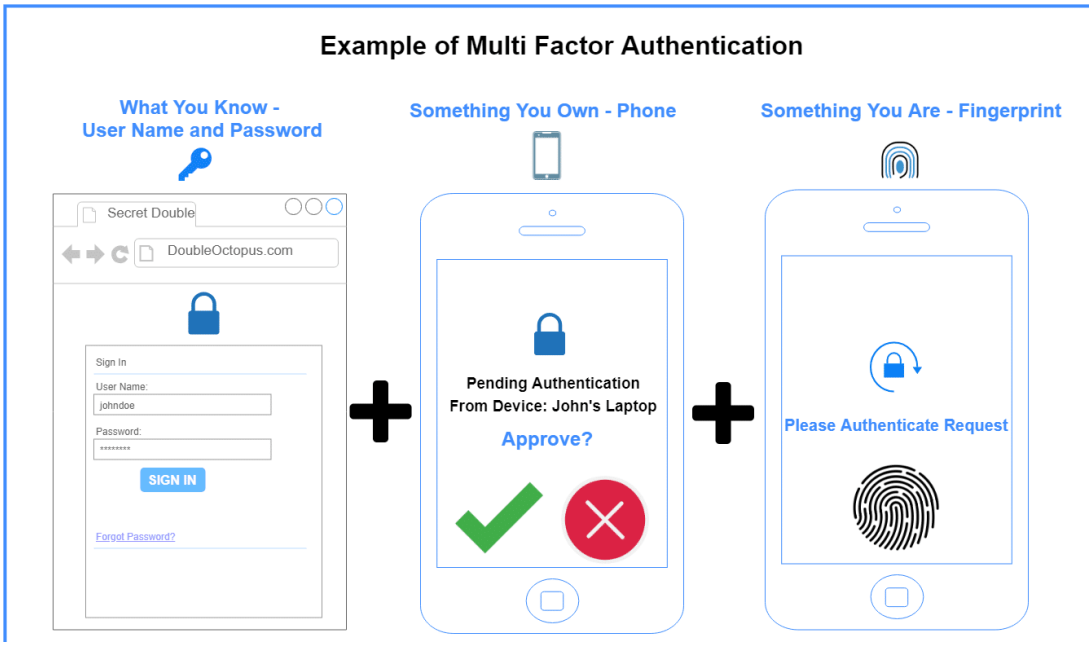


Figura 3-6 - Três fatores para a autenticação (Rahav, 2020)

Com a autenticação de dois fatores, o utilizador para se autenticar deve usar pelo menos dois destes fatores independentes. Isso introduz o conceito de *token*, que é usado para provar um dos dois fatores independentes exigidos acima.

Existem dois tipos de *token*.

- Os *token* de hardware, são dispositivos físicos que são usados para obter acesso a um determinado recurso.
- Os *token* de software, não são fisicamente alcançáveis, mas que existem como software ou aplicações em dispositivos móveis ou computadores.

Hoje em dia ao invés de se transportar hardware extra e dado ao facto de que o uso de *smartphone* está generalizado, os *tokens* via software superam os de hardware. A instalação destes é fácil, bastando aceder à respetiva *store*.

3.7. One time Password – OTP

One time Password (OTP) é uma forma de autenticação multi-fator. A autenticação multi-fator exige credencias adicionas para alem do uso de autenticação (utilizador palavra passe) para que se possa obter acesso a determinado sistema como por exemplo um SMS com um código.

OTP são senhas descartáveis geradas a partir de um "*seed*" previamente partilhado. O processo de geração de OTPs deve possuir duas entidades: uma geradora que poderá ser um dispositivo móvel com uma aplicação para o efeito e um servidor de verificação.

Existem dois tipos de OTP

1. HOTP - HMAC-based One-Time Password
2. TOTP Time-based One-time Password

3.8. HMAC-based One-Time Password - HOTP

HMAC- based One-Time Password (HOTP) também conhecido por OTP baseado num evento, em que o fator de movimentação em cada código é baseado num contador.

Cada vez que o HOTP é solicitado e validado, o fator de movimento é incrementado com base num contador. O código gerado é válido até que seja efetuado o pedido de um novo código e validado pelo servidor de autenticação. O gerador OTP e o servidor são sincronizados sempre que o código é validado e o utilizador obtenha acesso (MICROCOSM, 2018; Onelogin, 2020).

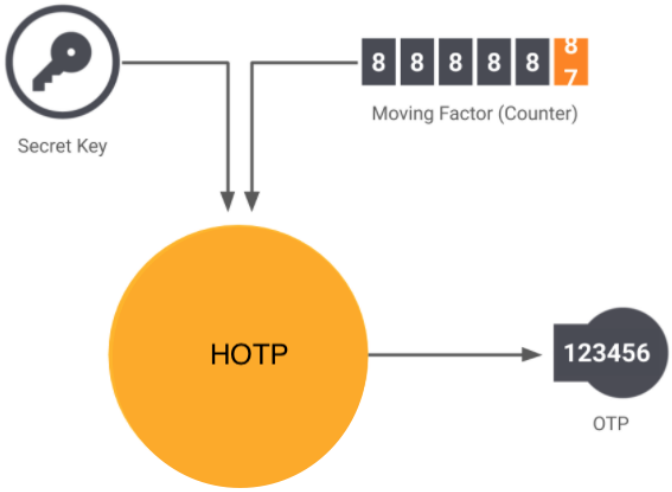


Figura 3-7 - HOTP funcionamento adaptado de (Onelogin, 2020)

3.9. Time-based One-time Password algorithm -TOTP

O algoritmo *The Time based One Time Password* (TOTP) é uma variante do algoritmo *HMAC One Time Password* (HTOP).

O TOTP é uma senha temporária gerada por um algoritmo para uso de autenticação de dois fatores de modo a ter acesso à aplicação ou site pretendido. Quando um TOTP é recebido através de uma aplicação instalada no dispositivo esse TOTP tem um período de validade a que se chama de iteração.

O TOTP é um método geralmente encontrado em mecanismos de autenticação multi-fator onde é solicitado uma credencial temporária e dinâmica. O TOTP usa uma função *hash*, por exemplo o HMAC-SHA512, para gerar um valor pseudoaleatório (Junior, 2019; Mengato et al., 2019; Plata & Calpito, 2020). A duração padrão do TOTP é de 30 segundos eliminado assim o risco de roubo do código. É quase impossível prever a próximo código gerado devido as características de mudança rápida (Plata & Calpito, 2020). A Figura 3-8 ilustra o exemplo do funcionamento de um TOTP.

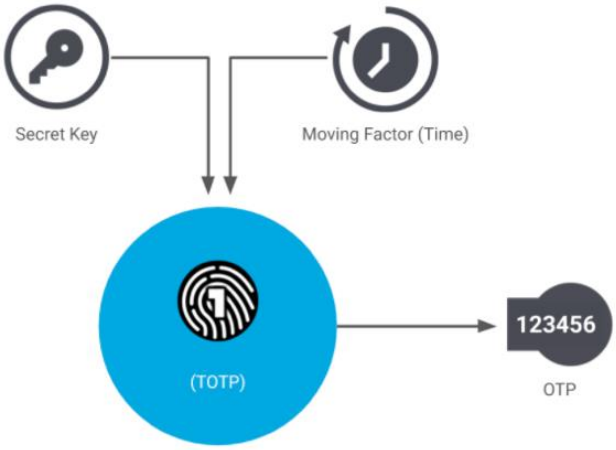


Figura 3-8 - TOTP exemplo adaptado de (Onelogin, 2020)

A Figura 3-9 exibe receção de código OTP através da aplicação móvel Autenticação.gov. O círculo a volta do código de segurança corresponde a uma iteração que representa o tempo de validade do OTP ou código de segurança.

Figura 3-9 - Receção de um OTP através da aplicação Autenticação.gov

Comparação do algoritmo HOTP com o algoritmo TOTP

Ambos os algoritmos OTP oferecem códigos de utilização única, no entanto, a principal diferença é que no HOTP um determinado OTP é válido até que seja usado ou até que um OTP seguinte seja usado. No HOTP, há vários códigos válidos de "próxima OTP". Isso ocorre porque o botão no *token* pode ser pressionado, incrementando o contador no *token*, sem que a OTP resultante seja enviada ao servidor de validação. Por este motivo, os servidores de

validação HOTP aceitam um conjunto de OTP, isto é, eles aceitam um OTP gerado por um contador que esteja dentro de um número definido de incrementos do valor do contador anterior armazenado no servidor. Esse intervalo é conhecido como janela de validação. Se o contador de *tokens* estiver fora do intervalo permitido pelo servidor, a validação falhará e o *token* deverá ser sincronizado novamente.

No TOTP é apenas valido um OTP num determinado momento. Numa perspetiva de segurança o algoritmo TOTP é claramente superior não estando tão suscetível a ataques de força bruta como o HOTP (MICROCOSM, 2018; Onelogin, 2020).

4. Tecnologias

Atualmente o sistema operativo móvel Android, desenvolvido pela Google domina o mercado, tornando-se o sistema operativo mais popular a nível mundial. Em 2018 a quota de mercado de smartphones com sistema operativo Android foi de 87% segundo o relatório da Gartner (Gartner, 2019).

Assim sendo, nesta secção descrevemos o sistema operativo móvel Android assim como enumeramos as ferramentas para deteção de vulnerabilidades no Android.

4.1. Sistema Operativo Android

Android é um sistema operativo de código aberto baseado em Linux, constituído por *middleware* e aplicações e é desenvolvido para uma vasta variedade de dispositivos (Linuxforyou, 2008). Foi desenvolvido inicialmente pelo Android Inc que foi comprado pela Google em 2005. A primeiro verão Android foi lançada em 2007 (Franek, 2017; Linuxforyou, 2008).

A arquitetura da plataforma Android pode ser separada nas seguintes camadas (Franek, 2017; Linuxforyou, 2008):

- **Linux *Kernel***: O uso do kernel Linux permite que a plataforma Android tire proveito de vários recursos como segurança, gestão de memória, gestão de processos, pilha de rede e um modelo de driver.
- **Hardware abstration layer**: Fornece uma interface padrão que permite aceder aos recursos de hardware numa *Framework* de alto nível. Consiste em várias bibliotecas, fornecendo uma interface para um componente de hardware específico, por exemplo, uma câmara.
- **Android runtime (ART)**: desde a versão 5.0 (API 21), que cada aplicação é executada através de um processo e instância distinto. ART é gravado para ser executado em dispositivos com pouca memória através de múltiplas máquinas virtuais sendo executado através de ficheiros *Dalvik Executables* (DEX). Estes fornecem como recursos principais a compilação *just-in-time (JIT)*, recolha de lixo otimizada ou melhor suporte à depuração com funcionalidade dedicada. No JIT a compilação ocorre antes da execução da aplicação o que o vai diferenciar do DEX. Com isso,

existe um aumento de velocidade de execução de até duas vezes em relação ao DEX, reduzindo-se a perceção de atraso.

- **Native C/C++ libraries:** Muitos componentes principais são construídos a partir de código nativo que requer bibliotecas nativas escritas em C e C ++. Usando o desenvolvimento do Android NDK pode-se aceder algumas dessas bibliotecas e escrever o próprio código C e C++.
- **Java API Framework:** Todo o conjunto de recursos do sistema operativo Android está disponível através de APIs desenvolvidas na linguagem de programação Java. Estas API englobam os componentes necessários para criar aplicações Android.
- **System applications:** São um conjunto de aplicações que permitem efetuar funcionalidades básicas tais como o envio de SMS, os contactos ou o calendário. Estas aplicações vêm incorporadas no próprio sistema operativo Android não podendo ser apagadas.

A Figura 4-1 mostra as principais camadas do sistema operativo Android (Android, 2019a; Franek, 2017)

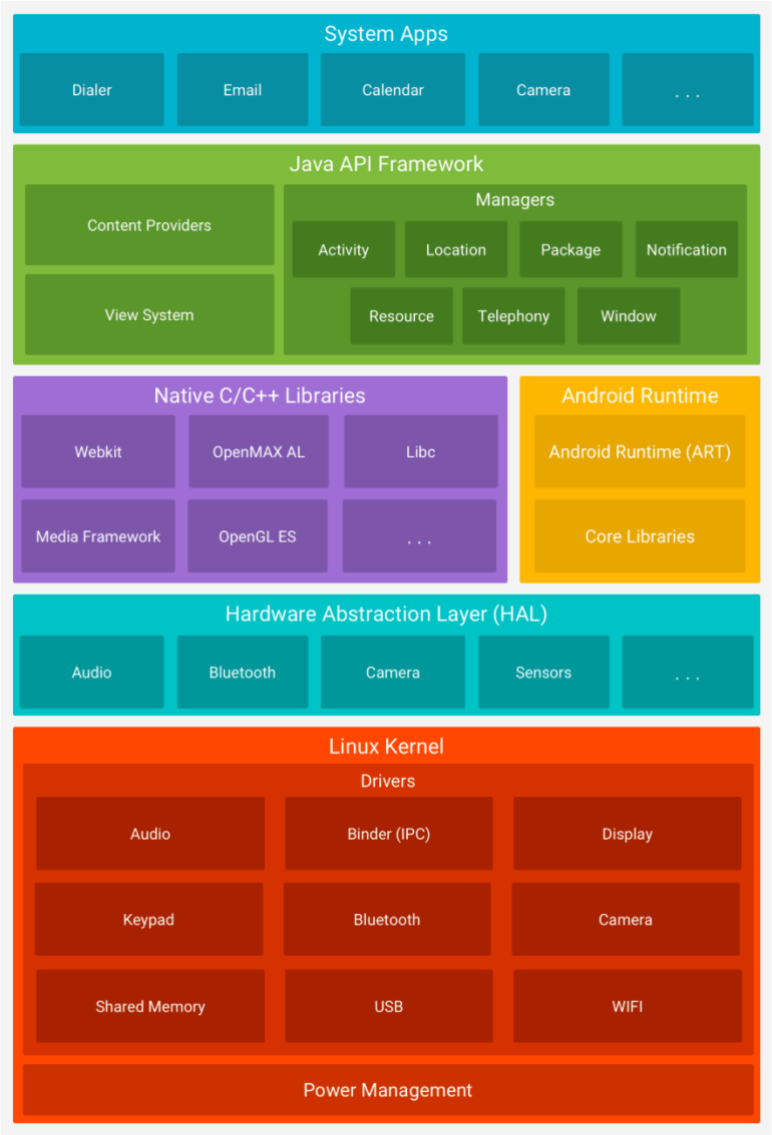


Figura 4-1 - Camadas do SO Android

4.2. Android - Aplicações

As aplicações Android são desenvolvidas nas linguagens de programação Kotlin, Java e / ou C++. A interface de programação de aplicativos (API) é utilizada para aceder a funcionalidade do Sistema Operativo (SO) Android. O código-fonte, os recursos e os dados do aplicativo são compilados num ficheiro Android Package (APK) usando o Android-SDK. Os ficheiros APK são usados para distribuir e instalar as aplicações Android (Android, 2019c).

O sistema operativo Android é um sistema operativo Linux multiutilizador, em que uma aplicação poderá ser executada por mais que um utilizador. O SO Android define permissões em todos os ficheiros de uma aplicação, de forma que os ficheiros possam ser acedidos apenas por essa aplicação específica. Cada processo possui sua própria máquina virtual (VM) e o

código da aplicação é isolado de outras aplicações quando executado. O princípio do privilégio mínimo é implementado no SO Android. Uma aplicação pode aceder apenas a recursos do sistema absolutamente essenciais para execução do seu trabalho. A aplicação deve solicitar permissões específicas ao utilizador do dispositivo Android, para obter acesso a recursos adicionais do sistema (Android, 2019c; Fauskrud, 2019).

As aplicações Android são criadas em quatro componentes:

- Atividades,
- Serviços,
- Recetores de transmissão,
- Provedores de conteúdo.

Cada componente tem uma finalidade distinta e tem um ciclo de vida específico que define a forma como o componente é criado e destruído.

O componentes **atividades** é a janela de uma aplicação que fornece uma interface de utilizador, e é o componente com o qual o utilizador interage.

O componente **serviços** é um ponto de entrada para manter um aplicativo em execução em segundo plano, seja qual for o motivo.

Existem duas subcategorias de serviços:

- serviços iniciados
- serviços vinculados.

Os serviços iniciados são usados quando uma aplicação possui algum trabalho inacabado e precisa continuar funcionando até que seja concluído.

Serviços vinculados fornece uma API para outros processos e é executada quando o sistema ou outra aplicação precisar.

O componente **recetores de transmissão**, são o componente que permite que uma aplicação receba transmissões em todo o sistema. As transmissões são recebidas pelo recetor de transmissão, mesmo que a aplicação não esteja sendo executada no momento, que permite que a aplicação inicie e execute alguma ação. As transmissões podem ser iniciadas pelas aplicações e pelo sistema e é usado principalmente como um gateway para poder comunicar com outros componentes.

O componente **provedores de conteúdo** gere um conjunto compartilhado de dados de aplicações que são armazenados num local acessível pelo aplicativo. Os dados da aplicação podem, por exemplo, ser armazenados no sistema de ficheiros (Android, 2019c), base de dados ou na nuvem. O provedor de conteúdo permite que uma aplicação publique itens de dados denominados usando um esquema de URI. A aplicação mapeia itens de dados para o espaço de nome do URI, e outras entidades podem aceder os itens de dados usando esses URIs. Como exemplo, o SO Android gere um provedor de conteúdo para compartilhar o acesso às informações de contacto do dispositivo. A aplicação pode aceder a informações de contato através do provedor de conteúdo, desde que ele tenha a permissão adequada para fazê-lo. Provedores de conteúdo também pode ser usado para ler / gravar dados privados do aplicativo (Android, 2019c; Fauskrud, 2019).

Atividades, serviços e recetores de transmissão são ativados enviando um *Intent* (uma mensagem assíncrona). Como as aplicações são executados em ambientes isolados, eles devem enviar um *Intent* para informar o sistema para iniciar um componente específico. Uma aplicação pode enviar uma intenção de começar por si próprio componente ou o componente de uma outra aplicação. Os provedores de conteúdo são ativados por algo chamado resolvedores de conteúdo (Android, 2019c; Fauskrud, 2019).

Como foi citado anteriormente as aplicações Android são instaladas através de ficheiros APK. Os ficheiros APK contêm o código fonte compilado, recursos e dados. A estrutura típica de um ficheiro APK é demonstrada na Figura 4-2

```
Application.APK/
    assets/
    lib/
        armeabi/
            libfoo.so
        x86/
            libfoo.so
    META-INF/
    res/
    AndroidManifest.xml
    classes.dex
    resources.arsc
```

Figura 4-2 - Ficheiro APK estrutura

A pasta *assets* contém os ficheiros que a aplicação pode recuperar usando a API do *AssetManager* (Tam et al., 2017).

A pasta *lib* contém os ficheiros nativos do código, ou seja, código C/C++ compilado para ABIs[6] especificas. A Java Native Interface (JNI) é usada para chamar funções nos ficheiros de código nativo (Android, 2019a).

A pasta META-INF contém os certificados dos ficheiros e os Hash - SHA1 de todos os ficheiros no APK (Tam et al., 2017).

A pasta res contém recursos das aplicações (por exemplo, imagens) que não são compilados em resources.arsc (Tam et al., 2017).

AndroidManifest.xml declara todos os componentes da aplicação, qualquer permissão exigida por esta, versões SDK[7], bibliotecas da API do Android necessárias para a aplicação estar ligada e, finalmente, recursos de hardware e software exigidos pela aplicação (Android, 2019c).

Classes.dex contém o bytecode Dalvik e é usado para executar a aplicação usando o Android runtime (ART) (Android, 2019b)

4.3. Android *malware*

Em 2018 foi comunicado a descoberta de um software malicioso que estava instalado de fábrica em mais de 5 milhões de dispositivos, em algumas das marcas mais vendidas no mercado, como é o caso da Huawei, Xiaomi, Lg ou Samsung (Khandelwal, 2018).

Este software malicioso aproveitava-se de algumas das permissões mais sensíveis do sistema operativo Android para executar ações maliciosas.

Milhões das aplicações maliciosas Android são detetadas todos os anos. Essas aplicações encontram-se disfarçadas como boas aplicações no Android *Marketplace* de forma a enganar os utilizadores. Quando uma aplicação maliciosa se encontra instalada num dispositivo Android

[6] ABIs: Os dispositivos Android usam diferentes CPUs, que suportam conjuntos de instruções diferentes. Todas as combinações de CPUs e instruções sets possui uma ABI (Application Binary Interface) que define como o código da máquina deve interagir com o sistema em tempo de execução. Uma aplicação deve selecionar qual ABI deseja usar (Android, 2019a)

[7] SDK versões: Diferentes dispositivos Android executam diferentes versões da plataforma Android (versão do SDK ou níveis da API). Uma aplicação deve especificar a versão do SDK de destino, bem como a versão mínima do SDK em que pode ser executada.

esta pode exibir anúncios, roubar credenciais bancárias, efetuar registo em serviços de SMS Premium ou encriptar o dispositivo e pedir um resgate. É essencial detetar e remover essas aplicações nas lojas (*marketplace*) antes que essas aplicações maliciosas sejam instaladas pelos utilizadores.

Também é importante categorizar essas ameaças. Ao se categorizar essas ameaças pode ajudar um analista de segurança na avalização de ameaças identificando estratégias de mitigação e técnicas de remoção (Fauskrud, 2019).

4.3.1. Construção

Existem três métodos diferentes que são frequentemente usados para criar *malware* para dispositivos moveis Android (Fauskrud, 2019; Polychronakis & (Eds.), 2017).

- ***Standalone (criados do zero)***: o *malware* foi escrito do zero.
- ***Repacking or piggybacking (reembalados)*****:** A aplicação é descompilada e o *malware* é inserido antes da aplicação ser novamente compilada. São usadas duas abordagens diferentes de reembalagem, a isolada e a integrada.

  Na isolada reembalar o código malicioso está incluído na aplicação, mas não tem nenhuma conexão com a funcionalidade original das aplicações. O código malicioso terá seu próprio manipulador de eventos como componente de ativação.

  Na integrada o autor do *malware* modifica o código original para inserir a carga maliciosa, tornando-o mais perigoso e com menor probabilidade de ser detetado.
- ***Library (biblioteca):*** O código malicioso está contido numa biblioteca numa aplicação benigna. A biblioteca é incluída pelo autor original da aplicação, que pode não ter conhecimento do Código malicioso. Esse método é comum para publicidade em *malware* (adware).

Num estudo elaborado com 24.650 aplicações maliciosos recolhidas entre de 2010 e 2016, chegou-se a conclusão de que 35% foram criados do zero, 7% foram reembaladas e 58% dos aplicativos continham uma biblioteca maliciosa (Polychronakis & (Eds.), 2017).

4.3.2. Distribuição das aplicações

Geralmente as aplicações Android são disponibilizadas através da loja da Google a chamada Google Play[8]. Estas também são usadas pelos autores de *malware* para distribuir *malware*

[8] https://play.google.com/store

disfarçado de aplicações fidedignas. A carga maliciosa pode estar oculta no *malware* ou ser efetuado o download pelo *malware* posteriormente.

O *malware* também é distribuído através de diferentes sites, usando um método chamado *drive-by-download*.

Drive-by-download pode ser definido como:

1. Downloads que uma pessoa autorizou, mas sem entender as consequências;
2. Qualquer download que ocorra sem o acesso de uma pessoa, por exemplo, o utilizador pode ser solicitado a fazer download do adobe *Flash player* enquanto visualiza um site (Fauskrud, 2019; Naway & Li, 2019).

4.3.3. Ativação

O código malicioso pode não ser acionado quando a aplicação é iniciada. Os métodos mais comuns de ativação são os seguintes:

- **By-host-app**: O código malicioso é ativado juntamente com o código da aplicação. Este método de ativação é usado para *malware* criado por reembalagem integrada (Fauskrud, 2019; Polychronakis & (Eds.), 2017).
- **Time-based:** O código malicioso é ativado algum tempo após a aplicação ser executada (Fauskrud, 2019; Polychronakis & (Eds.), 2017).
- **Event-based:** a carga útil é ativada com base em eventos relacionados ao Android, como quando a presença do utilizador, a conectividade do dispositivo muda ou quando um aplicativo é instalado ou apagado (Fauskrud, 2019; Polychronakis & (Eds.), 2017; Wei et al., 2017).

4.3.4. Servidor

O *malware* comunica com os servidores para aumentar as funcionalidades e se adaptar melhor ao meio ambiente. Ao comunicar com um servidor, o *malware* pode receber instruções, atualizações e enviar informações ao ator da ameaça. O *malware* Android é conhecido por usar SMS e HTTP para comunicação com os servidores (Fauskrud, 2019; Polychronakis & (Eds.), 2017; Wei et al., 2017).

4.3.5. Roubo de informação

Num estudo elaborado por (Wei et al., 2017), verificou-se que mais de 87% das amostras de *malware* recolhiam informações acerca do dispositivo infetado. Informações como IMEI, IMSI,

aplicações instaladas, versão do SO, idioma. O IMEI[9] e IMSI[10] são exclusivos para um dispositivo e podem ser usados como um identificador para comunicar com o servidor, embora as outras informações possam ser usadas pelo servidor para decidir outras ações a ter no dispositivo (Fauskrud, 2019; Polychronakis & (Eds.), 2017; Wei et al., 2017).

4.3.6. Persistência

Técnicas de persistência são utilizadas pelo *malware* para permanecer no dispositivo infetado após instalação. Estar mais tempo no dispositivo significa mais receita. A persistência pode ser alcançada por:

- Ser furtivo: o *malware* pode ocultar sua presença no dispositivo limpando logs, estando a ser executando em segundo plano e ocultando notificações por SMS e chamadas.
- Impedindo a remoção: Escondendo-se da lista de administradores de dispositivos, matando o software antivírus e bloquear o dispositivo.

4.3.7. Escalonamento de privilégios

Ao obter privilégios de administrador, o *malware* pode obter persistência e aceder a funcionalidades privilegiadas (por exemplo, alterar o PIN do ecrã de bloqueio, bloquear o dispositivo e excluir dados). O *malware* deve enganar o utilizador para conceder privilégios de administrador. O uso de Explorações de *root* (*root exploits*) tornaram-se menos populares devido ao aumento segurança do sistema operativo Android (Fauskrud, 2019; Wei et al., 2017).

4.3.8. Tipos de *malware*

Os tipos de *malware* mais comuns para Android são apresentados de seguida. De salientar que esses tipos não são mutuamente exclusivos e um *malware* pode ser classificado em um ou mais desses tipos (Chebyshev, 2019; Fauskrud, 2019; Polychronakis & (Eds.), 2017; Wei et al., 2017).

- **Trojan:** *Malware* que parece legítimo, mas contém uma componente maliciosa. normalmente requer interação do utilizador para instalação.
- **Premium Service Subscription:** O *malware* subscreve serviços SMS *premium*, de modo a gerar lucros, enquanto oculta essa atividade do utilizador.

[9] IMEI: international mobile station equipment identity.
[10] IMSI: international mobile subscriber identity

- **Banking Trojan:** Este deteta a existência de uma aplicação para acesso ao banco no dispositivo móvel. Alguns *banking trojans* substituem a aplicação bancária original por uma própria. Outros são conhecidos por criarem sobreposições do ecrã no momento do uso da aplicação real do acesso ao banco enganando o utilizador.
- **Dropper:** Os *dropper* são usados como um meio de ocultar uma carga maliciosa, evitando a sua deteção
- **Downloader:** Semelhante ao *dropper* com a diferença do *payload*[11] ser descarregado através de um servidor**.**
- **Ransomware:** O dispositivo do utilizador é bloqueado deixando de responder ou os ficheiros ficam encriptados. É exigido um resgate monetário para desbloquear o dispositivo ou desencriptar os ficheiros.
- **Adware:** Rouba os dados pessoais, exibe anúncios indesejados de uma maneira agressiva de modo ao utilizador efetuar download de aplicações potencialmente perigosas.
- **Spyware:** O spyware monitora a atividade do utilizador para recolher informações como localização, nomes de utilizador e palavras passe.
- **RiskTool:** Programas que incluem funcionalidades como ocultar ficheiros no sistema, ocultar a janela de aplicações em execução ou finalização de processos ativos. Esses programas não são necessariamente maliciosos por si mesmos. Um exemplo são os mineradores de criptomoedas (kaspersky, 2019).
- **Backdoor:** Programas que permitem acesso não detetado e não autorizado ao dispositivo móvel.
- **Worm:** Programas que fazem cópias de si mesmos e se espalham para outros dispositivos.

4.3.9. Nomes de *malware*

É prática comum categorizar *malware* em famílias de *malware*. Uma família de *malware* pode indicar atribuição de autor, campanha de *malware* ou outras características, como semelhanças no código-fonte.

As famílias de *malware* podem ser categorizadas com variantes. Um exemplo é a família Zen, que é um agrupamento de *malware* com base na atribuição do autor. Os autores do Zen

[11] Payload –Em segurança da informação o termo Payload refere-se a um código malicioso que executa uma ação destrutiva no sistema alvo, fornecendo acesso privilegiado e permissões (Security, 2018).

utilizaram diferentes estratégias de monetização para gerar receita. A variante mais simples da família de *malware* insere uma biblioteca de publicidade e um *trojan*, enquanto na outra variante mais complexa escalam táticas que levam o utilizador a clicar num link fraudulento (Fauskrud, 2019; Wei et al., 2017).

Os antivírus geralmente atribuem um rótulo a cada amostra de *malware*.

4.4. Análise de aplicações para dispositivos móveis Android.

Uma extensa lista de ferramentas e recursos de análise do Android pode ser encontrada em repositórios como Github[12], no entanto, descrevemos algumas das aplicações usadas neste projeto que permitem efetuar uma análise estática e dinâmica a determinada aplicação instalada num dispositivo móvel Android.

- **Santoku**

O Santoku é uma distribuição Linux open-source que contém diversas ferramentas para efetuar engenharia reversa e análises forenses, incluindo análise de *malware* às aplicações e dispositivos móveis (Santoku, 2019).

O Santoku é baseado na distribuição Linux Lubuntu, utiliza a interface *Lightweight X11 Desktop Environment* (LXDE), que aloca poucos recursos tornando-se deste modo uma distribuição muito rápida e eficiente. Sendo o Santoku uma distribuição Linux, esta disponibiliza um conjunto vasto de ferramentas, no entanto iremos apenas destacar as mais relevantes para este projeto.

- **Android Debug Bridge (ADB)**

Android Debug Bridge (ADB) é uma ferramenta cliente/servidor presente no Santoku que permite através da linha de comandos criar uma interface entre o dispositivo Android e o sistema operativo Santoku. O ADB é composto por três componentes (Google Developers, 2019a):

- **Cliente**, que envia comandos.
- **Daemon**, que executa comandos no dispositivo. Um daemon é executado como um processo em segundo plano em cada instância do emulador ou dispositivo.

[12] https://github.com/ashishb/android-security-awesome

    - **Servidor**, que gere a comunicação entre o cliente e o daemon. O servidor é executado como um processo de segundo plano em seu computador de desenvolvimento.

- **Dex2jar -** Esta ferramenta tem como função converter ficheiros Android DEX para ficheiros java JAR. Todo o processo e realizado através da linha de comandos. Caso a operação seja bem sucedida, o ficheiro obtido pode ser compilado de forma a se obter o código fonte da aplicação (Fauskrud, 2019; Pxb1988, 2019). Esta ferramenta também se encontra disponível no Santoku.

- **JD-GUI - Java decompiler -** Também presente no Santoku, JD-GUI é a ferramenta utilizada para completar a anterior a Dex2jar. Enquanto a aplicação anterior converte os ficheiros DEX para ficheiros JAR, esta incide nos ficheiros JAR obtendo o código fonte. Esta ferramenta contém interface gráfica e mostra o código fonte JAVA presente nos ficheiros CLASS incluídos nos JAR (Dupuy, 2019).

- **Apktool -** Esta ferramenta baseada em java é usada para descomplicar o ficheiro APK de forma a obter o código fonte e respetivos recursos (Apache, 2019). Com esta ferramenta é possível descomplicar o ficheiro AndroidManifest.xml, ficheiro que contém todas as informações de configuração da aplicação (Fauskrud, 2019; Google Developers, 2019b). Esta ferramenta encontra-se disponível no Santoku.

- **MobSF -** O MobSF é uma ferramenta *open-source* que permite a análise de aplicações nos sistemas operativos Android e iOS. Esta ferramenta disponível no Santoku permite descobrir as vulnerabilidades nos mais diversos campos tais como:
    - Permissões inseguras /impróprias;
    - Código inseguro;
    - Listagem de APIs.

  Esta aplicação permite ainda efetuar uma análise estática e dinâmica (Abraham & Superpoussin22, 2019; Fauskrud, 2019) podendo gerar relatórios da análise em formato .pdf.

- **Drozer -** Esta ferramenta também presente no Santoku, é uma das ferramentas mais avançadas para efetuar uma análise dinâmica. O Drozer inclui um conjunto diversificado de módulos que permite descobrir vulnerabilidades de segurança em aplicações ou dispositivos. Utiliza a arquitetura cliente-servidor onde o servidor é disponibilizado na forma de uma APK, para ser instalado no dispositivo Android. Já o Santoku, que já inclui a *framework*, terá o papel de cliente. Qualquer comando que seja inserido no Santoku será executado no dispositivo Android (InfoSecurity, 2019).

A Tabela 4-1 mostra um comparativo das aplicações usadas neste trabalho. Destacamos o Santoku que como Sistema operativo, engloba a maioria das outras ferramentas.

Tabela 4-1 - Ferramentas para análise de uma aplicação

	Aplicativo	Tipo ficheiro	Objetivo
Santoku	Distribuição Linux	diversos	Análise aplicações nos dispositivos móveis
ADB	Ferramenta Google		Interface entre servidor e dispositivo móvel
Dex2jar	Ferramenta	Android DEX	Converte ficheiros DEX para java
JD-GUI	Ferramenta	JAR	Obtém código fonte num ficheiro JAR
Apktool	Ferramenta	APK	Obtém código fonte num ficheiro APK
MobSF	Ferramenta		Descobrir vulnerabilidades numa aplicação
Drozer	Ferramenta		Descobrir vulnerabilidades numa aplicação

5. Análise da aplicação Autenticação.gov

Nesta secção descreve-se a aplicação Autenticação.gov, a instalação em dispositivos móveis com o sistema operativo móvel Android, ativação da Chave Móvel Digital através do site www.autenticação.gov.

Apresenta-se também o Cartão de Cidadão, documento fundamental e necessário para ativação da Chave Móvel Digital. Conclui-se com a realização de testes à aplicação.

5.1. Cartão de Cidadão

O Cartão de Cidadão (CC) é o novo documento de identificação dos cidadãos portugueses. É um documento autêntico que contêm os dados de cada cidadão relevantes para a sua identificação. O Cartão de Cidadão da República Portuguesa começou a ser emitido para os cidadãos nacionais em fevereiro de 2007 e foi criado como política integradora de desenvolvimento científico e tecnológico, integrando o "programa científico" desenvolvido como projeto de modernização.

Como funções principais, e de acesso direto ao cidadão, este cartão tem como funcionalidade imediata a identificação visual e presencial do cidadão, tal como acontecia com o antigo documento de identificação, o Bilhete de Identidade. Como função inovadora de utilização, o CC é dotado de uma nova funcionalidade de identificação e autenticação eletrónica que pode ser implementada em atos informatizados que assim o permitam.

Este possui diversas caraterísticas inovadoras. Uma delas é o facto de ser um cartão que substitui quatro outros cartões (Zúquete, 2018):

- Bilhete de Identidade
- Cartão de Contribuinte
- Cartão de Beneficiário da Segurança Social
- Cartão de Utente do Serviço Nacional de Saúde

Pretendeu-se assim evitar a propagação de suportes físicos sem, no entanto, reduzir o universo de números atribuídos a cada cidadão. Outras das caraterísticas inovadoras está relacionada com a tecnologia que o Cartão de Cidadão usufrui que é um *smartcard* (cartão inteligente), devido ao facto de este possuir um chip embebido. Este *smartcard* possui diversas funcionalidades tais como (Zúquete, 2018):

- Guardar informação pessoal de forma a esta poder ser validada informaticamente. Esta informação é composta por referências biométricas de impressões digitais do titular. Estas são usadas apenas internamente para validar uma impressão digital comunicada com sucesso;
- Guardar informação privada. Esta informação é a que o titular pode usar, mas não conhecer ou divulgar. Esta informação é constituída por três chaves criptográficas:
    - Uma chave simétrica de autenticação do titular;
    - Uma chave privada de um par de chaves assimétricas RSA, que permite a autenticação do titular;
    - Uma chave de um par de chaves assimétricas RSA, que permite produzir assinaturas digitais do titular.
- Guardar informação reservada. Esta é a informação que o titular conhece, mas que apenas disponibiliza de forma fidedigna, através do *smartcard* a quem o desejar, ou tiver autorização para a obter independentemente da vontade do titular. Esta informação é composta pela morada do titular.
- Guardar a informação pública de grande dimensão não memorizável. Informação esta constituída pela fotografia do titular e por certificados X.509.3 de chaves públicas do titular. Estas podem ser usadas para autenticar o titular ou a sua assinatura digital.
- Guardar toda a informação do titular observável no Cartão de Cidadão tais como:
    - Nome
    - Data de nascimento
    - Diversos números de identificação
    - Validade do cartão
- Efetuar operações criptográficas usando as diversas chaves que fazem parte da sua informação privada
- As operações efetuadas com *smartcard* que integra o Cartão de Cidadão necessitam que o mesmo indique um código secreto– Personal Identification Number (PIN). Cada CC possui três PIN necessários para:
    - Autorizar a indicação de morada;
    - Autenticar o titular;
    - Conceber uma assinatura digital.

Os PIN são elementos chave do Cartão do Cidadão que o tornam pessoal. A perda deste não permite a quem o encontrar usufruir das funcionalidades contidas no seu *smartcard*, pois o

número de tentativas de descoberta do seu PIN está limitado a três. Existe também um código de cancelamento de oito dígitos que poderá ser usado nestes casos e que torna inválidas toda as funcionalidade do *smartcard* (Zúquete, 2018).

Inicialmente, os pares de chaves assimétricas RSA de autenticação e assinatura tinham 1024 bits e as assinaturas eram calculadas com algoritmos de síntese MD5, SHA-1 ou RIPEMD-160.

A partir de maio de 2015 o comprimento do algoritmo foi atualizado para 2048 bits passando também a ser suportada a função de síntese SHA-256 (Zúquete, 2018). Recentemente uma nova verão do CC saiu passando a que os tamanhos das chaves sejam de 3072 bits. Esta atualização está implícita no verso do Cartão de Cidadão a partir da versão 006.008.24 que incorpora também um novo chip versão 6 (INCM, 2020). A Figura 5-1 mostra o local no CC onde se pode visualizar a versão.

Figura 5-1 - Versão do CC (INCM, 2020)

O número da versão do documento (CC) é constituído por 10 caracteres com a seguinte estrutura (INCM, 2020):

$$|A|A|A|.|B|B|B|.|C_1|C_2|$$

A, B e **C,** assumem valores [0-9][A-Z] com a dimensão total de 10 carateres incluindo os separadores ".". O Nº de versão é constituído por três grupos distintos, separados pelo carater".":

- **|A|A|A|** este grupo de caracteres corresponde ao número de versão dos componentes gráficos e de segurança do Cartão físico;
- **|B|B|B|** O segundo grupo de carateres corresponde ao número de versão dos elementos do *chip* (*chip*, versão do SO do *chip*, aplicações e caraterísticas do chip);
- **|C_1|C_2|** O primeiro caracter corresponde ao número de versão da raiz da cadeia de confiança do Cartão de Cidadão.
  O segundo carater corresponde à identificação da alteração do número de versão dos certificados do cidadão.

5.1.1. Autenticação com o *smartcard* do Cartão de Cidadão

No *smartcard* presente no Cartão de Cidadão constam os dados inscritos no cartão, com exceção da assinatura digitalizada, e as aplicações que permitem a execução das seguintes funcionalidades:

- IAS – aplicação responsável pelas operações de autenticação e assinatura eletrónica;
- EMV-CAP – aplicação responsável pela geração de palavras-chave únicas por canais alternativos (por exemplo: telefone);
- Match-on-card – aplicação responsável pela verificação biométrica de impressões digitais.

A Figura 5-2, mostra as aplicações e dados contidos no *smartcard* do Cartão de Cidadão (Ama, 2007).

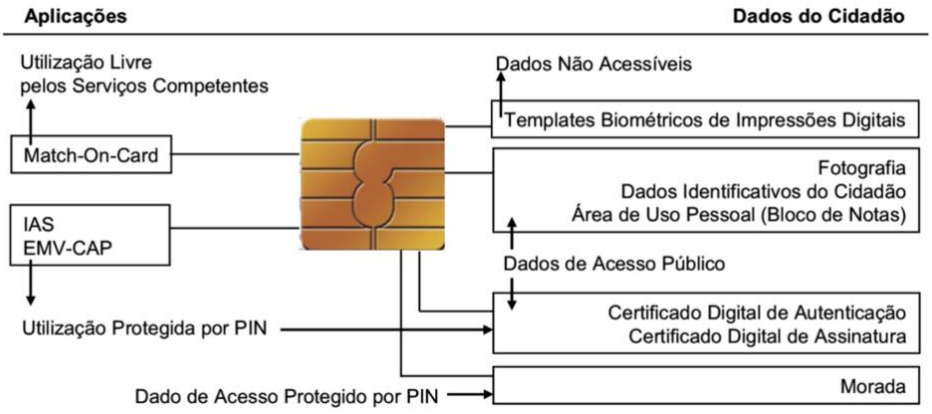


Figura 5-2 - Informação e aplicações residentes no *smartcard* do CC (Administrativa, 2007, p.9)

A autenticação com o *smartcard* do Cartão de Cidadão pode ser efetivada de duas formas destintas.

Uma das formas destina-se a autenticar o titular sem recorrer a meios computacionais usando o EMV-CAP, inserindo o cartão num leitor pessoal, digitando o PIN de autenticação no terminal e no ecrã do mesmo aparecerá uma senha descartável *one time password* (OTP) (Ama, 2007; Zúquete, 2018).

Esta senha descartável pode ser transmitida por qualquer meio de comunicação como por exemplo telemóvel ou email, a quem precisar de autenticar uma dada operação. O autenticador valida a senha descartável comunicando em conjunto com a identidade do sujeito a autenticar a uma entidade que consegue validar a senha descartável. Esta responderá afirmativamente se a senha estiver correta e, negativamente, caso contrário.

A senha descartável é gerada através de um algoritmo existente no *smartcard* e da chave simétrica secreta guardada no mesmo. A entidade com a responsabilidade de validar a senha descartável usa os mesmos elementos, algoritmo e chave de modo a validar se a senha descartável apesentada é valida ou não (Zúquete, 2018).

Outra forma de autenticação tem como objetivo autenticar o titular no universo computacional. Para esse efeito o *smartcard* contem um par de chaves assimétricas de autenticação que podem ser usadas por diversas aplicações e protocolos para autenticar o titular. O PIN de autenticação do titular deve de ser enviado para o *smartcard* sempre que necessário usar a chave privada. O *smartcard* disponibiliza um certificado X.509 com a chave pública de autenticação do titular. Este pode ser comunicado aos interlocutores do titular para que os mesmos possam verificar a validade da chave privada de autenticação do titular (Zúquete, 2018).

5.1.2. Assinaturas digitais com o Cartão de Cidadão

Uma das funcionalidades disponível no Cartão de Cidadão é a possibilidade de usar a assinatura digital presente neste. As assinaturas digitais são uma forma não repudiável de autenticação de documentos garantido em simultâneo a inalterabilidade de um documento assim como a sua autoria (Zúquete, 2018).

O *smartcard* presente no Cartão do Cidadão possui um par de chaves assimétricas de assinatura digital, podendo ser usadas por diversas aplicações para assinar documentos. O

certificado X.509.3 existente no *smartcard* disponibiliza a chave pública de validação da assinatura digital do titular. O PIN de assinatura digital deve ser usado de cada vez que for necessário usar a chave privada do par de chaves de assinatura digital do titular (Zúquete, 2018).

5.1.3. Certificados digitais utilizados pelo Cartão de Cidadão

Os certificados presentes no Cartão de Cidadão são certificados X.509.3 de chaves públicas do titular. As chaves podem ser utilizadas para autenticar o titular e usar a sua assinatura.

Um certificado digital é um documento eletrónico assinado criptograficamente por uma autoridade de certificação, associando uma chave pública a uma entidade (Almeida, 2009; Hutchison & Mitchell, 2006) . Esta entidade pode ser uma pessoa, uma organização, uma aplicação informática ou qualquer outra entidade confiável pela autoridade de certificação *Certification Authority* (CA). No certificado reside ainda um conjunto de atributos que definem o propósito do mesmo (e.g. Assinatura) e que caracterizam a entidade certificada.

Uma CA é uma entidade que gere certificados, definindo políticas, mecanismos de geração e distribuição de certificados e listas de revogação de certificados. As autoridades de certificação podem utilizar hierarquias de certificação assinando certificados que contêm a chave pública de outras autoridades de certificação (Almeida, 2009; SCEE, 2020).

Na Figura 5-3 é possível visualizar e identificar as hierarquias das entidades de certificação dos certificados presentes no Cartão de Cidadão.

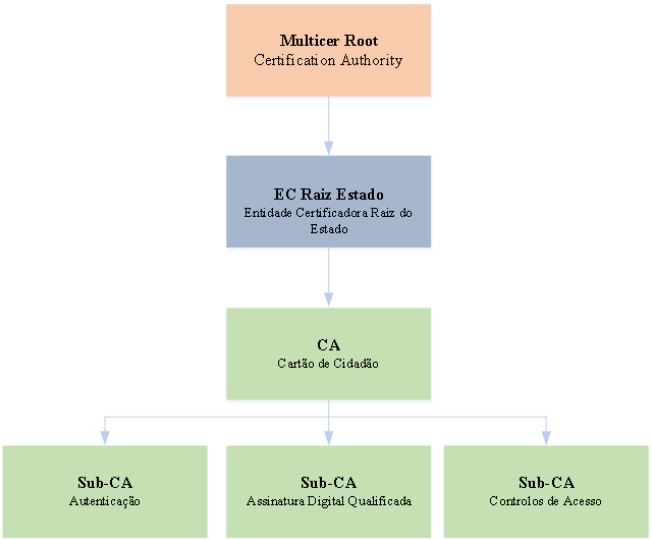


Figura 5-3 - Hierarquia de CA dos certificados presentes no Cartão de Cidadão adaptado de (SCEE, 2020)

Os certificados presentes no CC são certificados X.509 versão 3, sendo utilizado este mecanismo de extensão, por exemplo, para distinguir o tipo de utilização da chave pública presente (X509v3). No certificado de Assinatura Digital do CC encontra-se presente a extensão "*Key Usage*" que garante o não-repudio da assinatura Figura 5-4.

Figura 5-4 - Certificado da Assinatura Digital

No certificado exibido na figura acima outros atributos são possíveis de observar para alem do não-repudio. Atributos dos quais destacamos:

- ***Serial Number**:* número de serie, que é um número inteiro atribuído por uma CA, devendo ser único para cada certificado emitido por essa CA.
- ***Signature Algorithm**:* indica o método de assinatura utilizado pela CA para assinar o certificado.
- ***Issuer Name***: encontra-se a entidade que emitiu e assinou o certificado.
- ***Valid***: A validade do certificado é especificada com uma data de início de validade *Not Valid Before* e uma data de término *Not Valid After*. O espaço de tempo entre estas duas datas define o período de validade do certificado, em que este pode ser utilizado para a realização de assinaturas.
- ***Public Key info***: identifica os dados relacionados com a chave pública.

5.1.4. *Middleware* do Cartão de Cidadão

Middleware é um software que fica entre o sistema operativo e as aplicações em execução nele, no caso do Cartão de Cidadão o *middleware* é o que estabelece a ponte entre as aplicações que usam o *smartcard* e os próprios *smartcard*. Em geral o *middleware* é fornecido pelos fabricantes do *smartcard* (Microsoft, 2019).

No caso do cartão podemos encontrar dois tipos de *middleware*:

1. PTEID PKCS11
2. PTEID Lib

O primeiro PTEID PKCS11 é o que permite usar o Cartão de Cidadão como dispositivo criptográfico.

O segundo tipo de *middleware* PTEID Lib, permite realizar operações com o Cartão de Cidadão que são especificas do documento de identificação não tendo paralelo nos dispositivos criptográficos. Exemplos dessas operações são a obtenção da fotografia, morada ou alteração do PIN entre outras (Zúquete, 2018).

5.1.5. SCMD - Serviço Chave Móvel Digital

A Chave Móvel Digital (CMD) é um sistema simples e seguro de autenticação dos Cidadãos em vários sites públicos e privados, apenas com um número de telemóvel e um PIN de 4 dígitos. Este serviço necessita de ser solicitado online através do site www.autenticacao.gov.pt/ ou presencialmente num balcão de atendimento (Espaço do Cidadão).

A Chave Móvel Digital contem um sistema de autenticação com dois fatores de segurança. Estes são (Ama, 2019):

- uma palavra-chave escolhida pelo cidadão (PIN);
- um código de segurança numérico e temporário recebido por SMS, e-mail ou mensagem direta no Twitter

Em alterativa ao SMS ou mensagem direta no Twitter foi desenvolvida a aplicação Chave Móvel Digital. Esta permite a receção do código de segurança associado a cada autenticação através de notificação *push* para o *smartphone* (Ama, 2019).

5.1.6. Aplicação Chave Móvel Digital - CMD

A Chave Móvel Digital foi lançada em 2015, pela AMA, permitindo que, através da associação do número de identificação civil a um único número de telemóvel e ou a um único endereço de correio eletrónico e com o mesmo código pessoal, os cidadãos possam dispensar outras senhas no acesso aos serviços públicos disponibilizados na Internet (Ama, 2019).

Numa segunda fase surge a aplicação CMD em alternativa a autenticação via SMS ou mesmo através das redes sociais (Twitter). A aplicação CMD encontra-se disponível nas lojas oficias da Google - Play Store e da Apple através da App Store.

A CMD é um meio de autenticação que através da associação de um número de telemóvel ao número de identificação civil para um cidadão português e o número de passaporte para um cidadão estrangeiro, possibilitando assinar eletronicamente documentos de vários formatos. A assinatura através da Chave Móvel Digital permite ao cidadão português ou estrangeiro, assinar um determinado documento com uma palavra-chave por si escolhida e respetivo código de segurança. Ao executar a autenticação com Chave Móvel Digital é utilizado o número de telemóvel, o código PIN da CMD ou o código numérico único e temporário de 6 dígitos enviado por SMS para o número de telemóvel anteriormente referido ou via e-mail. Com a aplicação da CMD permite ao utilizador utilizar a sua chave no Portal SNS, da Segurança Social, IMT, entre outros.

Com a CMD podemos assinar documentos digitais com a mesma validade de uma assinatura à mão. Para tal é necessário ter a assinatura digital da CMD ativada e código PIN de assinatura da CMD

5.1.7. Instalação e Configuração da aplicação Autenticação.gov

A aplicação encontra-se disponível nas lojas oficias da Google - Play Store e da apple através da App Store com o nome AUTENTICAÇÃO GOV[13]. A Figura 5-5 mostra a aplicação na Play Store da Google. Na App Store o processo é similar, no entanto, este estudo tem incidência na aplicação para o sistema operativo Android.

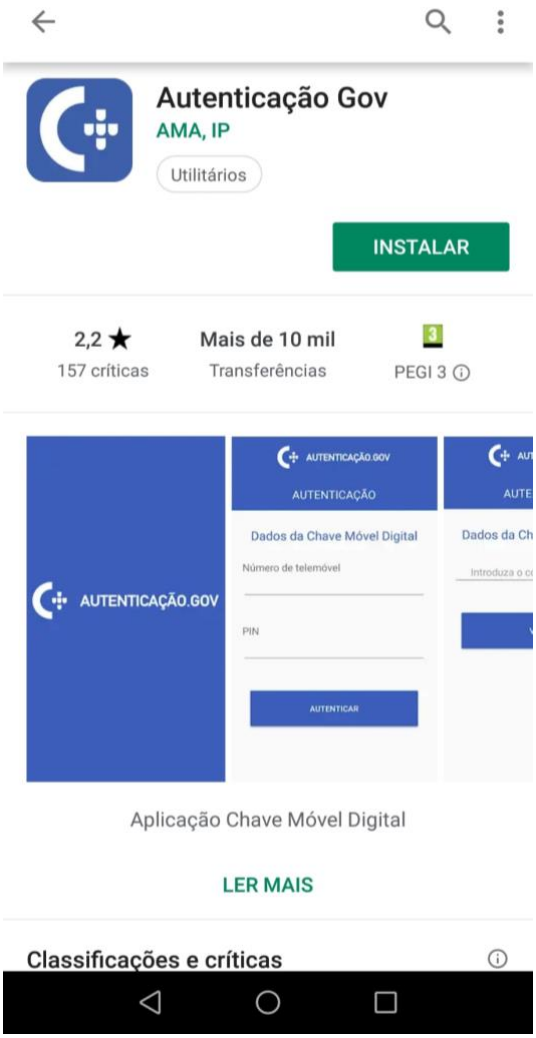


Figura 5-5 - Play Store – Autenticação.gov

Para que a aplicação Autenticação.gov funcione corretamente é necessário aceder ao site www.autenticacao.gov.pt/cmd-pedido-chave para efetuar o pedido da CMD. É necessário ter um leitor do Cartão de Cidadão e os respetivos códigos fornecidos a quanto da emissão deste.

A Figura 5-6 mostra a informação geral da CMD no site www.autenticacao.gov.pt. Nesta área, para alem da visualização dos dados pessoais é possível alterar o número de telemóvel,

[13] Autenticação Gov - Google Play: https://tinyurl.com/y4cnemb2

email, ativar a aplicação móvel, ativar a aplicação móvel via Twitter assim como subscrever a Assinatura Digital.

É também necessário definir um PIN de autenticação. Esse PIN é importante, pois é necessário aquando o uso da aplicação.

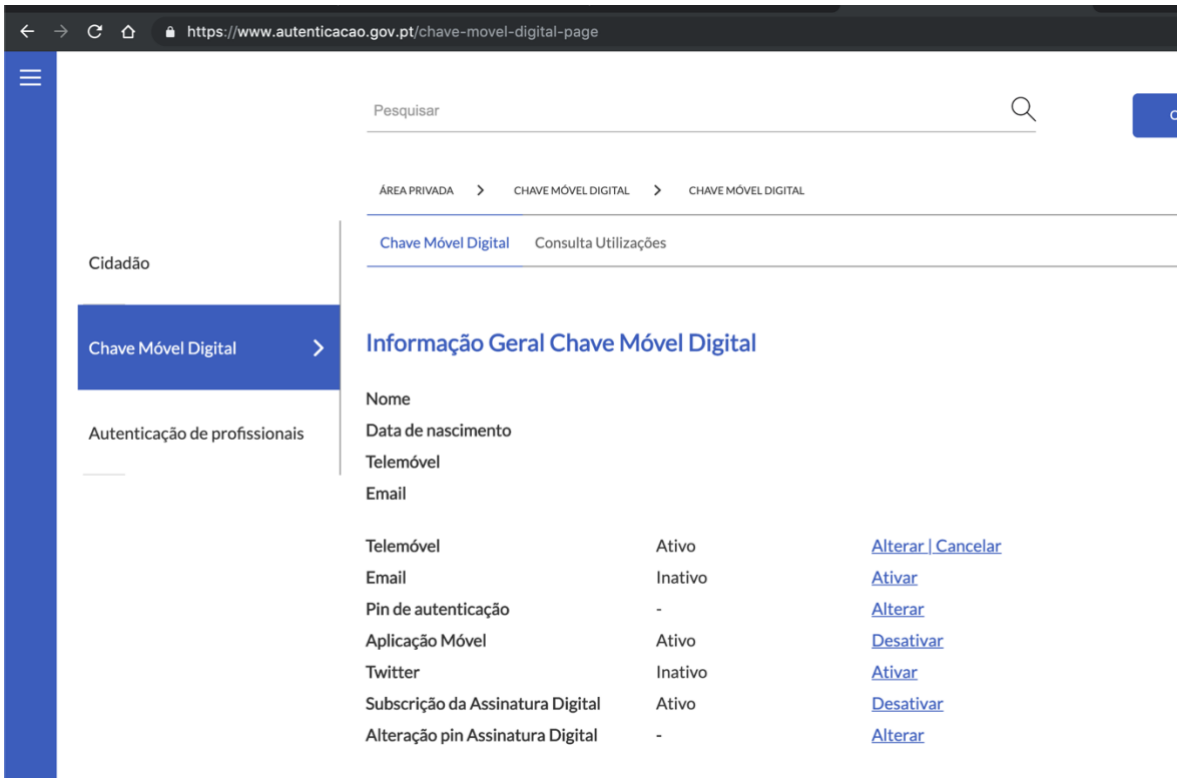


Figura 5-6 - Chave Móvel Digital ativação

5.1.8. Autenticação com a chave móvel digital – CMD

A aplicação CMD substitui a autenticação através do CC. Para autenticar através do CC é necessário o Cartão de Cidadão, leitor de cartões e o respetivo PIN de autenticação.

Para autenticar através da Chave Móvel Digital é necessário o número de telemóvel ou e-mail, o PIN da CMD e o código de segurança numérico e temporário (Ama, 2019).

O processo de autenticação de um documento é semelhante, no entanto, após seleção do ficheiro a autenticar é necessário selecionar autenticação com Chave Móvel Digital como podemos visualizar na Figura 5-7.

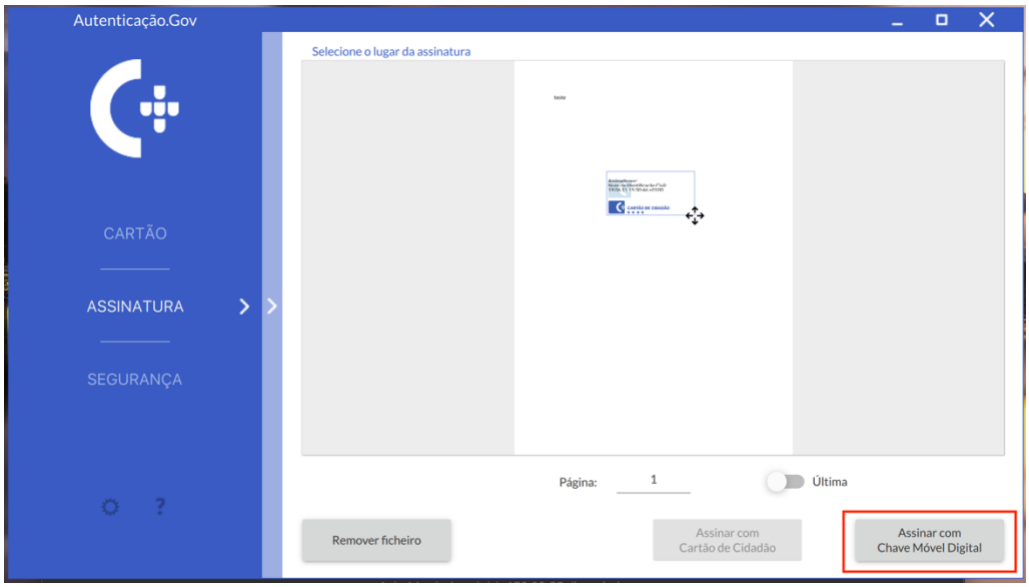


Figura 5-7 - Assinar com Chave Móvel Digital

O próximo passo é inserir o número de telemóvel onde a aplicação Autenticação.gov está instalada e ativada e o PIN de Assinatura Figura 5-8. O número de telemóvel e o PIN de Assinatura são os que foram introduzidos no site www.auntenticação.gov.pt como mostra a Figura 5-6.

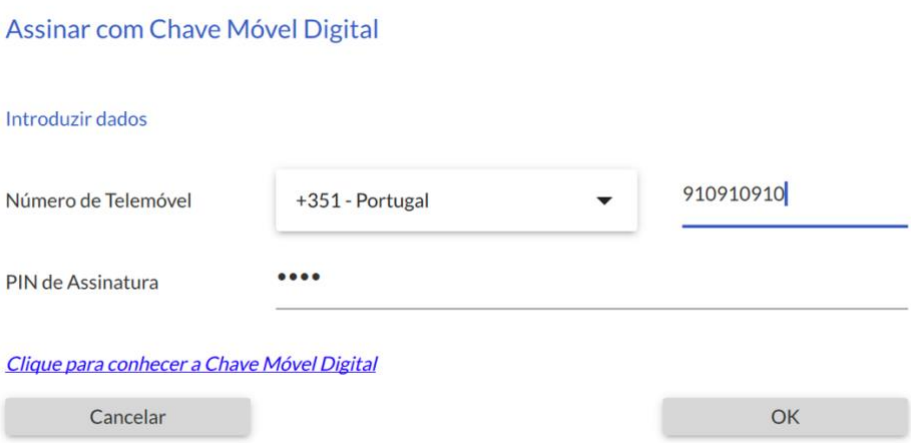


Figura 5-8 - Assinar com Chave Móvel Digital - dados

No dispositivo móvel ao abrir a aplicação Autenticação.gov, irá aparecer o nome do ficheiro a autenticar e o respetivo código de segurança de 6 carateres a fornecer para concluir o processo de autenticação do documento.

Figura 5-9 - Código de Segurança

Na Figura 5-10 é inserido o código de segurança que foi gerado no dispositivo móvel como mostra a Figura 5-9.

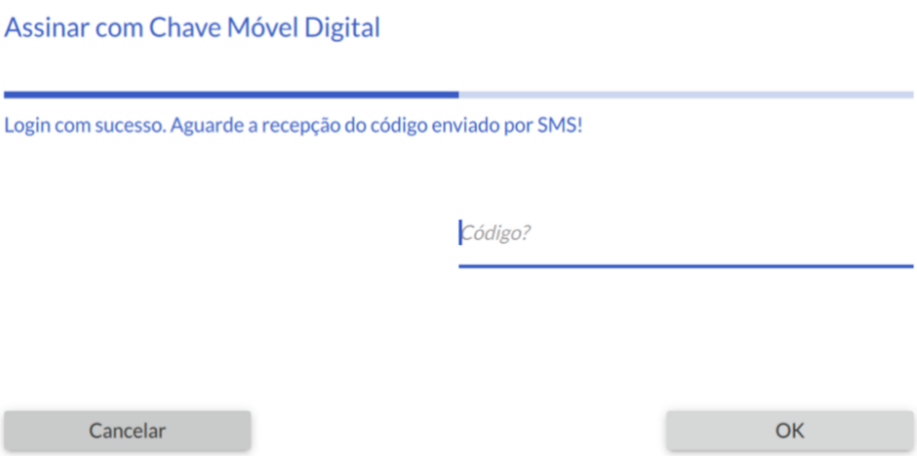


Figura 5-10 - Introdução código de segurança

Após introdução correta do código de segurança e para concluir a autenticação do documento é apresentada uma mensagem a informar do sucesso da operação Figura 5-11, possibilitando a abertura imediata do ficheiro assinado.

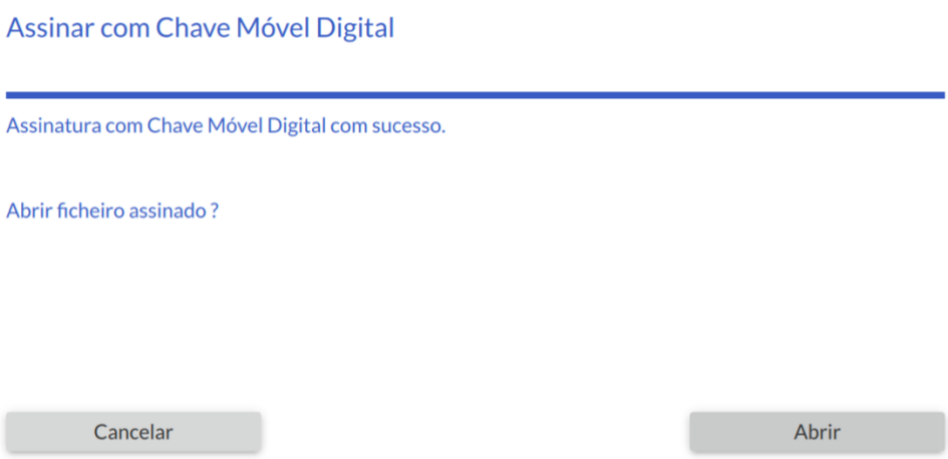


Figura 5-11 - Termino da assinatura com a CMD

5.2. Testes à aplicação Autenticação.Gov

Neste ponto numa primeira fase iremos proceder a análise da aplicação Autenticação.Gov para dispositivos móveis com o sistema operativo Android e serão efetuados testes de análise estática, análise dinâmica, análise dos logs e por fim ao código fonte. Por fim serão expostos os resultados obtidos.

5.2.1. Cenário de testes

Foram elaborados dois cenários de testes.

Neste ponto exibe-se o cenário de testes para efetuar uma análise estática e dinâmica da aplicação analisado seus comportamentos logs e código.

O processo de análise da aplicação iniciou-se com a elaboração de um cenário de testes. O primeiro cenário de testes engloba três componentes principais:

- Sistema Operativo *Linux Santoku*;
- Dispositivo móvel Android virtual criado através do *Genymotion*;
- Aplicação Autenticação.Gov.

A Figura 5-12 ilustra o cenário concebido para realizar os testes.

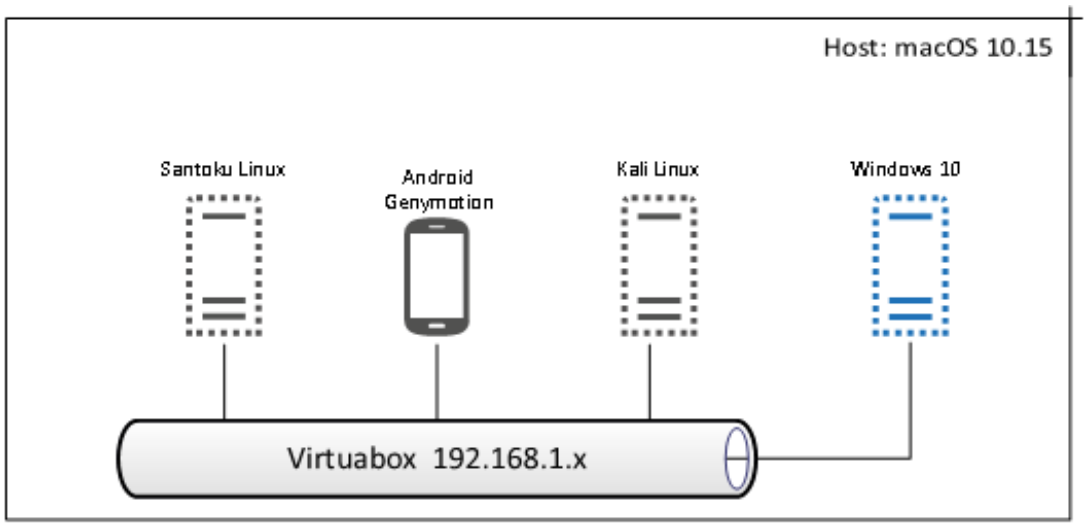


Figura 5-12 – Diagrama do cenário de testes

Para além dos 3 componentes mencionados (*Santoku Linux*, *Android Genymotion* e a aplicação autenticação.gov) foram adicionados mais dois componentes.

- Kali Linux, para elaboração de testes;
- Windows 10, para instalação de uma aplicação de *decompiler* apenas disponível para este sistema operativo.

Como mencionado no capítulo Análise de *Malware*, o Santoku é uma distribuição Linux que engloba a maioria das ferramentas necessárias para este projeto. A versão instalada é a 0.5 e foi instalada através de uma máquina virtual (VM) no *hypervisor* virtualbox versão 5.2.26. A VM foi criada com as seguintes especificações:

- 2Gb de memoria RAM
- 1 disco para armazenamento de dados de 15Gb
- 2 processador
- 2 placas de rede

A Figura 5-13 mostras as principais características da VM com o sistema operativo Linux Santoku no *hypervisor* VirtualBox.

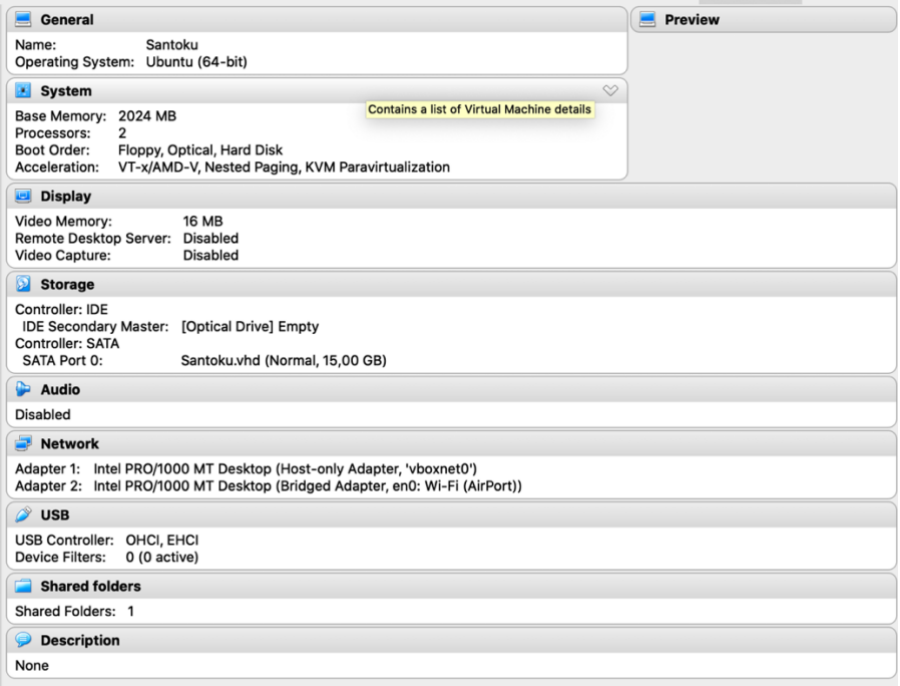


Figura 5-13 - Especificações da VM com os SO Santoku

O SO Santoku baseia-se na distribuição *linux Lubuntu* e utiliza a interface gráfica *Lightweight X11 Desktop Environment* (LXDE), que requer poucos recursos de hardware tornando-o muito rápido e eficiente. A Figura 5-14 mostra o ambiente de trabalho do sistema operativo Santoku.

Figura 5-14 - Ambiente de trabalho do SO Santoku

Para efetuar os diversos testes à aplicação Autenticação.Gov (versão 2.0.0) foi usado um emulador para a Android o *Genymotion* (versão 3.1.1). O *Genymotion* consiste num programa que é instalado no SO neste caso no macOS. A instalação é efetuada através do site

https://www.genymotion.com/, sendo necessário efetuar um registo. Para que possa emular um dispositivo móvel que selecionamos para o projeto o *GenyMotion* necessita do Oracle VM VirtualBox.

O dispositivo criado foi baseado no *Android Nougat* (Android versão 7.1), onde foi instalada a aplicação Autenticação.Gov. A Figura 5-15 mostra as especificações do dispositivo virtual.

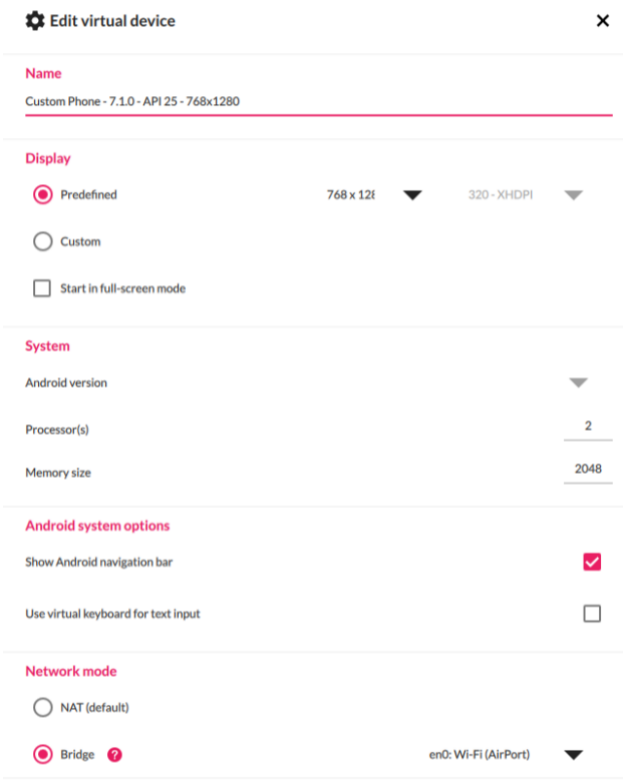


Figura 5-15 - GenyMotion especificações do dispositivo virtual

A figura anterior mostra as configurações que foram usadas para o emular o dispositivo móvel Android através do emulador GenyMotion, bem como as configurações do dispositivo móvel usado neste projeto. A escolha recaiu pelo Android 7.1 por uma questão de compatibilidade e por ainda existirem muitos dispositivos com essa versão no mercado fruto da fragmentação existente no SO Android. Na Figura 5-16 podemos visualizar algumas das configurações do dispositivo móvel virtual usado neste projeto assim como efetuar o iniciar "start" do dispositivo móvel ou o editar que vai para o menu da Figura 5-15. É possível visar também alguns dos diversos *templates* disponíveis.

Quando queremos iniciar o dispositivo móvel é neste menu que o fazemos. Automaticamente a VM irá iniciar no *VirtualBox*.

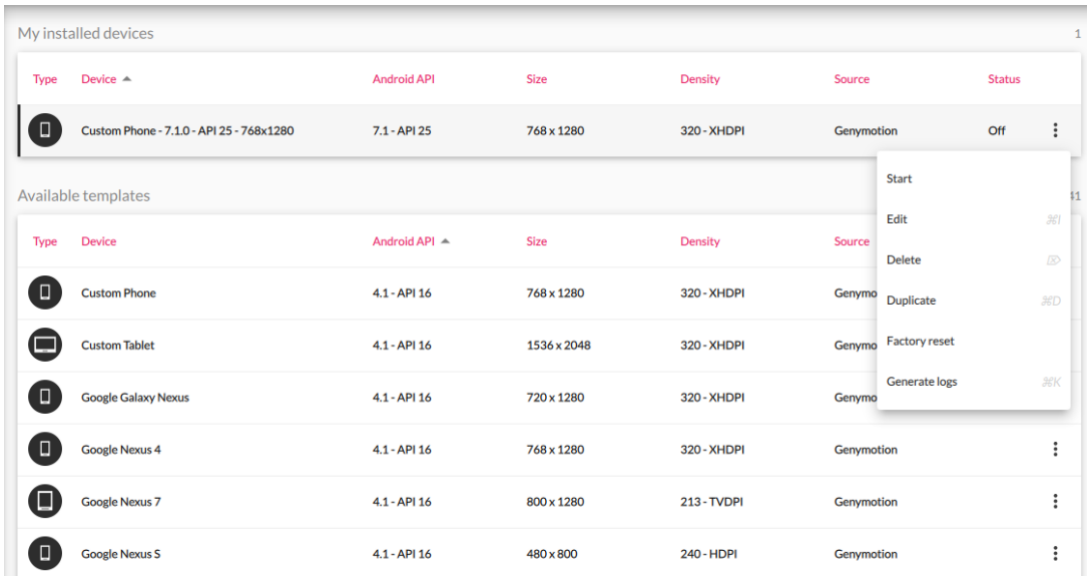


Figura 5-16 - GenyMotion

Na Figura 5-17 visualizamos as características do *GenyMotion* no *hypervisor VirtualBox*. Para que se consiga comunicar com o Santoku foi adicionada a placa de rede em modo de *bridge*

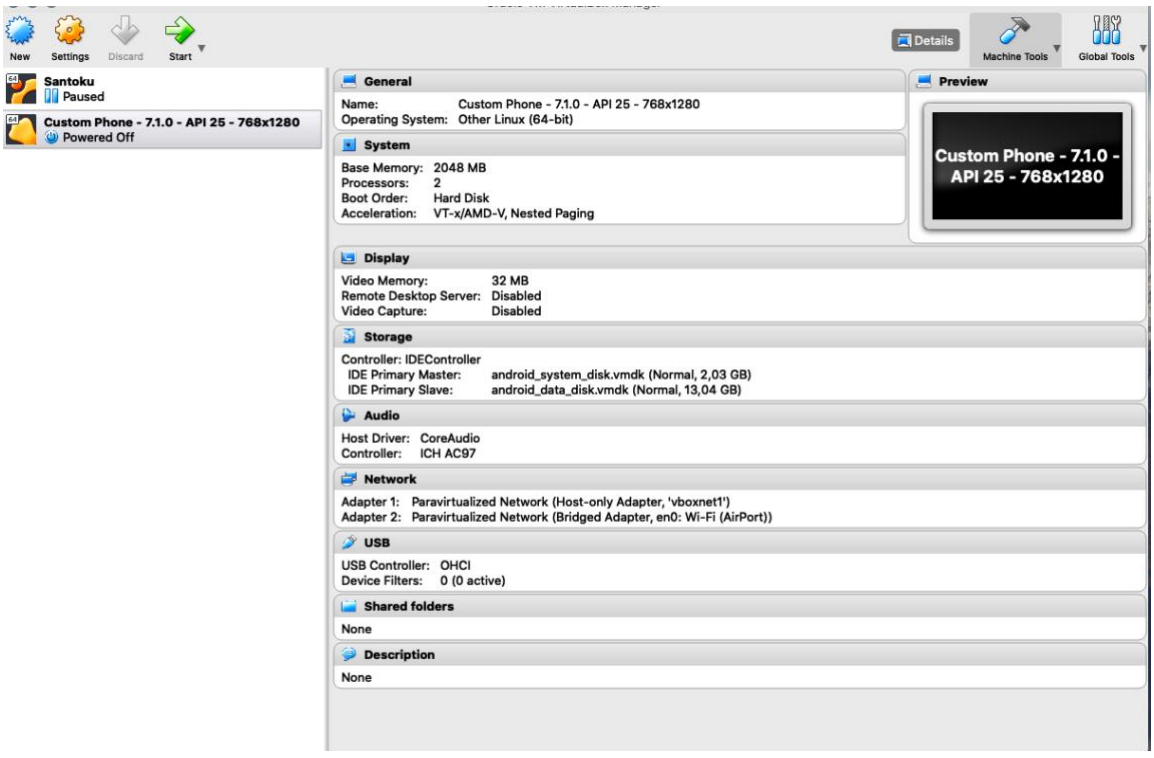


Figura 5-17 - VirtualBox especificações do dispositivo móvel Android

Como citado no capítulo 4.4 a ferramenta *Android Debug Bridge* (ADB) permite através da linha de comandos criar uma interface entre o dispositivo Android (cliente) e o sistema

operativo (servidor). Na Tabela 5-1 podemos visualizar alguns dos comandos ADB mais usados durante a realização de testes.

Tabela 5-1 - Listagem de comandos ADB

Comando	descrição
adb connnect [ip]	Cria ligação ao dispositivo Android
adb devices	Lista todos os dispositivos Android ativos
adb -s [id] shell	Acede à shell de um dispositivo Android
adb install [aplicação]	Instala uma aplicação no Android
adb uninstall [aplicação]	Desinstala uma aplicação no Android
adb push [ficheiro] [diretório destino]	Copia ficheiros do Servidor para o Android
adb pull [ficheiro]	Copia ficheiros do Android para o Servidor
adb logcat	Gera um registo com mensagens do sistema geradas pela classe Log

Obtido o endereço IP do dispositivo virtual Android (192.168.1.217) foi possível começar a efetuar alguns testes. O primeiro teste foi verificar os dispositivos conectados ao servidor Santoku realizado o comando *adb devices* como mostra a Figura 5-18.

```
1  tiagojoao@santoku-VirtualBox:~$ adb devices
2  List of devices attached
3  192.168.1.217:5555  device
```

Figura 5-18 - ADB devices comando

Para efetuar uma ligação entre o servidor e o dispositivo Android e efetuado o comando adb connect 192.168.1.217 como podemos visualizar na Figura 5-19. A partir deste momento é possível realizar as mais diversas tarefas no dispositivo Android, tais como a instalação de novas aplicações, acesso à *shell* do sistema ou visualização de *logs* do dispositivo Android.

```
1  tiagojoao@santoku-VirtualBox:~$ adb connect 192.168.1.217
2  * daemon not running. starting it now on port 5037 *
3  * daemon started successfully *
```

Figura 5-19 - ADB connect

As aplicações Android são distribuídas na forma de um ficheiro compactado com a extensão .apk, que significa Android Package. Estes ficheiros não passam de ficheiros ZIP que contém o código da aplicação, os recursos e os *metadados* necessários para o funcionamento desta. Para proceder à análise da aplicação Autenticação.gov foi necessário obter o respetivo ficheiro APK.

Ao descompactar o APK da aplicação Autenticação.gov através do comando unzip Autenticação\ Gov_v2.0.0.apk -d AutenticacaoGOV é possível observar todo o seu conteúdo como podemos verificar na Figura 5-20.

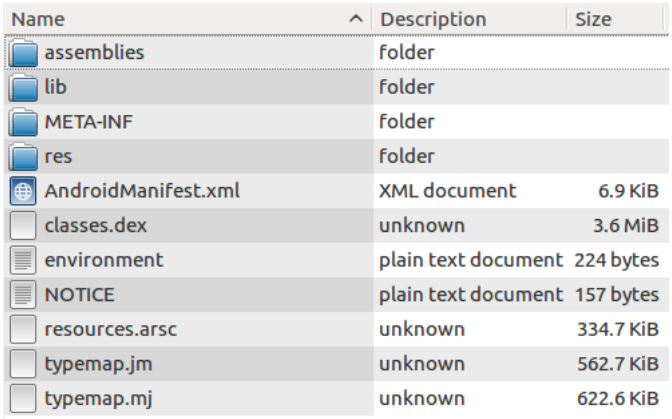


Figura 5-20 - Ficheiros e pastas contidos na aplicação Autenticação.GOV

Da figura acima é relevante destacar a estrutura de ficheiros e pastas que é uma estrutura típica utilizada nos ficheiros APK como já mencionado anteriormente no capítulo 4.2.

5.2.2. Permissões da aplicação Autenticação.gov

O SO Android está desenvolvido para que qualquer aplicação que seja instalada no dispositivo solicite acesso a determinada informação e recursos do dispositivo.

No momento da instalação da aplicação, é solicitado o acesso conjunto de permissões que o utilizador terá de aceitar se pretender usufruir da mesma. Cada permissão tem uma denominação exclusiva que é usada para a referenciar no código. Por omissão, todas as permissões vêm especificadas no ficheiro AndroidManifest.xml (Figura 5-20) que a Google obriga a colocar na raiz da aplicação, para facilitar esta análise. Importa analisar cada uma delas e perceber os perigos que possam representar.

Apesar de o ficheiro se encontrar em Extensible Markup Language (XML), este está num formato XML binário. Para que seja possível a sua leitura, é necessário proceder à sua

descompilação usando a ferramenta Apktool. A Figura 5-21 mostra a execução do comando apktool.

```
tiagojoao@santoku-VirtualBox:~/Desktop$ apktool d Autenticação_Gov_v2.0.0.apk ./Autenticação_Gov_v2.0.0.apk
I: Using Apktool 2.3.4 on Autenticação_Gov_v2.0.0.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
```

Figura 5-21 - Apktool para descompilar Autenticação.GOV

Concluído a execução do comando apktool podemos agora visualizar o ficheiro AndroidManifest.xml Figura 5-22, que nos permite visualizar as permissões solicitadas pela aplicação Autenticação.GOV.

```
<?xml version="1.0" encoding="utf-8" standalone="no"?><manifest xmlns:android="http://schemas.android.com/apk/res/android"
android:installLocation="preferExternal" package="pt.ama.autenticacaogov" platformBuildVersionCode="25" platformBuildVersionName="7.1.1">
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.USE_FINGERPRINT"/>
    <uses-permission android:name="android.permission.BIND_NOTIFICATION_LISTENER_SERVICE"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.READ_LOGS"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
    <uses-permission android:name="android.permission.VIBRATE"/>
    <uses-permission android:name="android.permission.WAKE_LOCK"/>
    <uses-permission android:name="com.example.proj.permission.C2D_MESSAGE"/>
    <permission android:name="pt.ama.autenticacaogov.permission.C2D_MESSAGE"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="pt.ama.autenticacaogov.permission.C2D_MESSAGE"/>
    <uses-permission android:name="com.google.android.c2dm.permission.RECEIVE"/>
    <uses-permission android:name="android.permission.GET_ACCOUNTS"/>
    <application android:allowBackup="false" android:debuggable="false" android:icon="@drawable/leftarrow" android:label="Autenticação Gov"
    android:name="android.app.Application">
        <activity android:icon="@drawable/aground" android:label="Autenticação Gov"
        android:name="md5ba9292671f075a189d6b15232e1c2e4f.MainActivity" android:screenOrientation="portrait" android:theme="@style/MainTheme">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <receiver android:name="md5a7279444c0fe86aa930f169dd7d0faa5.GcmBroadcastReceiver"
        android:permission="com.google.android.c2dm.permission.SEND">
            <intent-filter>
                <action android:name="com.google.android.c2dm.intent.REGISTRATION"/>
                <category android:name="pt.ama.autenticacaogov"/>
            </intent-filter>
            <intent-filter>
                <action android:name="com.google.android.gcm.intent.RETRY"/>
                <category android:name="pt.ama.autenticacaogov"/>
            </intent-filter>
            <intent-filter>
                <action android:name="com.google.android.c2dm.intent.RECEIVE"/>
                <category android:name="pt.ama.autenticacaogov"/>
            </intent-filter>
        </receiver>
        <service android:name="md5a7279444c0fe86aa930f169dd7d0faa5.GcmService"/>
        <activity android:label="Web Authenticator" android:name="md5a104545e4d19c4ffe9ec3d5074a3b979.FormAuthenticatorActivity"/>
        <activity android:label="Web Authenticator" android:name="md5a104545e4d19c4ffe9ec3d5074a3b979.WebAuthenticatorActivity"/>
        <provider android:authorities="pt.ama.autenticacaogov.mono.MonoRuntimeProvider.__mono_init__" android:exported="false"
        android:initOrder="2147483647" android:name="mono.MonoRuntimeProvider"/>
        <meta-data android:name="com.android.vending.derived.apk.id" android:value="1"/>
    </application>
</manifest>
```

Figura 5-22 - AndroidManifest.xml

5.2.3. Levantamento dos potenciais perigos das permissões

Obtidas as permissões através do ficheiro AndroidManifest.xml importa perceber a função de cada uma delas e possível risco que esta possa representar. Como resumo foi elaborada a Tabela 5-2

Tabela 5-2 - Análise às permissões solicitadas pela aplicação autenticação.GOV (Google, 2020)

Permissão	Descrição	Risco
.ACCESS_NETWORK_STATE	Permite que a aplicação aceda a informações sobre as redes	Normal
.USE_FINGERPRINT	Possibilita que a aplicação use a impressão digital	Normal
.BIND_NOTIFICATION_LISTENER_SERVICE	Consente que a aplicação envie notificações	Programável
.INTERNET	Permite que a aplicação inicie comunicação de rede	Normal
.READ_EXTERNAL_STORAGE	Possibilita a leitura de um dispositivo externo	Perigoso
.READ_LOGS	Permite que o aplicativo leia os arquivos de log de baixo nível do sistema	Perigoso
.READ_PHONE_STATE	Permite apenas o acesso de leitura ao estado do telefone, incluindo o número de telefone do dispositivo, as informações atuais da rede, o status de todas as chamadas	Perigoso
.VIBRATE	Classe que permite que o dispositivo vibre.	Normal
.WAKE_LOCK	Permite usar o PowerManager para impedir o processador entre em modo economia de energia ou o ecrã escureça	Normal
.C2D_MESSAGE	Autoriza que a aplicação envie notificações push up ao utilizador	Programável
.WRITE_EXTERNAL_STORAGE	Possibilita que a aplicação grave no armazenamento externo por exemplo cartão de memória	Perigoso
.GET_ACCOUNTS	Concede acesso à lista de contas no Serviço de Contas	Perigoso

Da tabela acima verificamos a existência de algumas permissões classificadas como "perigosas" pela Google. O acesso completo a dispositivos externos, a leitura de ficheiros de logs do sistema operativo assim como o acesso ao estado do dispositivo móvel são classes classificadas como perigosas, podendo comprometer os dados do disposto móvel onde estas forem usadas no desenvolvimento de uma aplicação como é o caso da aplicação em estudo autenticação.GOV.

5.2.4. Análise estática - extração do código-fonte

Como referido no capítulo 4.4 toda a arquitetura das aplicações Android assenta em ficheiros DEX muito difíceis de interpretar. Para realizar uma análise mais profunda a simples extração do ficheiro Android Package (APK) é insuficiente. O acesso ao código-fonte só é possível realizando o processo de engenharia reversa. Como já foi citado a engenharia reversa consta de

uma análise dinâmica de *malware* a uma aplicação no caso deste projeto a aplicação autenticação.gov.

A engenharia reversa irá permitira identificar todos os *exploits* ou vulnerabilidades que o código-fonte possa conter.

Ler e compreender o código descompilado é uma tarefa difícil. A maneira mais simples de analisar é obtendo o código-fonte. O *bytecode* da Dalvik incluído no ficheiro DEX é uma linguagem interpretada que pode ser convertido para código-fonte semelhante ao original. O código pode ser obtido convertendo os ficheiros DEX presentes na APK para um ficheiro JAR. Esta operação pode ser realizada através da aplicação dex2jar (versão 2.0). Toda a interação é realizada através da linha de comandos como podemos visualizar na Figura 5-23 mostra o comando utilizador para a efetuar a conversão.

```
1  tiagojoao@santoku-VirtualBox:~$ d2j-dex2jar /home/tiagojoao/Desktop/Autenticação\ Gov_v2.0.0.apk
2  dex2jar /home/tiagojoao/Desktop/Autenticação Gov_v2.0.0.apk -> Autenticação Gov_v2.0.0-dex2jar.jar
3  tiagojoao@santoku-VirtualBox:~$
```

Figura 5-23 - Dex2jar para obter ficheiro JAR

Após obtido o ficheiro JAR este pode ser descompilado de forma a obter o código-fonte da aplicação através da aplicação JD-GUI. Este utilitário *open-source* permite mostrar o código-fonte Java presente nos ficheiros CLASS incluídos mo ficheiro JAR.

Para tal através do programa JD-GUI abrir o ficheiro JAR obtido anteriormente (Autenticação Gov_v2.0.0-dex2jar.jar) como visualizar na Figura 5-24. Na figura a baixo podemos visualizar as várias *activities* que compõe a aplicação, ao selecionar qualquer uma delas é mostrado o respetivo código.

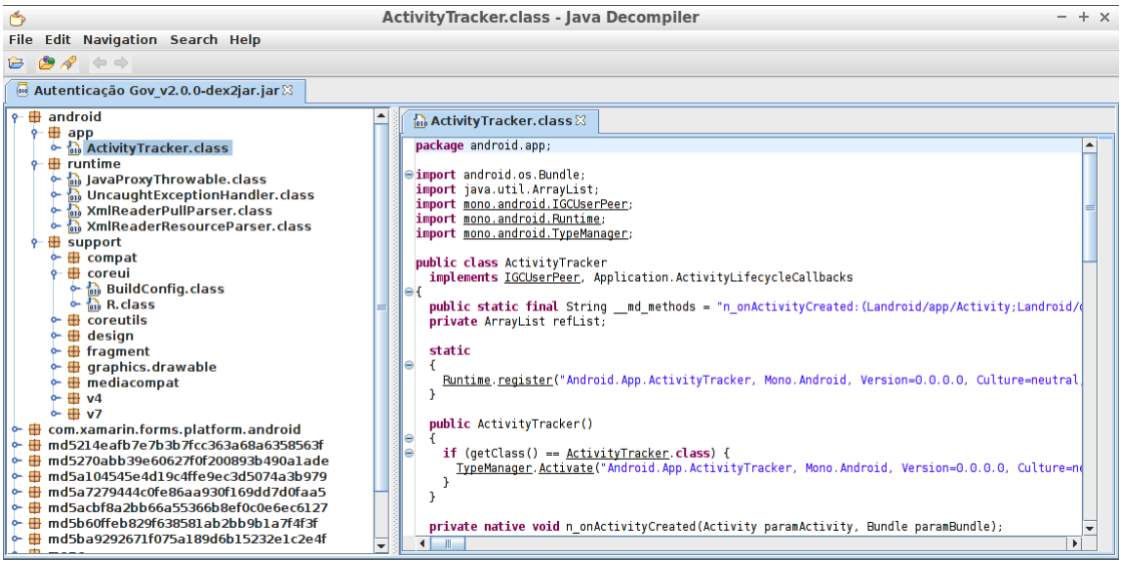


Figura 5-24 - Interface do JD-GUI

A análise estática é executada sem recorrer à execução do software ou aplicação. Este tipo de análise é efetuado sobre o código-fonte do software ou, no caso de aplicações Android, sobre o ficheiro APK através de ferramentas que tentam detetar possíveis vulnerabilidades no código-fonte usando técnicas como *Taint Analysis* ou *Data Flow Analysis*. Para realizar a análise estática à aplicação Autenticação.gov foi utilizada a ferramenta MobSF.

O MobSF é uma *framework* de testes para aplicações móveis que suporta os sistemas operativos Android e iOS. Esta *framework* pode ser usada para análises de segurança rápidas e eficazes, sendo aceites ficheiros APK, IPA e APPX assim como código-fonte comprimido (Abraham & Superpoussin22, 2019).

A *framework* MobSF não esta incluída no sistema operativo Santoku, sendo necessário proceder à sua instalação. O SO Santoku usa uma versão do Ubuntu que não é compatível com as últimas versões do MobSF.

Para testar a última versão do MobSF (3.0) foi usada a VM com o Kali Linux versão 2018.1. Fez-se o download através do repositório do GitHub como podemos visualizar na Figura 5-25.

```
1  git clone https://github.com/MobSF/Mobile-Security-Framework-MobSF.git
```

Figura 5-25 - MobSF GitHub

Os pré-requisitos necessário para a instalação do MobSF são:

- Install Git: `sudo apt get install git`
- Install Python **3.6 - 3.7:** `sudo apt-get install python3`
- Install JDK 8+: `sudo apt-get install openjdk-8-jdk`

Para que tudo seja instalado corretamente, o MobSF disponibiliza um ficheiro de texto *requirements.txt* que inclui todos os comandos necessário para a instalação das diversas dependências / pré-requisitos. É possível instalar todo de uma vez ao executar o camando da Figura 5-26.

```
1  ./setup.sh
```

Figura 5-26 - MobSF instalação

Após conclusão dos passos anteriores com sucesso é possível visualizar na linha de comandos a status da instalação como nos mostra a Figura 5-27.

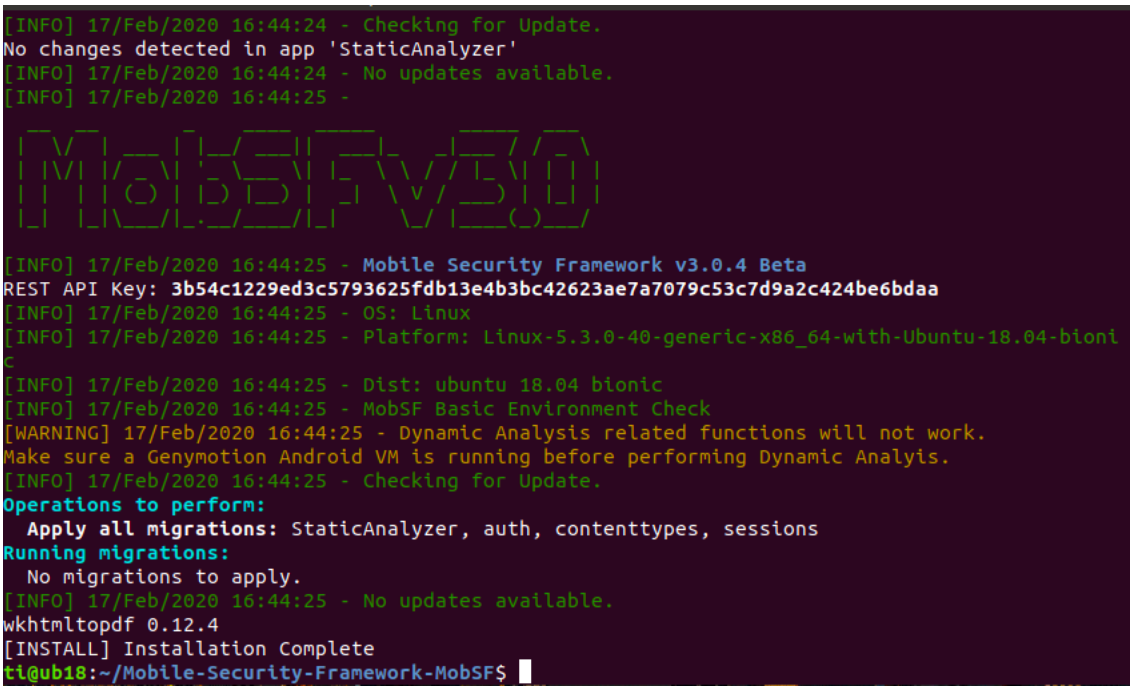


Figura 5-27 - MobSF instalação

Se todas as operações anteriores forem concluídas com sucesso é possível iniciar a *framework* MobSF executando o seguinte comando: ./run.sh.

A Figura 5-28 mostra o resultado do comando ***./run.sh*** que inicia a *framework* MobSF.

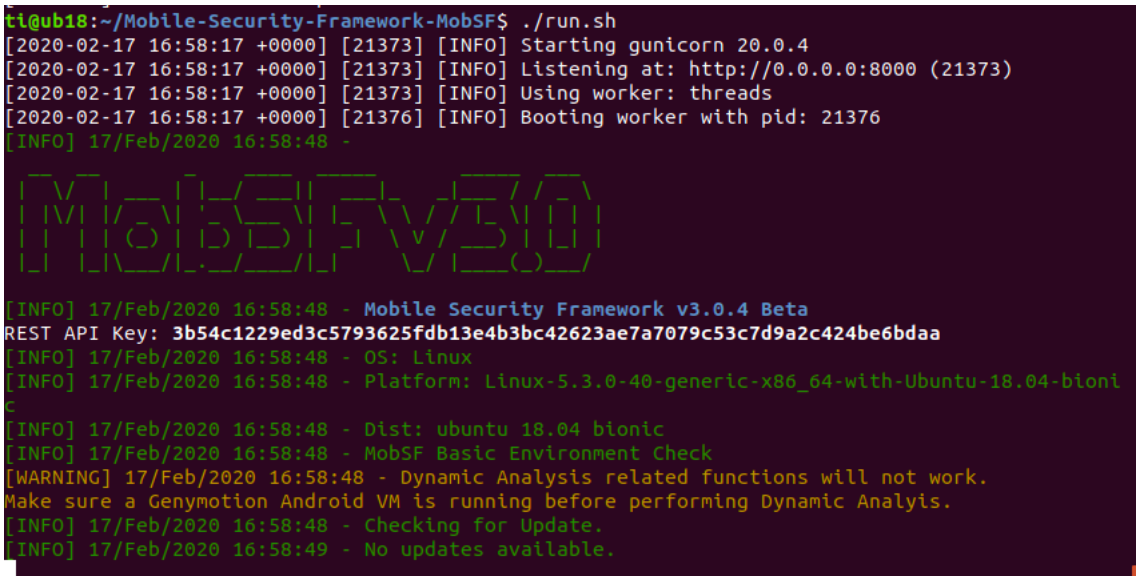


Figura 5-28 - Inicialização do servidor MobSF após instalação

O servidor está a ser executado localmente no porto 8000 (porto padrão), o acesso à interface gráfica pode ser realizado pelo browser no endereço http://127.0.0.1:8000/.

A

Figura 5-29 mostra a interface web do MobSF.

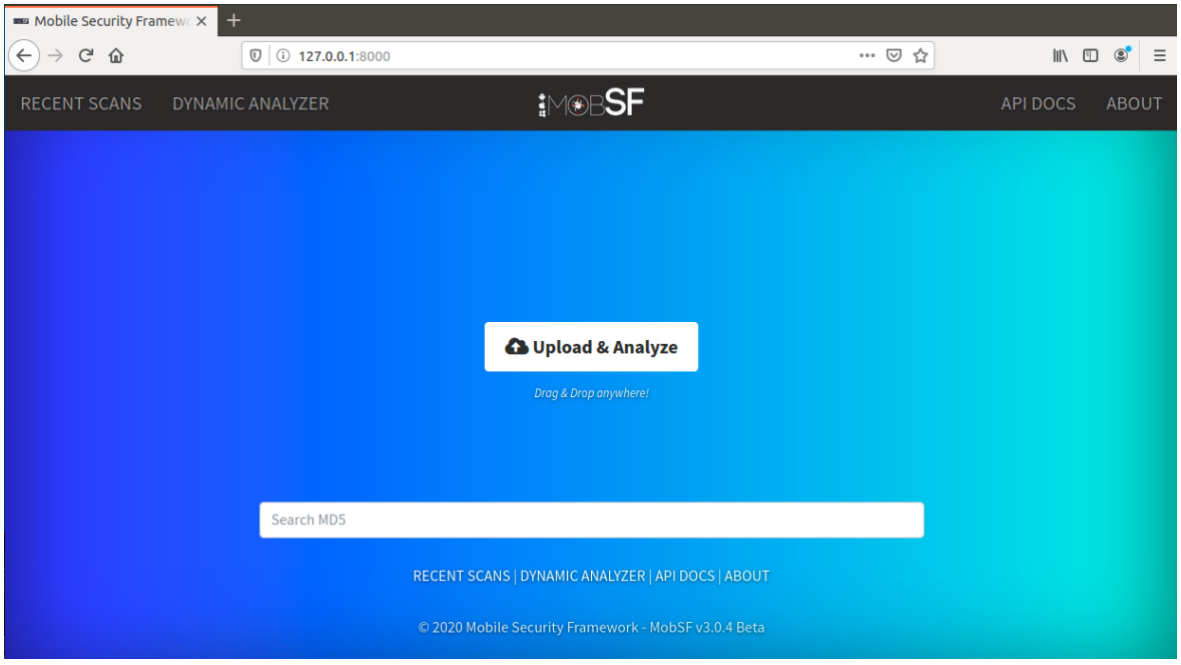


Figura 5-29 - Interface web do MobSF

Neste ponto o servidor MobSF encontra-se devidamente configurado e pronto para iniciar a análise do APK. Para prosseguir basta fazer o *updoad* do ficheiro autenticação_gov.apk. A

Figura 5-30 mostra os resultados da análise estática após termos efetuado o *upload* do ficheiro autenticação_gov.apk.

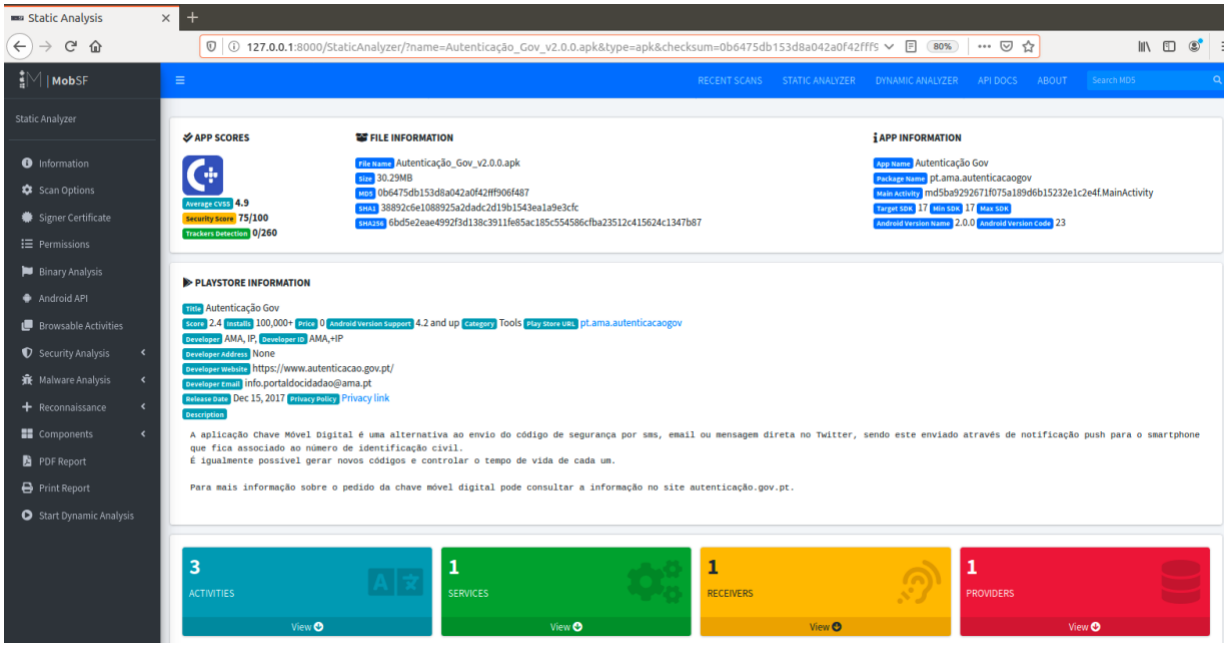


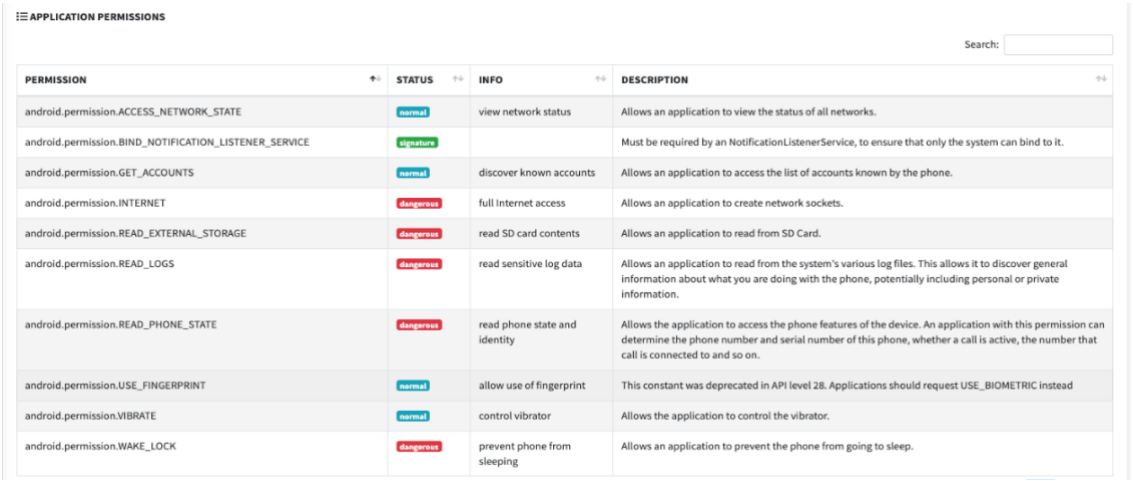
Figura 5-30 - MobSF Análise estática à aplicação Autenticação.gov

Terminada a análise o *MobSF* exibe os resultados, que também poderão ser exportados em PDF se o utilizador assim entender. O resultado da análise estática está disponível no Anexo A – MobSF deste projeto.

Iremos destacar, no entanto, os problemas detetados ao nível da análise do código-fonte:

- App can read/write to External Storage. Any App can read data written to External Storage;
- IP Address disclosure.

Poderemos verificar também de uma forma rápida Figura 5-31, as permissões que a aplicação Autenticação.gov necessita. Estas já foram levantadas após análise do ficheiro AndroidManifest.XML no ponto 5.2.3.Introdução



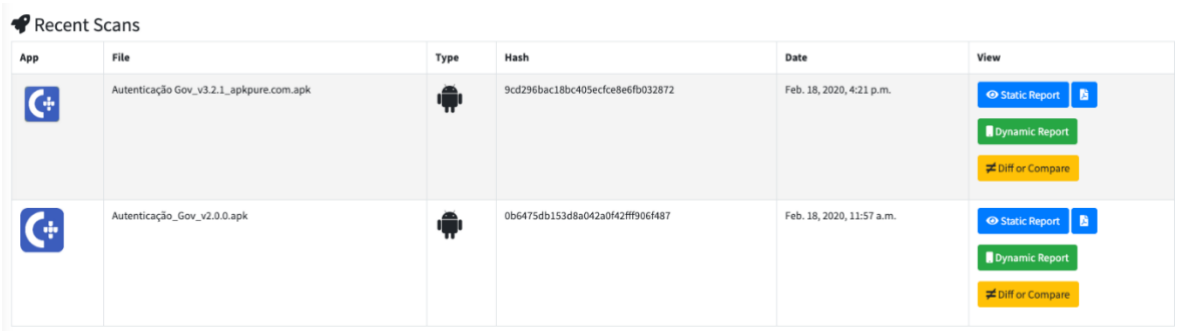
APPLICATION PERMISSIONS

Search:

PERMISSION	STATUS	INFO	DESCRIPTION
android.permission.ACCESS_NETWORK_STATE	normal	view network status	Allows an application to view the status of all networks.
android.permission.BIND_NOTIFICATION_LISTENER_SERVICE	signature		Must be required by an NotificationListenerService, to ensure that only the system can bind to it.
android.permission.GET_ACCOUNTS	normal	discover known accounts	Allows an application to access the list of accounts known by the phone.
android.permission.INTERNET	dangerous	full Internet access	Allows an application to create network sockets.
android.permission.READ_EXTERNAL_STORAGE	dangerous	read SD card contents	Allows an application to read from SD Card.
android.permission.READ_LOGS	dangerous	read sensitive log data	Allows an application to read from the system's various log files. This allows it to discover general information about what you are doing with the phone, potentially including personal or private information.
android.permission.READ_PHONE_STATE	dangerous	read phone state and identity	Allows the application to access the phone features of the device. An application with this permission can determine the phone number and serial number of this phone, whether a call is active, the number that call is connected to and so on.
android.permission.USE_FINGERPRINT	normal	allow use of fingerprint	This constant was deprecated in API level 28. Applications should request USE_BIOMETRIC instead
android.permission.VIBRATE	normal	control vibrator	Allows the application to control the vibrator.
android.permission.WAKE_LOCK	dangerous	prevent phone from sleeping	Allows an application to prevent the phone from going to sleep.

Figura 5-31 - Permissões da Aplicação

Outro teste que foi efetuado foi a comparação de diferentes versões da aplicação. A versão 2.0, versão existente aquando o início deste projeto e a versão 3.2.1, a versão atual. Foram efetuados os *uploads* dos APK das duas versões como podemos visualizar na Figura 5-32.



Recent Scans

App	File	Type	Hash	Date	View
	Autenticação Gov_v3.2.1_apkpure.com.apk		9cd296bac18bc405ecfce8e6fb032872	Feb. 18, 2020, 4:21 p.m.	Static Report / Dynamic Report / Diff or Compare
	Autenticação_Gov_v2.0.0.apk		0b6475db153d8a042a0f42fff906f487	Feb. 18, 2020, 11:57 a.m.	Static Report / Dynamic Report / Diff or Compare

Figura 5-32 - Autenticação_Gov.APK versões análise

A resultado da comparação das duas aplicações incide nas permissões.

O resultado da comparação mostra-nos as permissões comuns às duas versões como podemos visualizar na Figura 5-33.

PERMISSIONS

Common

PERMISSION	STATUS	INFO
android.permission.ACCESS_NETWORK_STATE	normal	view network status
android.permission.WAKE_LOCK	dangerous	prevent phone from sleeping
android.permission.USE_FINGERPRINT	normal	allow use of fingerprint
android.permission.BIND_NOTIFICATION_LISTENER_SERVICE	signature	
android.permission.INTERNET	dangerous	full Internet access
android.permission.READ_EXTERNAL_STORAGE	dangerous	read SD card contents
android.permission.READ_LOGS	dangerous	read sensitive log data
android.permission.READ_PHONE_STATE	dangerous	read phone state and identity
android.permission.VIBRATE	normal	control vibrator
pt.ama.autenticacaogov.permission.C2D_MESSAGE	signature	Allows cloud to device messaging
android.permission.WRITE_EXTERNAL_STORAGE	dangerous	read/modify/delete SD card contents
com.google.android.c2dm.permission.RECEIVE	signature	C2DM permissions
android.permission.GET_ACCOUNTS	normal	discover known accounts

Figura 5-33 - Permissões comuns

Na Figura 5-34, mostra-nos as permissões que constam apenas em cada versão.

Only in pt.ama.autenticacaogov - 3.2.1

PERMISSION	STATUS	INFO
com.samsung.android.providers.context.permission.WRITE_USE_APP_FEATURE_SURVEY	dangerous	Unknown permission from android reference
android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS	normal	
android.permission.CAMERA	dangerous	take pictures and videos
android.permission.ACCESS_WIFI_STATE	normal	view Wi-Fi status

Only in pt.ama.autenticacaogov - 2.0.0

PERMISSION	STATUS	INFO
com.example.proj.permission.C2D_MESSAGE	signature	Allows cloud to device messaging

Figura 5-34 - Permissões de cada versão

Na versão 3.2.1 foram identificadas duas novas permissões classificadas como perigosas:

- **com.samsung.android.providers.context.permission.WRITE_USE_APP_FEATURE_SURVEY**. Esta permissão existente numa biblioteca entretanto descontinuada da Samsung permite a ofuscação de código (Stackoverflow, 2017).

- **android.permission.CAMERA.** Esta permissão também descontinuada pela Google e classificada como perigosa permite o uso indevido da câmara do dispositivo móvel (Google, 2020).

5.2.5. Análise dinâmica

Como mencionado no capítulo 2.2, a análise dinâmica é uma técnica que se aplica a análise em tempo real. A análise dinâmica têm como objetivo de identificar vulnerabilidades e avaliar a segurança da mesma realizando testes de penetração. A identificação de vulnerabilidades determina quais os pontos onde uma aplicação pode ser explorada ou ameaçada. Já os testes de penetração exploram as falhas que a aplicação possui e mede as consequências de possíveis incidentes avaliando os mecanismos de defesa que são utilizados.

Para que a análise dinâmica seja executada em tempo real, é necessário proceder à sua instalação no dispositivo móvel. Como foi referido anteriormente o ADB facilita o acesso ao dispositivo móvel Android. A Figura 5-35 exemplifica a instalação através da ferramenta ADB. Outra opção para proceder a instalação poderia ser através da Play Store.

```
1  adb connect 192.168.1.217
2      connected to 192.168.1.217:5555
3  adb install Autenticação_Gov_v2.0.0.apk
4      Sucess
```

Figura 5-35 - Instalação da aplicação através do ADB

A Figura 5-36 mostra o software *Genymotion* que emula o dispositivo móvel com a aplicação Autenticação.GOV instalada.

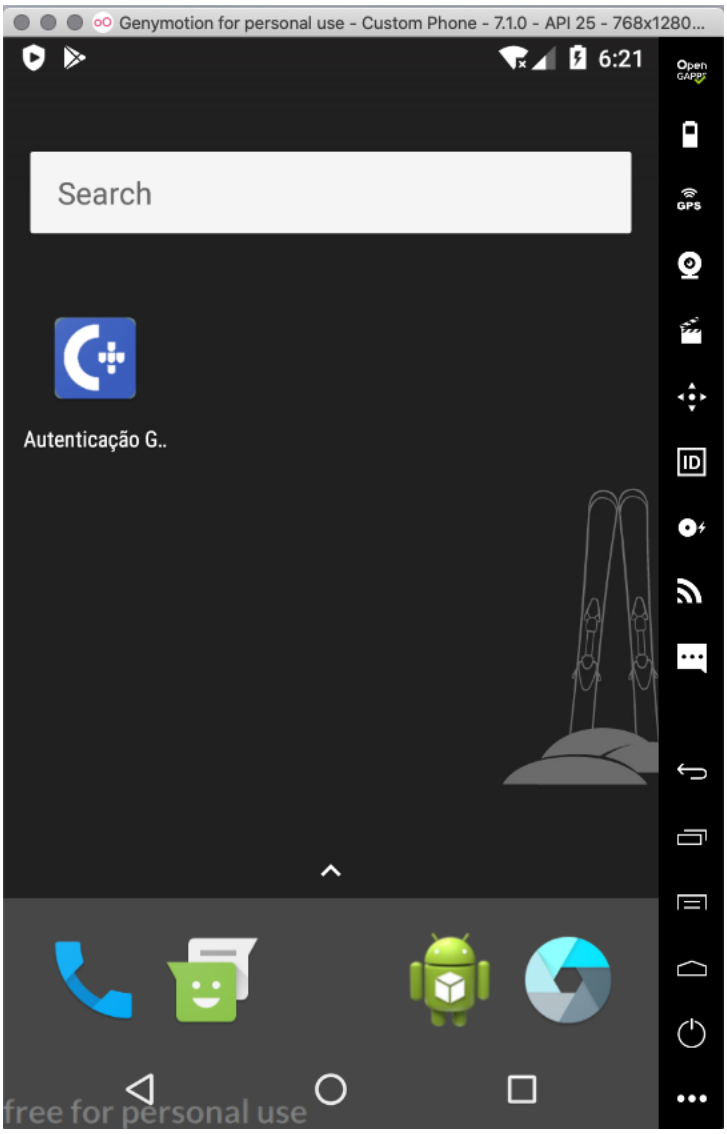


Figura 5-36 - Genymotion com a aplicação Autenticação.Gov

Para efetuar a análise dinâmica uma ferramenta essencial é o Drozer. Esta ferramenta faz parte do Santoku como observamos na Figura 5-37, ou esta poderá ser descarregada através do GitHub e instalada em qualquer distribuição do Linux.

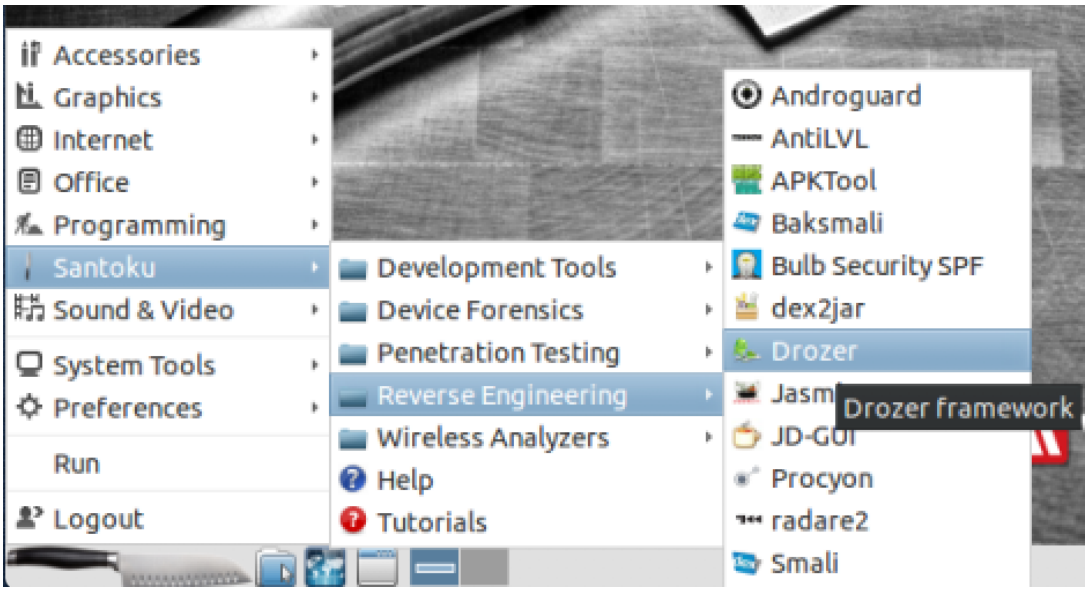


Figura 5-37 – Drozer Santoku

Como já foi citado anteriormente a *framework* Drozer utiliza uma arquitetura cliente-servidor, onde o servidor é disponibilizado na forma de uma APK para ser instalado no dispositivo Android. O Cliente será a máquina virtual onde está instalado o Santoku como podemos visualizar na Figura 5-37.

O Drozer requer dois componentes:

- **Console** – Interface disponível através da linha de comandos que permite interagir com o dispositivo através de um agente.
- **Agent** – Aplicação Android disponibilizada de forma gratuita que quando em execução usa determinado porto para realização dos testes.

Para instalação do agente foi efetuado o download do APK e instalado através dos comandos adb Figura 5-38.

```
1  tiagojoao@santoku-VirtualBox:~$ adb connect 192.168.1.198
2  * daemon not running. starting it now on port 5037 *
3  * daemon started successfully *
4  connected to 192.168.1.198:5555
5  tiagojoao@santoku-VirtualBox:~$ adb install drozer-agent-2.3.4.apk
6  254 KB/s (633111 bytes in 2.432s)
7  Success
8  tiagojoao@santoku-VirtualBox:~$
```

Figura 5-38 - Instalação Drozer Agent

Por defeito, o *Drozer server* encontra-se à "escuta" no porto tcp 31415, como podemos visualizar na

Figura 5-39. É necessário utilizar o *port forwarding* para encaminhar o tráfego gerado para o Santoku através do comando **adb forward tcp:31415 tcp:31415**.

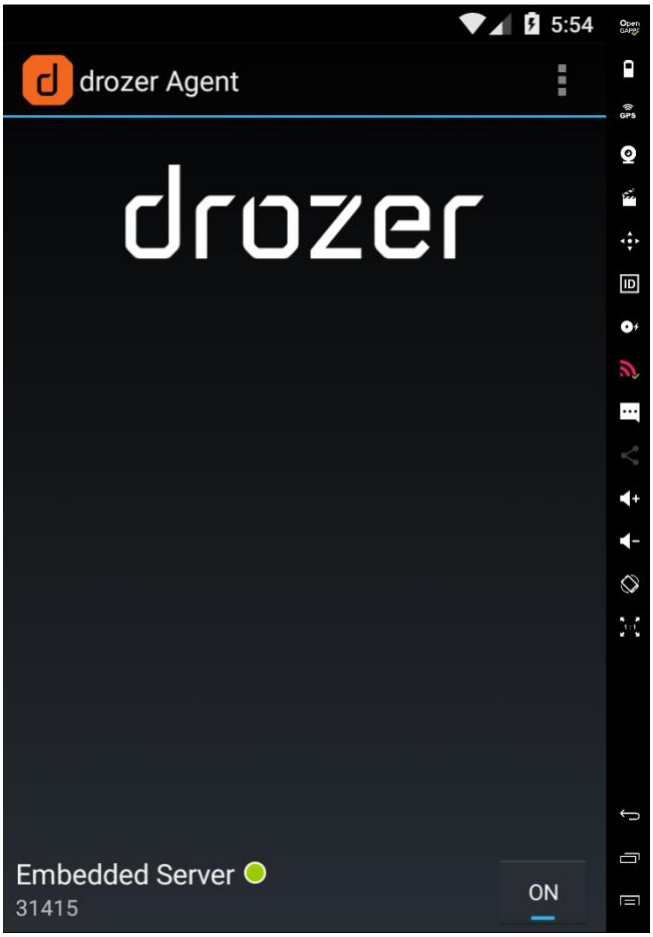


Figura 5-39 - Drozer Server ativo no porto 31415

O *Drozer console* é basicamente uma interface na linha de comandos que permite a execução de módulos instalados na *Framework*. A Figura 5-40 mostra todos os passos necessários para aceder à Drozer console.

```
tiagojoao@santoku-VirtualBox:~$ adb connect 192.168.1.198
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
connected to 192.168.1.198:5555
tiagojoao@santoku-VirtualBox:~$ adb forward tcp:31415 tcp:31415
tiagojoao@santoku-VirtualBox:~$ drozer console connect
Selecting 8e9a88ee490cbd57 (Genymotion Custom Device 7.1.1)

             ..                    ..:.
            ..o..                  .r..
             ..a..  . ....... .  ..nd
               ro..idsnemesisand..pr
               .otectorandroidsneme.
            .,sisandprotectorandroids+.
          ..nemesisandprotectorandroidsn:.
         .emesisandprotectorandroidsnemes..
       ..isandp,..,rotectorandro,..,idsnem.
       .isisandp..rotectorandroid..snemisis.
       ,andprotectorandroidsnemisisandprotec.
      .torandroidsnemesisandprotectorandroid.
      .snemisisandprotectorandroidsnemesisan:
      .dprotectorandroidsnemesisandprotector.

drozer Console (v2.3.3)
dz>
```

Figura 5-40 - Terminal de consola Drozer

Através do comando ***list*** conseguimos visualizar os módulos disponíveis na *Framework* Drozer.

```
1  dz> list
2  app.activity.forintent         Find activities that can handle the given intent
3  app.activity.info              Gets information about exported activities.
4  app.activity.start             Start an Activity
5  app.broadcast.info             Get information about broadcast receivers
6  app.broadcast.send             Send broadcast using an intent
7  app.package.attacksurface      Get attack surface of package
8  app.package.backup             Lists packages that use the backup API (returns true on
9                                 FLAG_ALLOW_BACKUP)
10 app.package.debuggable         Find debuggable packages
11 app.package.info               Get information about installed packages
12 app.package.launchintent       Get launch intent of package
13 app.package.list               List Packages
14 app.package.manifest           Get AndroidManifest.xml of package
15 app.package.native             Find Native libraries embedded in the application.
16 app.package.shareduid          Look for packages with shared UIDs
17 app.provider.columns           List columns in content provider
18 app.provider.delete            Delete from a content provider
19 app.provider.download          Download a file from a content provider that supports
20 (...)
```

Figura 5-41 - Listagem de alguns módulos da framework Drozer

Para analisar a aplicação é necessário saber o nome do seu package. Através do comando **run app.package.list** linha 13 da Figura 5-41 são listados os nomes de todos os *pakages* das aplicações instaladas no dispositivo móvel.

Na Figura 5-42 linha 18, verificamos que o nome do package da aplicação Autenticação.Gov é **pt.ama.autenticacaogov**.

```
dz> run app.package.list
com.android.cts.priv.ctsshim (com.android.cts.priv.ctsshim)
com.google.android.ext.services (Android Services Library)
com.example.android.livecubes (Example Wallpapers)
com.android.providers.telephony (Phone and Messaging Storage)
com.android.providers.calendar (Calendar Storage)
com.android.providers.media (Media Storage)
com.google.android.onetimeinitializer (Google One Time Init)
com.google.android.ext.shared (Android Shared Library)
com.android.wallpapercropper (com.android.wallpapercropper)
com.android.documentsui (Files)
com.android.externalstorage (External Storage)
com.android.htmlviewer (HTML Viewer)
com.android.quicksearchbox (Search)
com.android.mms.service (MmsService)
com.android.providers.downloads (Download Manager)
com.android.messaging (Messaging)
pt.ama.autenticacaogov (Autenticação Gov)
com.google.android.configupdater (ConfigUpdater)
com.android.defcontainer (Package Access Helper)
com.android.providers.downloads.ui (Downloads)
com.android.vending (Google Play Store)
com.android.pacprocessor (PacProcessor)
com.android.certinstaller (Certificate Installer)
com.android.carrierconfig (com.android.carrierconfig)
com.google.android.launcher.layouts.genymotion (Genymotion Home Screen)
com.genymotion.systempatcher (com.genymotion.systempatcher.SystemPatcherApp)
android (Android System)
(...)
dz>
```

Figura 5-42 - Nome dos packages instalados

O Drozer permite obter informações do package. Através do comando run **app.package.info -a** [package] no qual package é o nome da aplicação, é possível adquirir informações como o nome do processo, versão instalada, ID do utilizador, grupo e permissões como podemos visualizar na figura seguinte.

```
dz> run app.package.info -a pt.ama.autenticacaogov
Package: pt.ama.autenticacaogov
  Application Label: Autenticação Gov
  Process Name: pt.ama.autenticacaogov
  Version: 2.0.0
  Data Directory: /data/user/0/pt.ama.autenticacaogov
  APK Path: /data/app/pt.ama.autenticacaogov-1/base.apk
  UID: 10085
  GID: [3003]
  Shared Libraries: null
  Shared User ID: null
  Uses Permissions:
  - android.permission.ACCESS_NETWORK_STATE
  - android.permission.USE_FINGERPRINT
  - android.permission.BIND_NOTIFICATION_LISTENER_SERVICE
  - android.permission.INTERNET
  - android.permission.READ_EXTERNAL_STORAGE
  - android.permission.READ_LOGS
  - android.permission.READ_PHONE_STATE
  - android.permission.VIBRATE
  - android.permission.WAKE_LOCK
  - com.example.proj.permission.C2D_MESSAGE
  - android.permission.WRITE_EXTERNAL_STORAGE
  - pt.ama.autenticacaogov.permission.C2D_MESSAGE
  - com.google.android.c2dm.permission.RECEIVE
  - android.permission.GET_ACCOUNTS
  Defines Permissions:
  - pt.ama.autenticacaogov.permission.C2D_MESSAGE
```

Figura 5-43 - Drozer informações do package pt.ama.autenticacaogov

O *Drozer* possibilita a obtenção da mais variada informação executando os módulos que a *framework* dispõe. O comando list permite listar esses módulos.

5.2.6. Superfície de Ataque

Em termos simples, a superfície de ataque é composta por todo o ambiente de rede de uma organização que um invasor pode tentar explorar para realizar um ataque bem-sucedido, incluindo todos os protocolos, interfaces, softwares e serviços implementados.

No cenário ideal, as equipas de segurança simplesmente reduzem a superfície de ataque a praticamente zero, corrigindo todas as vulnerabilidades conhecidas, atualizando frequentemente todo o hardware e o software, corrigindo todos os erros de configuração em tempo real, entre outras coisas.

A superfície de ataque no contexto da análise de uma aplicação mobile pode ser compreendida como todo e qualquer meio que possa ser utilizado para um ataque, tais como as próprias funcionalidades da aplicação.

Fora as funcionalidades, podemos ver a superfície de ataque executando o Drozer, que tem módulos que funcionam para verificar este tipo de ataque.

Executando o comando **run app.package.attacksurface** [package] no qual package é o nome da aplicação, conseguimos determinar como as aplicações potencialmente maliciosas irão interagir com os componentes de uma determinada aplicação e obter uma visão de alto nível dos componentes exportados pela aplicação. A Figura 5-44 mostra a execução do comado "**run app.package.attacksurface pt.ama.autenticacaogov**".

```
dz> run app.package.attacksurface pt.ama.autenticacaogov
Attack Surface:
  1 activities exported
  1 broadcast receivers exported
  0 content providers exported
  0 services exported
dz>
```

Figura 5-44 - Attack surface à aplicação Autenticação.Gov

Para obtermos uma listagem mais detalhada de todos os componentes exportados utilizamos os seguintes módulos:

- **app.activity.info** – executando o comando **run app.activity.info -a [package]** onde *package* é o nome do *package* do aplicativo, o parâmetro opcional -u ao fim permite mostrar as atividades não exportadas. A Figura 5-45 mostra as atividades exportadas e não exportadas.

```
dz> run app.activity.info -a pt.ama.autenticacaogov -u
Package: pt.ama.autenticacaogov
  Exported Activities:
    md5ba9292671f075a189d6b15232e1c2e4f.MainActivity
  Hidden Activities:
    md5a104545e4d19c4ffe9ec3d5074a3b979.FormAuthenticatorActivity
    md5a104545e4d19c4ffe9ec3d5074a3b979.WebAuthenticatorActivity
dz>
```

Figura 5-45 - Drozer atividade exportadas e não exportadas

Casos as atividades indicadas não tenham nenhuma permissão especial, isso significa que se pode abrir as mesmas fora do fluxo padrão da aplicação, sendo possível fazer testes como burlar o login, verificar se há informações sensíveis que podem ser acedidas, entre outros.

No caso específico deste projeto verificamos que a atividade exportada esta cifrada através de um algoritmo de *hash* MD5, que atualmente está considerando como não seguro, devido ao aumento da capacidade de processamento dos computadores que levam a que os ataques de força bruta sejam realizados de forma cada vez mais rápida (MD5ONLINE, 2020).

- **app.broadcast.info** - Quando algum evento ocorre, *broadcasts* registados para aquele evento específicos são ativados. Eles são componentes responsáveis por receber e tratar mensagens do sistema e de outras aplicações e verificar a existência de e-mails não lidos. Existem dois tipos principais de *broadcast* (Zhao, 2018):
    - Do sistema, que são ações como BOOT_COMPLETE, INPUT_METHOD_CHANGED, AIRPLANE_MODE_CHANGED e POWER_ CONNECTED,
    - Customizados, que podem ser gerados pelas aplicações Android.

A figura abaixo demostra como um atacante se pode aproveitar de um *receiver* para pegar dados ou realizar um *exploit*.

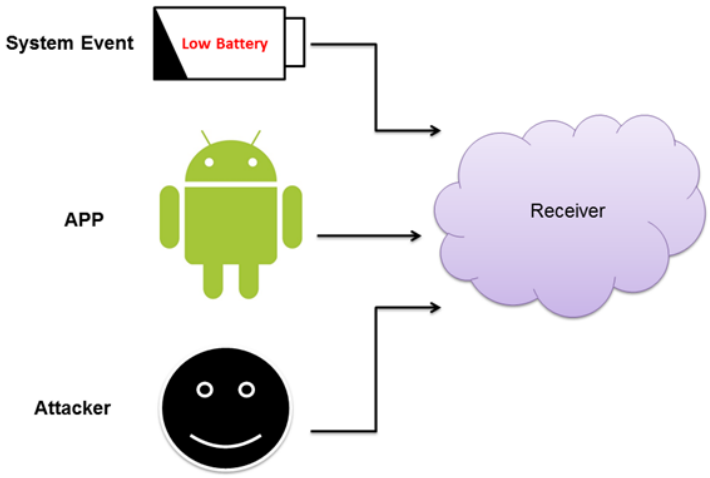


Figura 5-46 - Android Broadcast (Zhao, 2018)

Para visualizamos os *broadcasts* registados, executamos no Drozer o comando **run app.broadcast.info -a [package]**, Figura 5-48, no qual package é o nome do package da aplicação. Em alternativa, podemos encontrar os broadcast receivers exportados encontrados no ficheiro AndroidManifest.xml com a tag <receiver>, como podemos visualizar na Figura 5-47.

```
</activity>
<receiver android:name="md5a7279444c0fe86aa930f169dd7d0faa5.GcmBroadcastReceiver"
android:permission="com.google.android.c2dm.permission.SEND">
    <intent-filter>
```

Figura 5-47 - *Broadcast receivers* no ficheiro *AndroidManifest.xml*

Verificamos que através do ficheiro AndroidManifest.xml ou através da consola do Drozer os *broadcast* exportados encontram-se cifrados através de uma função de *hash* MD5 que atualmente não é segura como foi mencionado anteriormente.

```
1  dz> run app.broadcast.info -a pt.ama.autenticacaogov
2  Package: pt.ama.autenticacaogov
3    Receiver: md5a7279444c0fe86aa930f169dd7d0faa5.GcmBroadcastReceiver
4  dz>
```

Figura 5-48 - Drozer *broadcast receivers*

- **app.service.info** – este irá exibirá todos os serviços utilizados pela aplicação Autenticacao.Gov. Esse é responsável pelas ações em *background*, ou seja, em segundo plano. Também são utilizados para acesso a recursos entre diversas aplicações, como por exemplo o uso de sensores. Estes são identificados no *AndroidManifest.xml* pela tag <service>. No Drozer o comando é: **run app.service.info -a [package]** com o parâmetro opcional -u para incluir os *services* não exportados.

```
dz> run app.service.info -a pt.ama.autenticacaogov -u
Package: pt.ama.autenticacaogov
  Exported Services:
  Hidden Services:
    md5a7279444c0fe86aa930f169dd7d0faa5.GcmService
      Permission: null
dz>
```

Figura 5-49 - Drozer serviços exportados

Na Figura 5-49 assim como a Figura 5-44 volta-se a confirmar que não existem serviços exportados. Ter um *service* disponível sem nenhuma permissão de controle é considerado perigoso, pois este pode ser utilizado por qualquer aplicação maliciosa instalada no mesmo dispositivo móvel.

5.2.7. Execução de um ataque de SQL *Injection* à aplicação autenticao.gov

Efetuar um ataque deste tipo consiste na inserção de uma *query SQ*L na base de dados que pode resultar no furto de informação.

Anteriormente verificou-se que não existe nenhum *content provider* Figura 5-44, a ser exportado, o que significa que a aplicação não usa qualquer base de dados local (Google Developers, 2020). Para validar essa inexistência realizou-se um ataque SQL injection através do Drozer recorrendo ao módulo **run scanner.provider.injection** Figura 5-50.

```
dz> run scanner.provider.injection -a pt.ama.autenticacaogov
Scanning pt.ama.autenticacaogov...
Not Vulnerable:
  content://pt.ama.autenticacaogov.mono.MonoRuntimeProvider.__mono_init__
  content://pt.ama.autenticacaogov.mono.MonoRuntimeProvider.__mono_init__/

Injection in Projection:
  No vulnerabilities found.

Injection in Selection:
  No vulnerabilities found.
dz>
```

Figura 5-50 - Ataque de SQL *Injection* à aplicação Autenticacao.Gov

A aplicação não guarda informação em base de dados locais como verificamos na figura anterior. Verificamos também que não existem vulnerabilidades na aplicação passiveis de ataque *SQL injetion*.

Da figura anterior analisamos igualmente que a aplicação Autenticação.gov faz uso da arquitetura *Xamarin Android* que por sua vez são executados no ambiente de execução Mono que é escrito na linguagem de programação C.

Este ambiente de execução é executado lado a lado com a máquina virtual ART. Estes ambientes virtuais são executados sobre o *kernel Linux* e fornecem várias APIs de código ao utilizador que permite aos programadores acederem ao sistema Android. A figura seguinte mostra um esquema da arquitetura *Xamarin Android* (Chauhan, 2020; Microsoft.com, 2020)

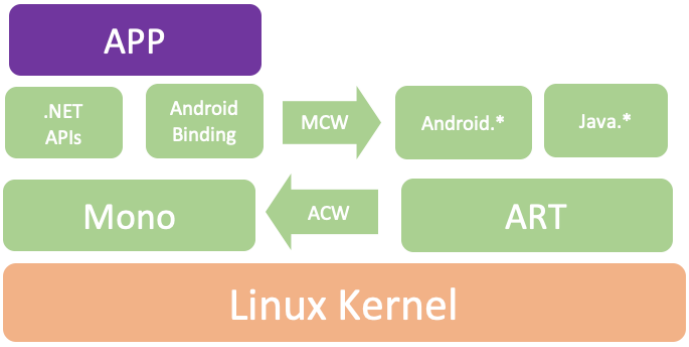


Figura 5-51 - Arquitetura Xamarin Android adaptado de (Microsoft.com, 2020)

5.2.8. Debuggable da aplicação Autenticacao.Gov

No momento do desenvolvimento de uma aplicação móveis esta pode ser marcada como *debuggable* para facilitar a fase de testes e a deteção de erros, permitindo que sejam definidos *breakpoints* durante a execução.

É recorrente algumas aplicações serem publicadas na *store* Google Play ainda como *debuggable*. Isto significa que invasores poderão obter acesso as informações de autenticação ou dados guardados por outras aplicações de uma forma muito facilitada. Invasores podem também acionar a execução de código remoto por meio do aplicativo e executar algum código no contexto de aplicativos.

O Drozer através do comando **app.package.debuggable** permite listar todas as aplicações que se encontram marcadas como *debuggable*.

```
1  dz> run  app.package.debuggable
2  Package: com.mwr.dz
3    UID: 10083
4    Permissions:
5      - android.permission.INTERNET
6
7  dz>
```

Figura 5-52 - Aplicações *debuggable* instaladas no dispositivo

A Figura 5-52 mostra as aplicações que se encontram em modo *debuggable*.

Podemos concluir que o package pt.ama.autenticacaogov não é *debuggable*, pois não consta na lista.

5.2.9. Análise dos logs da aplicação Autentição.Gov

As aplicações Android podem armazenar alguns dados como logs. Os logs são guardados num repositório central e todas as aplicações Android instaladas no dispositivo móvel podem aceder a esses registos. Nas versões mais recentes do Android, as permissões para aceder a esses logs foram alteradas. O armazenamento de informações sensíveis do utilizador não deve ser permitido. As informações pode ser desde *tokens* de autenticação ou ID's de sessão ou mesmo credenciais de acesso.

Cada aplicação Android é executada no seu próprio processo *Linux*. De modo a filtrar a informação apenas sobre a aplicação pretendida é necessário saber qual o Process Identifier (PID) da aplicação. Será então necessário aceder a *Shell* do dispositivo móvel e visualizar os processos em execução. Na figura seguinte podemos visualizar o comando executado. O PID do processo encontra-se na segunda coluna que é o 4631.

```
1  ~$ adb shell ps | grep pt.ama.autenticacaogov
2  u0_a86    4631  265   997464 155560    ep_poll f62c9bb9 S pt.ama.autenticacaogov
3  ~$
```

Figura 5-53 - PID da aplicação Autenticação.Gov

Para que fosse gerado informação de log a aplicação foi instalada novamente de modo a ser iniciada a sessão na aplicação. Através do comando *adb logcat* foi possível ter acesso ao log da aplicação (Zhao, 2017).

```
~$ adb logcat | grep 4631
10-06 13:36:13.837  4631  4631 W art     : Unexpected CPU variant for X86 using defaults: x86
10-06 13:36:13.845   574   585 I ActivityManager: Start proc 4631:pt.ama.autenticacaogov/u0a87 for activity pt.ama.autenticacaogov/md5ba9292671f075a189d6b15232e1c2e4f.MainActivity
10-06 13:36:13.867  4631  4631 W monodroid: Trying to load sgen from: /data/app/pt.ama.autenticacaogov-1/lib/x86/libmonosgen-2.0.so
10-06 13:36:13.933  4631  4631 W monodroid-gc: GREF GC Threshold: 46080
10-06 13:36:13.937  4631  4631 W monodroid: Calling into managed runtime init
10-06 13:36:14.614  4631  4631 W art     : Before Android 4.1, method android.graphics.PorterDuffColorFilter
android.support.graphics.drawable.VectorDrawableCompat.updateTintFilter(android.graphics.PorterDuffColorFilter, android.content.res.ColorStateList, android.graphics.PorterDuff$Mode)
would
have incorrectly overridden the package-private method in android.graphics.drawable.Drawable
10-06 13:36:14.695  4631  4631 V FingerprintManager: FingerprintManagerService was null
10-06 13:36:14.695  4631  4631 W FingerprintManager: isFingerprintHardwareDetected(): Service not connected!
10-06 13:36:14.697  4631  4631 W X:MainActivity: Não foi detetado leitor de impressões digitais
10-06 13:36:15.131  4631  4631 W art     : JNI RegisterNativeMethods: attempt to register 0 native methods for md5b60ffeb829f638581ab2bb9b1a7f4f3f.Platform_DefaultRenderer
10-06 13:36:15.136  4631  4631 W art     : JNI RegisterNativeMethods: attempt to register 0 native methods for md5b60ffeb829f638581ab2bb9b1a7f4f3f.ImageRenderer
10-06 13:36:15.163  4631  4631 W art     : JNI RegisterNativeMethods: attempt to register 0 native methods for md5b60ffeb829f638581ab2bb9b1a7f4f3f.LabelRenderer
10-06 13:36:15.279  4631  4631 W art     : JNI RegisterNativeMethods: attempt to register 0 native methods for md5b60ffeb829f638581ab2bb9b1a7f4f3f.ActivityIndicatorRenderer
10-06 13:36:15.292  4631  4631 W art     : JNI RegisterNativeMethods: attempt to register 0 native methods for md5270abb39e60627f0f200893b490a1ade.ViewRenderer_2
10-06 13:36:15.320  4631  4631 D X:MainActivity: OnResume
10-06 13:36:15.341  4631  4631 D         : static HostConnection *HostConnection::createUnique(): call
10-06 13:36:15.341  4631  4631 D         : HostConnection::get() New Host Connection established 0xe4b90c90, tid 4631
10-06 13:36:15.344  4631  4631 D         : HostComposition ext GL_OES_vertex_array_object ANDROID_EMU_gles_max_version_2
10-06 13:36:15.344  4631  4631 W         : Process pipe failed
10-06 13:36:15.349  4631  4631 W art     : JNI RegisterNativeMethods: attempt to register 0 native methods for md5a7279444c0fe86aa930f169dd7d0faa5.GcmBroadcastReceiver
10-06 13:36:15.360  4631  4631 W art     : JNI RegisterNativeMethods: attempt to register 0 native methods for md5a7279444c0fe86aa930f169dd7d0faa5.GcmService
10-06 13:36:15.398  4631  4661 D libEGL  : Emulator has host GPU support, qemu.gles is set to 1.
10-06 13:36:15.399  4631  4661 E libEGL  : load_driver(/system/lib/egl/libGLES_emulation.so): dlopen failed: library "/system/lib/egl/libGLES_emulation.so" not found
10-06 13:36:15.401  4631  4661 D libEGL  : loaded /system/lib/egl/libEGL_emulation.so
10-06 13:36:15.401  4631  4661 D libEGL  : loaded /system/lib/egl/libGLESv1_CM_emulation.so
10-06 13:36:15.411  4631  4661 D libEGL  : loaded /system/lib/egl/libGLESv2_emulation.so
10-06 13:36:15.435  4631  4661 D         : HostConnection::get() New Host Connection established 0xf5c230e0, tid 4661
10-06 13:36:15.438  4631  4661 D         : HostComposition ext GL_OES_vertex_array_object ANDROID_EMU_gles_max_version_2
10-06 13:36:15.451  4631  4661 I OpenGLRenderer: Initialized EGL, version 1.4
10-06 13:36:15.451  4631  4661 D OpenGLRenderer: Swap behavior 1
10-06 13:36:15.451  4631  4661 W OpenGLRenderer: Failed to choose config with EGL_SWAP_BEHAVIOR_PRESERVED, retrying without...
10-06 13:36:15.451  4631  4661 D OpenGLRenderer: Swap behavior 0
10-06 13:36:15.454  4631  4661 D EGL_emulation: eglCreateContext: 0xf5c058a0: maj 2 min 0 rcv 2
10-06 13:36:16.037  4631  4631 I Choreographer: Skipped 30 frames!  The application may be doing too much work on its main thread.
10-06 13:36:41.346  4631  4631 W IInputConnectionWrapper: finishComposingText on inactive InputConnection
10-06 13:36:47.981  4631  4631 W IInputConnectionWrapper: finishComposingText on inactive InputConnection
10-06 13:37:21.063  4631  4631 W IInputConnectionWrapper: finishComposingText on inactive InputConnection
10-06 13:37:23.433  4631  4631 I art     : Starting a blocking GC Explicit
10-06 13:37:23.439  4631  4631 I art     : Explicit concurrent mark sweep GC freed 1418(62KB) AllocSpace objects, 0(0B) LOS objects, 39% free, 5MB/9MB, paused 122us total 6.130ms
10-06 13:40:01.386  4631  4631 D X:MainActivity: OnPause
10-06 13:44:31.438  4631  4631 D X:MainActivity: OnResume
10-06 13:46:10.355   574  1047 I ActivityManager: Killing 4631:pt.ama.autenticacaogov/u0a87 (adj 900): remove task
10-06 13:46:10.378   574   731 D ActivityManager: cleanUpApplicationRecord -- 4631
```

Figura 5-54 - Log capturado através do comando adb logcat

A Figura 5-54 exibe a execução do comado **adb logcat | grep 4631**. Ao mesmo tempo no dispositivo móvel está a ser executada a aplicação Autenticação.Gov. Durante o tempo de execução da aplicação não foi possível visualizar quaisquer dados pessoais ou mensagens de *debug*. A aquisição de *logging* capturada não foi evidenciado nenhuma informação que comprometa a aplicação.

5.2.10. Ofuscação no código-fonte

A compilação de código utilizando a engenharia reversa é um processo difícil e demorado. A análise é bastante dificultada quando o código é desenvolvido e construído deliberadamente para dificultar o processo de análise. A ofuscação de código é todo o conjunto de técnicas utilizadas com o objetivo de dificultar ou impossibilitar a análise de código (Dang et al., 2014).

Importa compreender que a ofuscação não é exclusivamente utilizada para beneficiar o *malware*. Um outro exemplo onde a ofuscação é aplicada é na proteção de propriedade intelectual. Muitos programas comerciais têm algum tipo de proteção contra cópias não autorizadas. Alguns sistemas empregam ofuscação com o propósito de ocultar detalhes na instalação de certas componentes do sistema (Dang et al., 2014).

Os métodos de ofuscação incluem reordenação de código, transformações para substituir nomes de identificadores significativos no código original por nomes aleatórios sem sentido (renomeação de identificadores), inclusão de código sem significado (lixo eletrónico),

reatribuição variáveis, codificação de *strings*, geração de código de nível médio falso, supressão de constantes, ramificação de controlo de fluxos, entre outros.

Ao realizar a descompilação da aplicação Autenticação.gov chegou-se a conclusão de que o ficheiro jar obtido não contem código. Após a abertura deste na aplicação JD-GUI é possível concluir que esta não contem código que comprometa a aplicação. Na Figura 5-55 é possível visualizar o código obtido e a única classe disponível R.class.

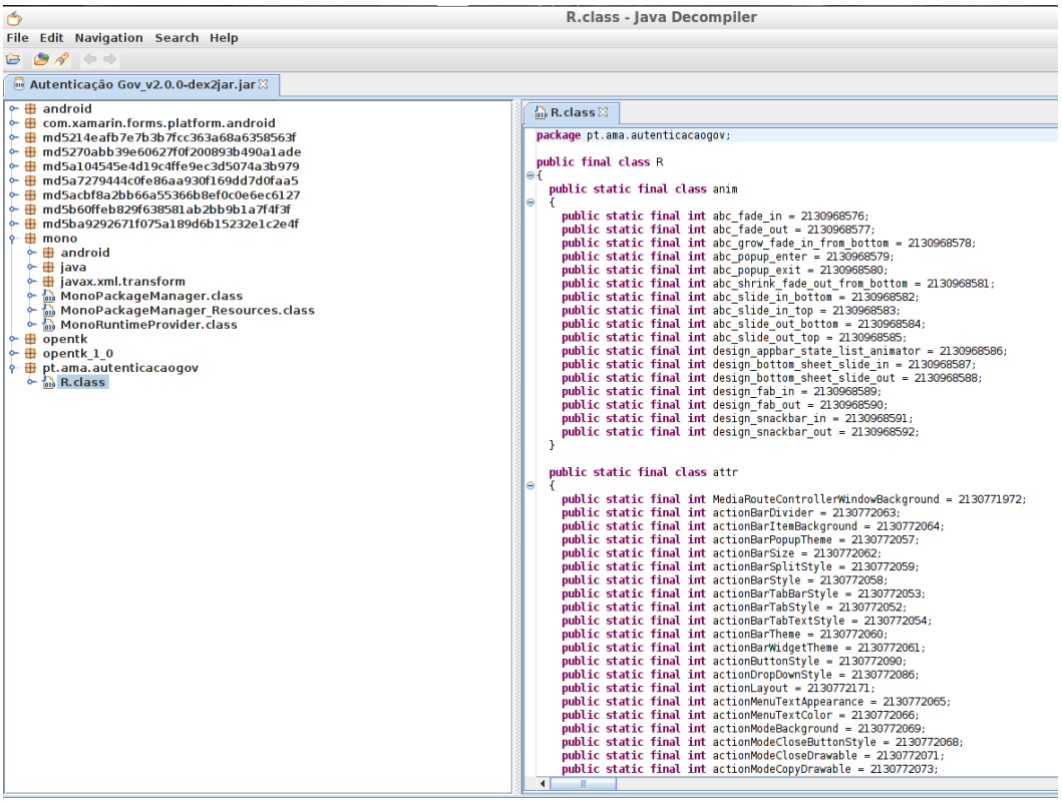


Figura 5-55 – Extrato de Código da aplicação Autenticao.GOV

Isto acontece porque a aplicação Autenticação.gov foi desenvolvida numa plataforma agnóstica ao sistema operativo. Quando a aplicação é compilada é definido o sistema operativo a qual a aplicação se destina, Android ou iOS.

5.2.11. Código Fonte

Não sendo possível ter acesso ao código fonte através da abertura do ficheiro .jar, procurou-se a obtenção deste através de outros métodos. Foi efetuada uma pesquisa no GitHub sobre a aplicação Autenticação.gov, no entanto, não foi encontrada informação relevante para este projeto.

Anteriormente quando foi efetuado ataque de *SQL injection* foi descrito que a aplicação Autenticação.gov fazia uso da plataforma *Xamarin Android*. Esta plataforma possibilita o desenvolvimento de aplicações noutras linguagens de programação Figura 5-51 nomeadamente .NET ou c#.

O software Telerik JustDecompile disponível para download em www.telerik.com permite visualizar código C# e .NET. Este software só esta disponível para SO Windows, e faz o mesmo que o JD-GUI mas para outras linguagens.

Para aceder ao código descompactou-se o ficheiro APK da aplicação Autentição.gov. A Figura 5-56 mostra os ficheiros e pastas que estavam no APK Autenticação.gov.

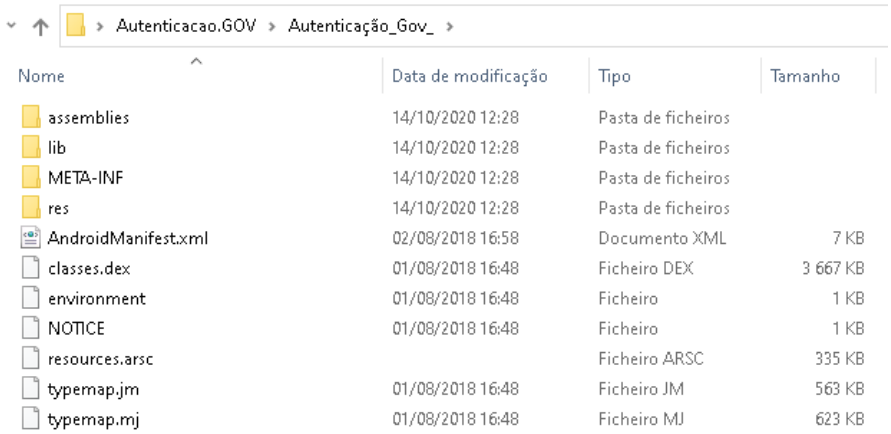


Figura 5-56 – Ficheiros e pastas do APK autenticação.gov

Através do software Teleric JustDecompile abriram-se os ficheiros .dll da package ama.AutenticacaoGov da pasta assemblies Figura 5-57.

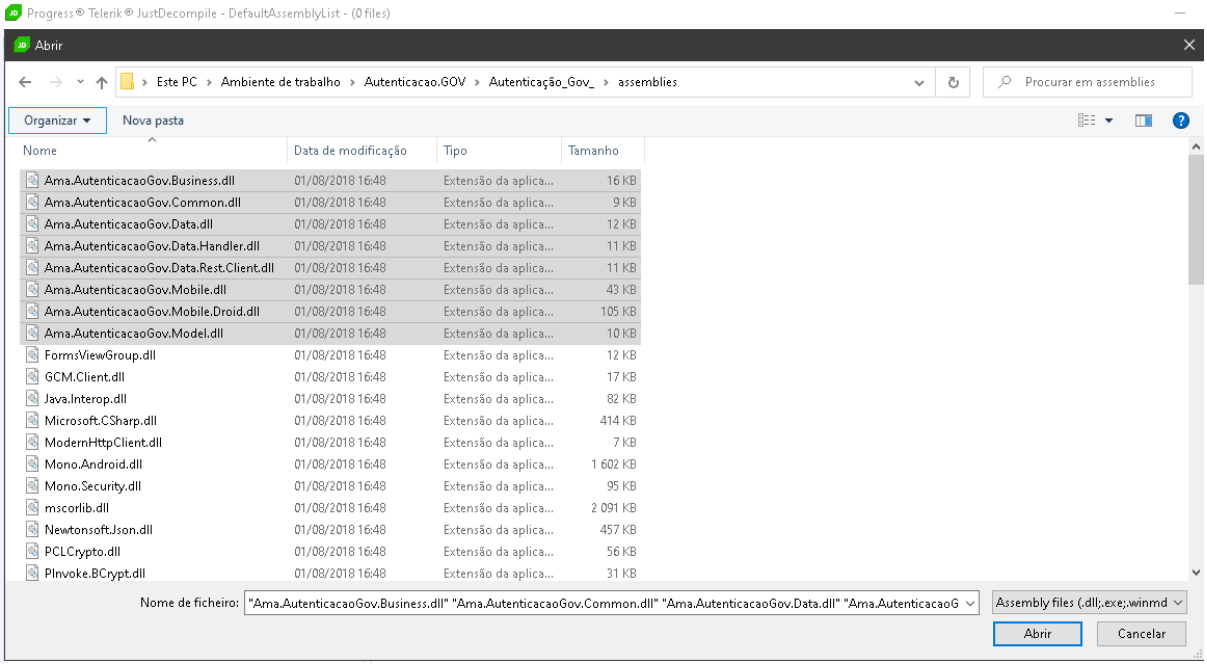


Figura 5-57 - Ficheiros dll da pasta assemblies

O acesso aos ficheiros .dll foi conseguido como demonstra a Figura 5-58.

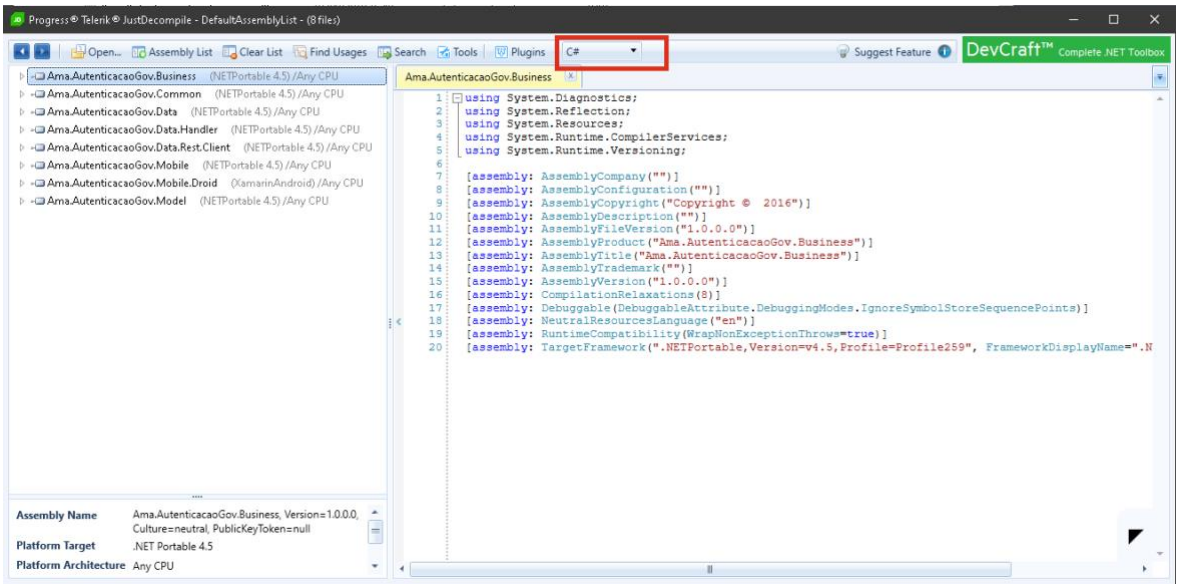


Figura 5-58 - Acesso ao código da aplicação Autenticação.gov

Ao ser efetuada uma análise ao código-fonte obtido, facilmente concluímos que este não apresenta nenhum tipo de ofuscação estando vulnerável a possíveis ataques. A ofuscação é utilizada com a intenção de dificultar a leitura do código-fonte, ou quando se pretende realizar ações mal-intencionadas.

Assim o acesso para criação de uma aplicação maliciosa fica disponível, fazendo pequenas alterações no código-fonte obtido, pois todas as classes se encontram legíveis assim como é possível ter acesso à logica da aplicação.

A Figura 5-59 mostra o acesso ao código-fonte onde é possível de visualizar o nome das classes assim como a estrutura usada.

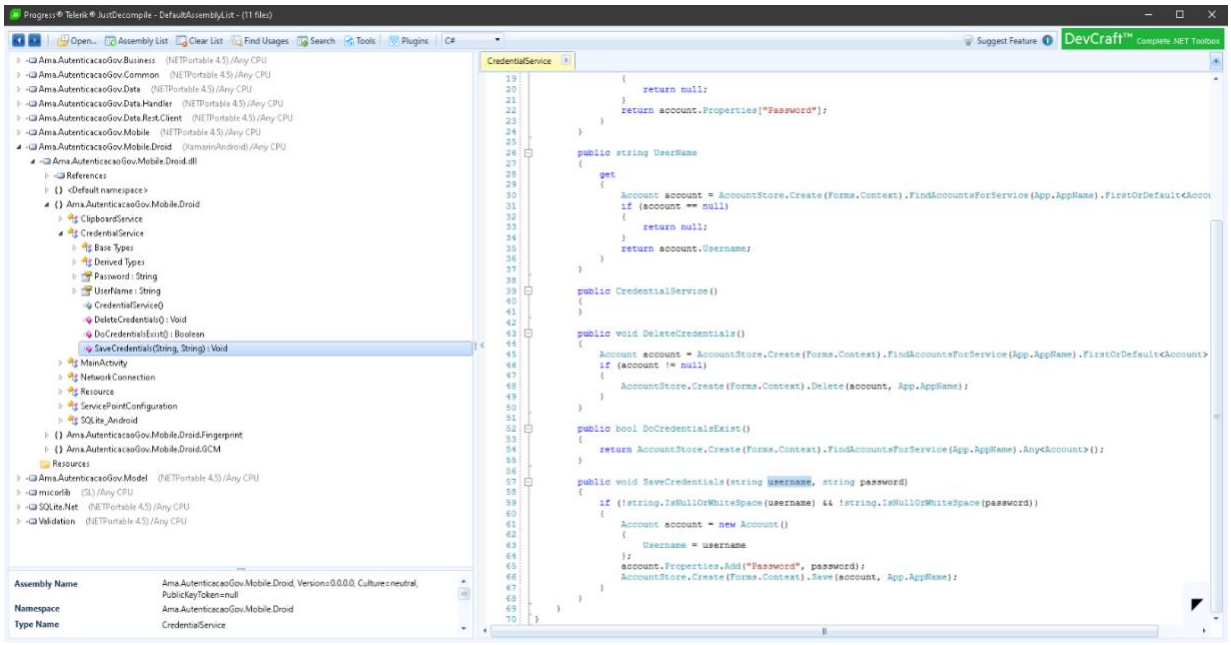


Figura 5-59 – Autenticação.gov código-fonte

Foi realizada uma pesquisa no software Teleric JustDecompile pela palavra password. A Figura 5-60 mostra o resultado. Na pesquisa efetuada não foram encontradas passwords. Verificou-se também o uso de uma base de dados *sqlite.NET* como mostra a figura anterior.

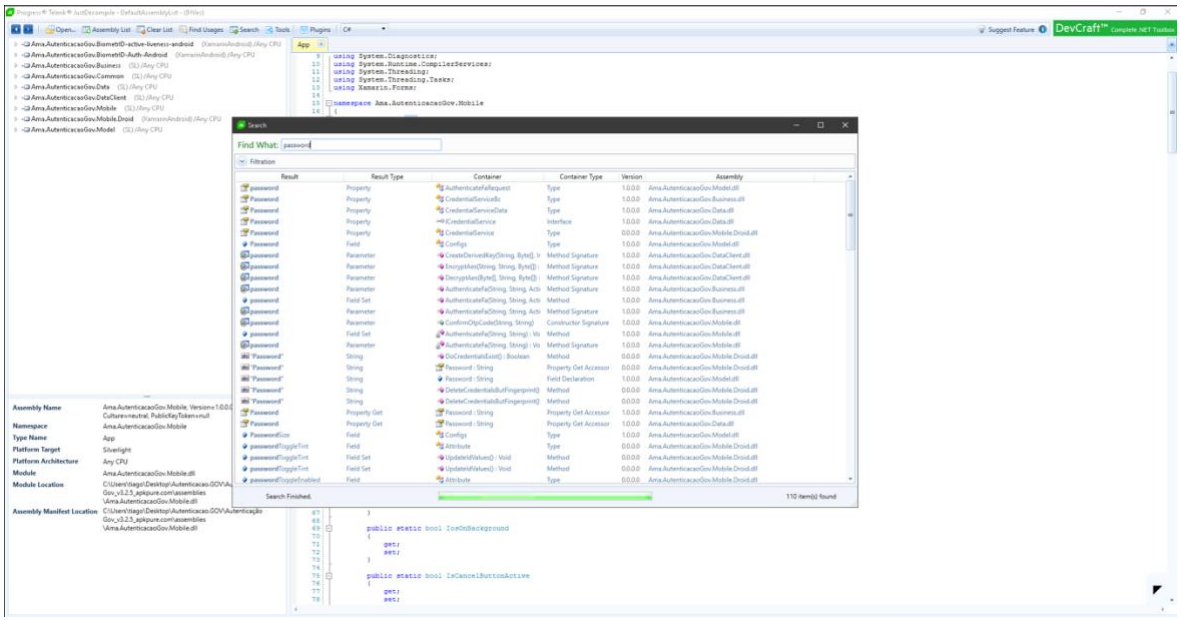


Figura 5-60 - Pesquisa password

Foi realizado o mesmo processo de abertura de ficheiros através do APK para a versão mais recente disponível à data, a versão 3.2.5.

Verifica-se que o nome dos ficheiros dll mudaram com a inserção de novas características como a biometria, no entanto, o código-fonte mantêm-se acessível Figura 5-61.

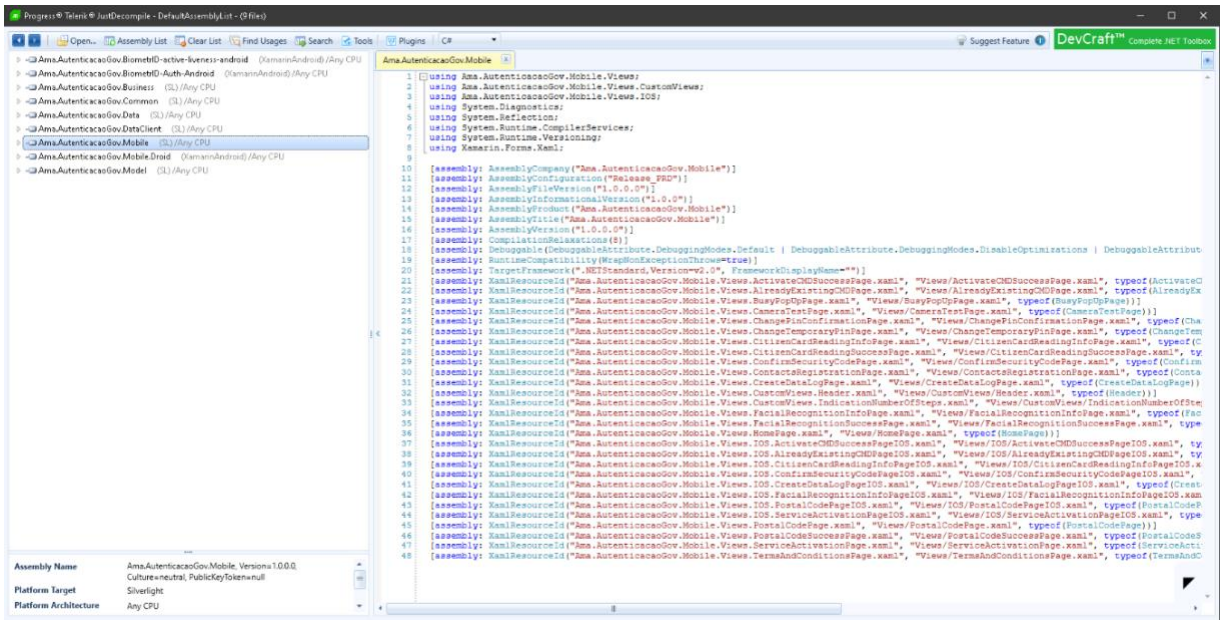


Figura 5-61 - Código-fonte versão 3.2.5

5.3. Assinatura com a aplicação Autenticação.gov

Pretende-se efetuar uma comparação entre dois métodos de autenticação através da aplicação Autenticação.gov.

Numa primeira fase vai-se realizar a autenticação de um documento através da Chave Móvel Digital. Utilizando o mesmo documento também se vai realizar a autenticação com a assinatura eletrónica existente no Cartão de Cidadão.

O objetivo principal de teste será efetuar uma comparação entre os dois métodos de autenticação: Chave Móvel Digital e Cartão de Cidadão.

5.3.1. Cenário de testes

Foi elaborado um segundo cenário para realização de testes. Este cenário, engloba os seguintes componentes:

- Windows 7 com leitor de cartões;
- Aplicação *Procdot* e *process monitor*;
- Aplicação do Cartão de Cidadão versão 3.3.0.

O cenário usado está exibido na Figura 5-62

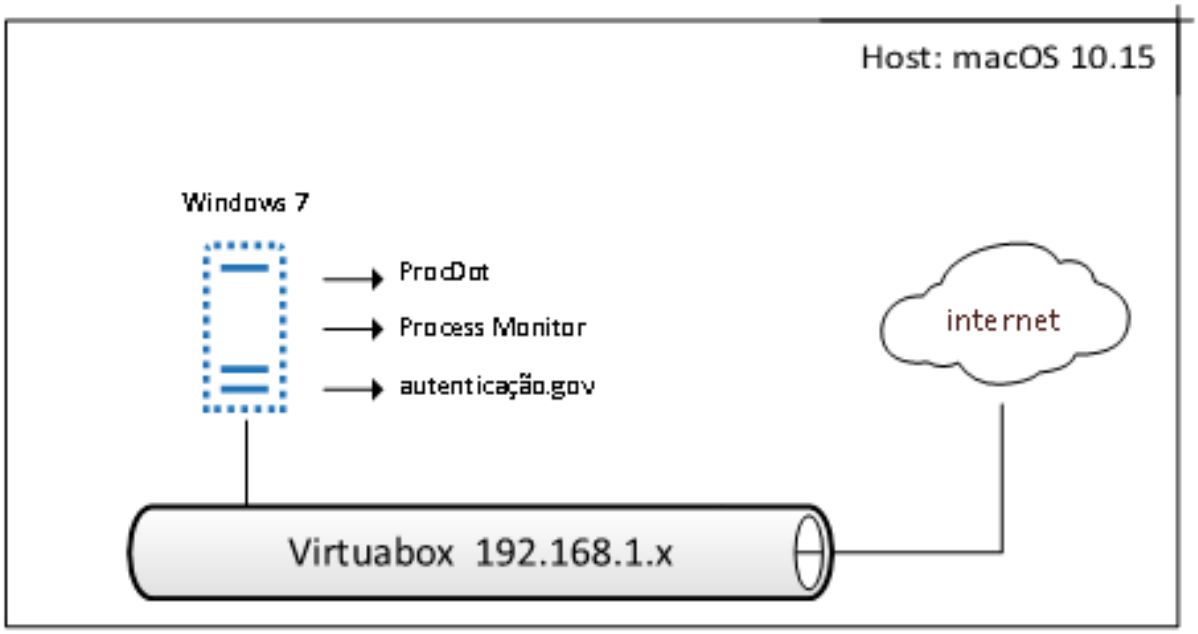


Figura 5-62 - Cenário teste da aplicação

5.3.2. Assinatura com Cartão de Cidadão

O processo de assinatura de um documento é fácil. Precisamos de usar o programa Autenticação.gov selecionar o ficheiro pretendido e colocar o PIN da assinatura. Concluído o processo é possível verificar a assinatura originada. A Figura 5-63 exibe o certificado obtido através da assinatura com o Cartão de Cidadão. Nesta imagem está implícito as entidades certificadoras mencionadas no ponto 5.1.2.

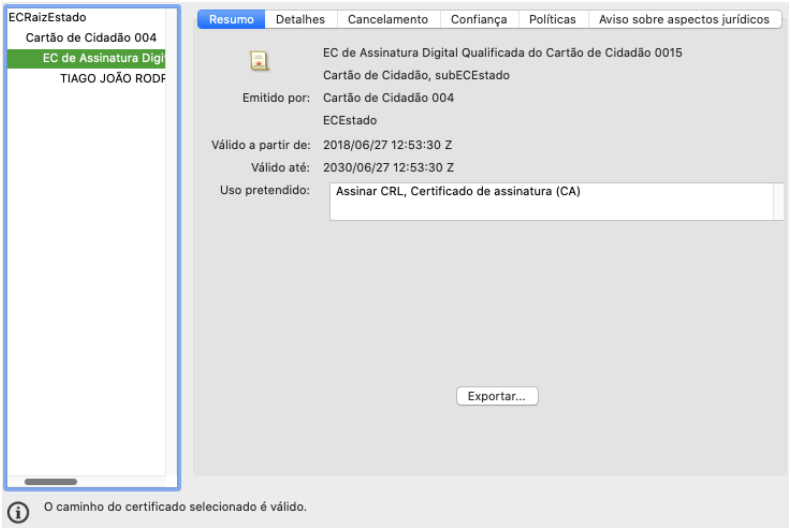


Figura 5-63 – Certificado Assinatura Cartão de Cidadão

Através da assinatura realizada exportamos o certificado de modo a poder verificar algumas informações presentes. A Figura 5-64 exibe informação presente no certificado.

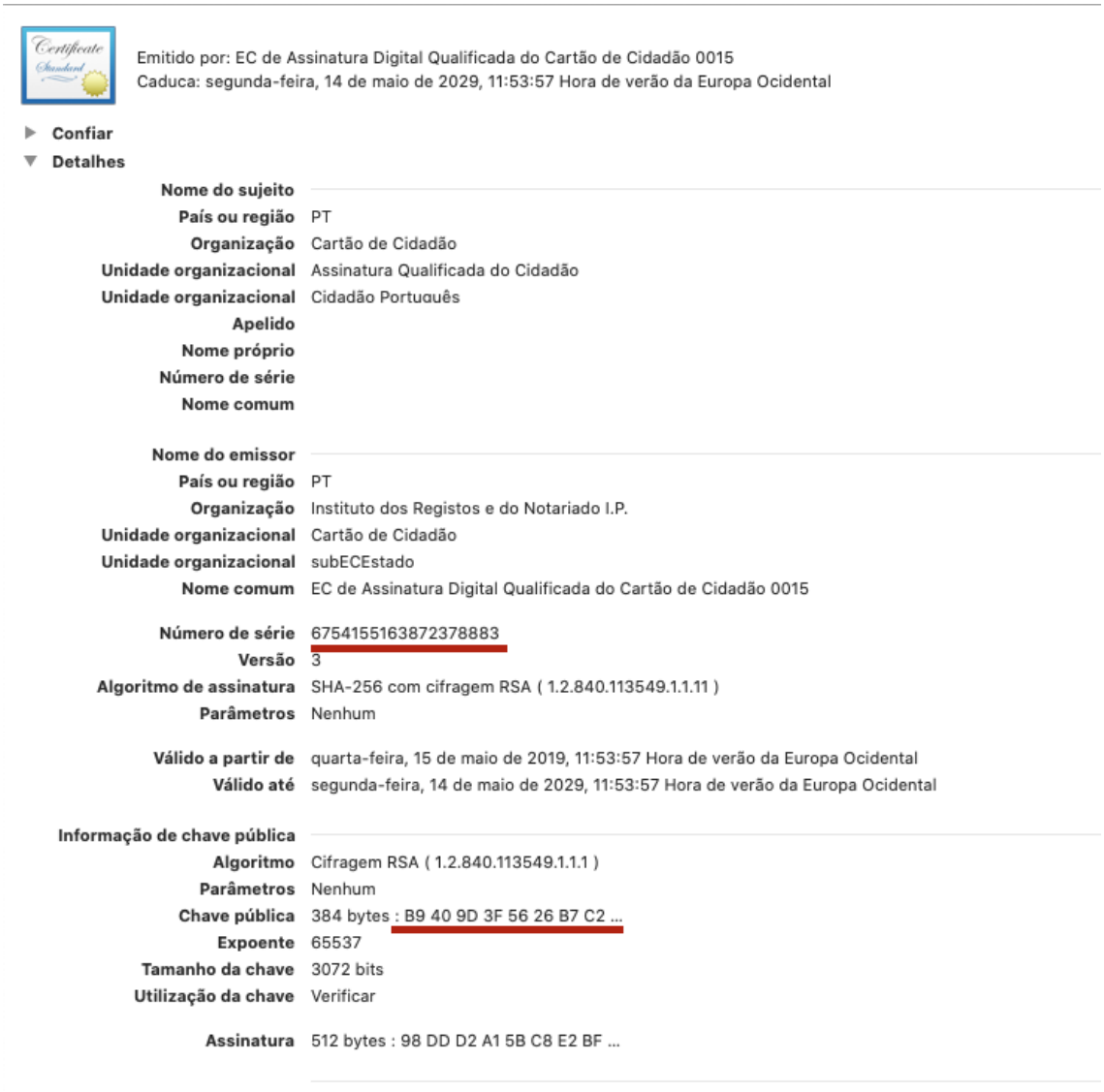


Figura 5-64 – Informações do certificado do Cartão de Cidadão

As informações pessoais foram omitidas, todavia, é possível verificar na informação exibida como o número de série do certificado e informações relativas a chave pública.

De modo a validar as alterações efetuadas no computador no momento do uso da aplicação Autenticação.gov para assinar um documento, foi usado o programa Procdot. Este programa irá registar todas as alterações no registo, comunicações assim como a criação de novos ficheiros.

O resultado obtido numa versão simplificada, está exibido na Figura 5-65. Nesta é possível visualizar a existências de acesso a servidores externos assim como ao *registry*. O Anexo B – ProcDot - Registo de alterações durante a assinatura com o Cartão de Cidadão demostra todos as ligações efetuadas.

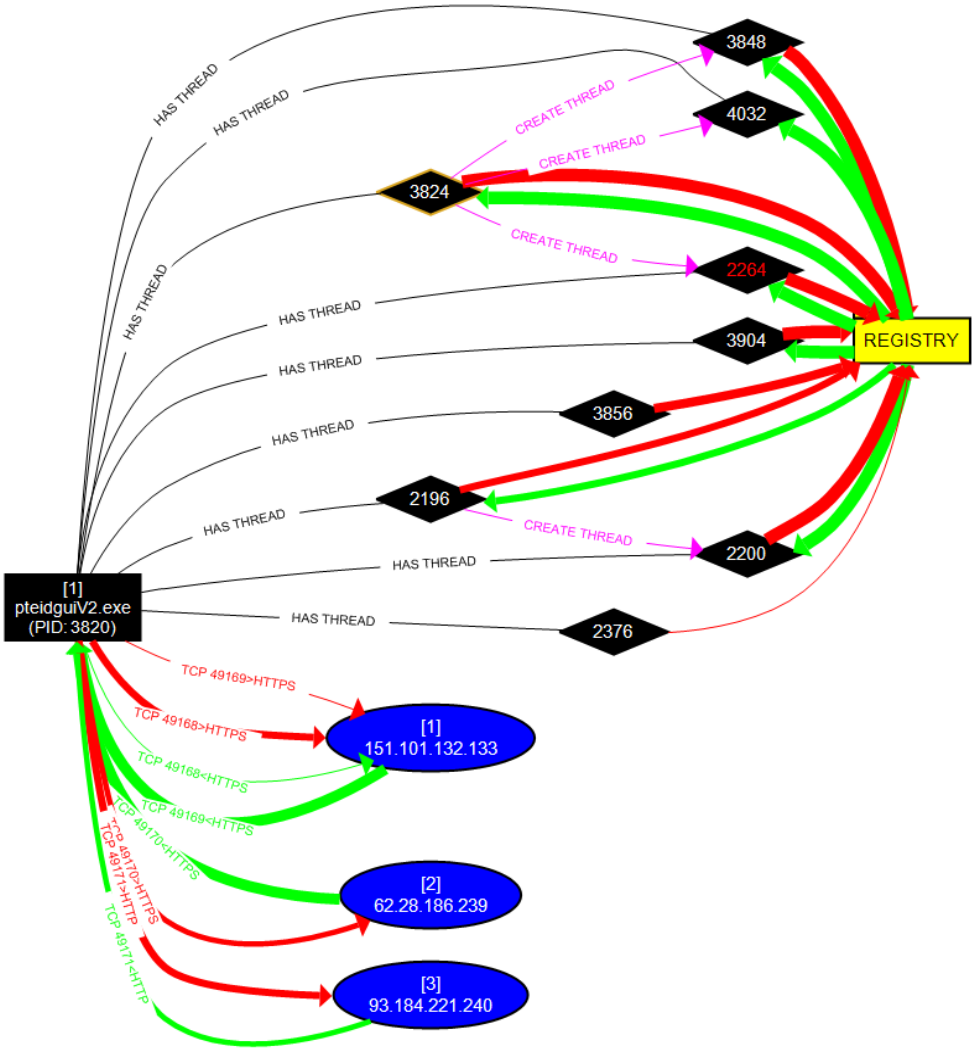


Figura 5-65 - Procdot registo alterações durante a assinatura com o Cartão de Cidadão

Para verificar a necessidade de comunicação com servidores externos, o acesso à Internet foi retirado e o teste novamente efetuado.

A Figura 5-66 exibe as alterações efetuadas durante o processo de assinatura de um documento com o Cartão de Cidadão sem acesso à Internet. Verifica-se a inexistência de ligações ao exterior e a assinatura é finalizada com êxito.

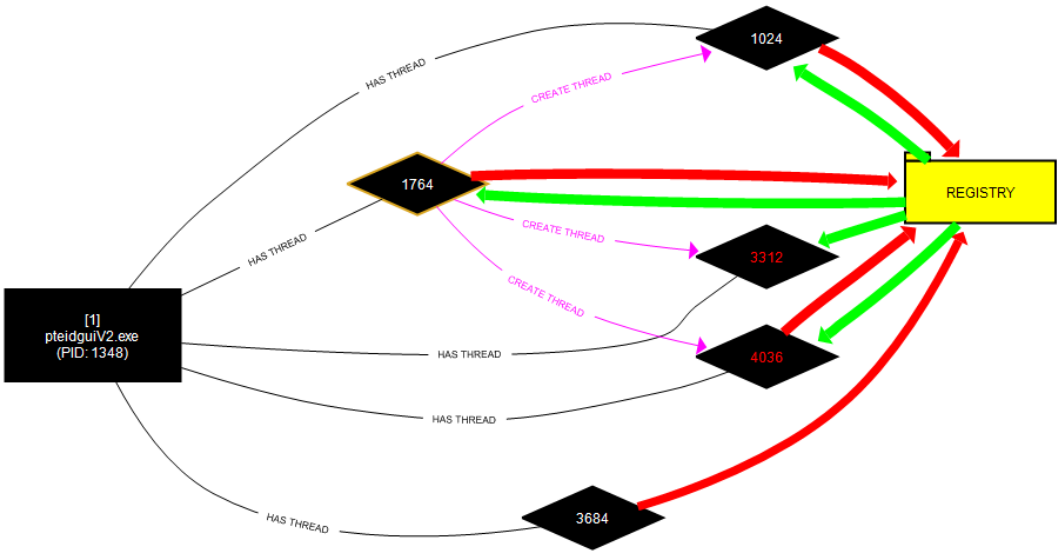


Figura 5-66 - Procdot registo alterações durante a assinatura com o Cartão de Cidadão sem Internet

5.3.3. Assinatura com a Chave Móvel Digital

O processo de assinatura através da CMD é similar ao processo de assinar com o CC, bastando selecionar no programa Assinar com Chave Móvel Digital e colocar o número de telemóvel e o respetivo PIN. Durante o processo será necessário introduzir o código gerado pela aplicação Autenticação.gov no dispositivo móvel ou o SMS enviado.

A Figura 5-67 exibe o certificado obtido através da assinatura com a CMD. Nesta imagem está implícito as entidades certificadoras. A hierarquia é diferente da exibida anteriormente com o Cartão de Cidadão.

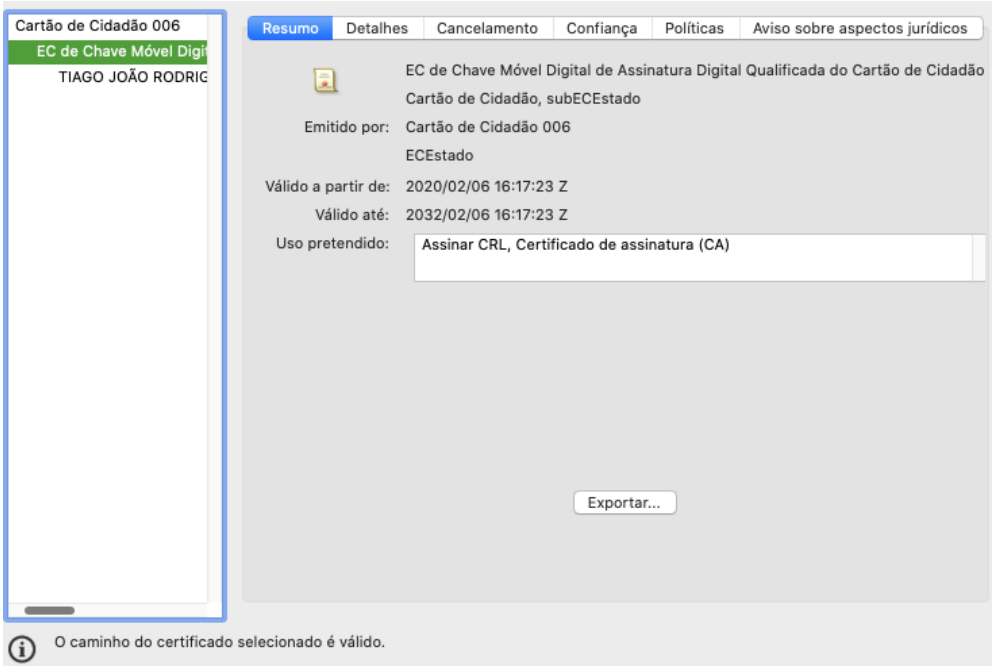


Figura 5-67 - Certificado Assinatura CMD

Exportamos o certificado para que possamos verificar a informação presente neste. A Figura 5-68 exibe os detalhes do certificado após assinatura com a Chave Móvel Digital. Na mesma foi omitida a informação pessoal, contudo da informação exibida destacamos o número de série do certificado e informações relativas à chave pública.

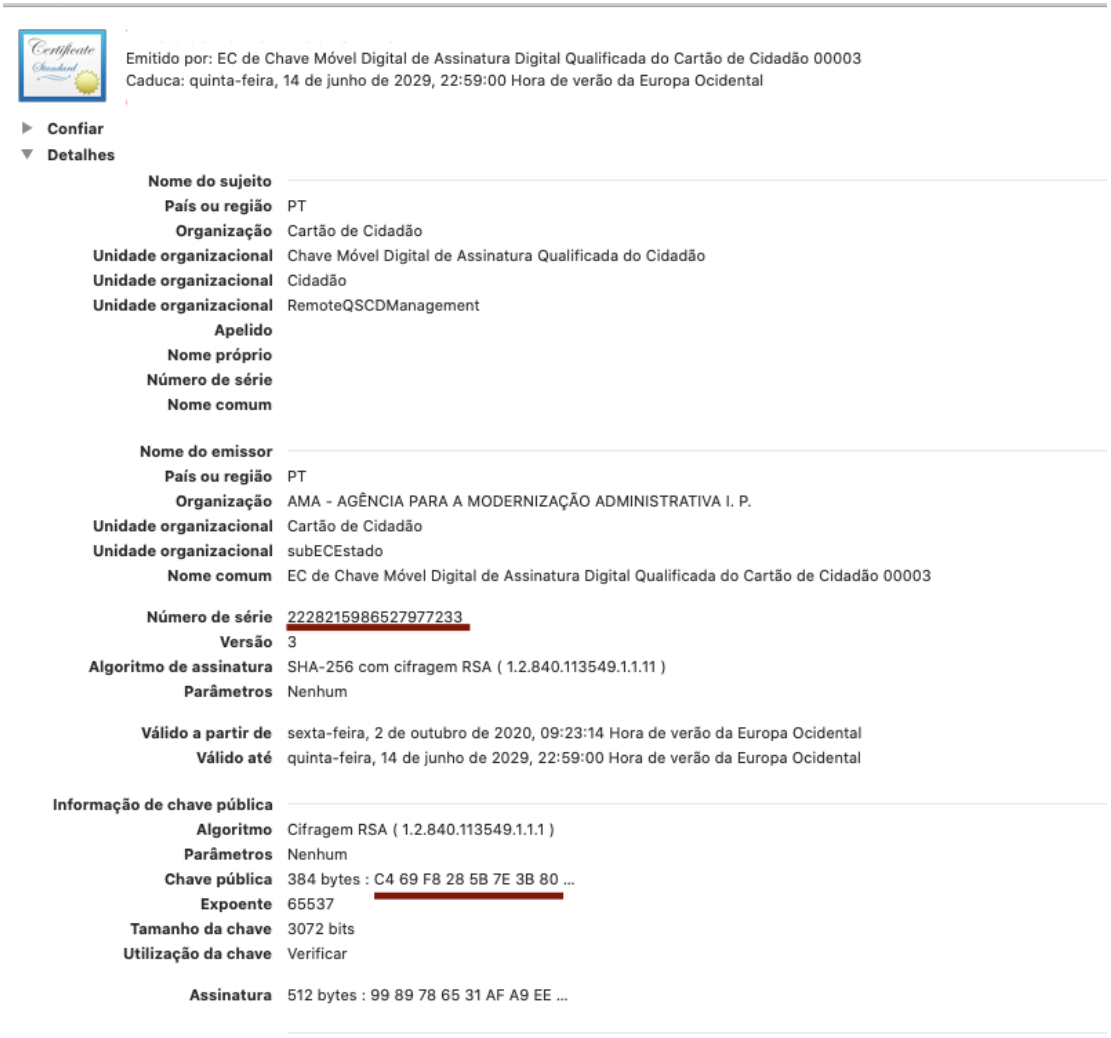


Figura 5-68 - Informações do certificado da Chave Móvel Digital

Para assinar um documento através da CMD é necessário existir um acesso á Internet para que a comunicação com os servidores seja possível. Esta condição não é necessária ao efetuar a assinatura através do CC, no entanto, verificou-se acesso a servidores externos. A Figura 5-69 mostra de uma forma resumida o registo efetuado no momento da assinatura através da CMD.

O acesso a servidores externos é evidente, no entanto, os endereços IP dos servidores não são rigorosamente os mesmos, assim como a ordem como a comunicação é estabelecida durante o processo.

O resultado deste teste de forma expandida está no Anexo C – ProcDot - Registo de alterações durante a assinatura com a CMD.

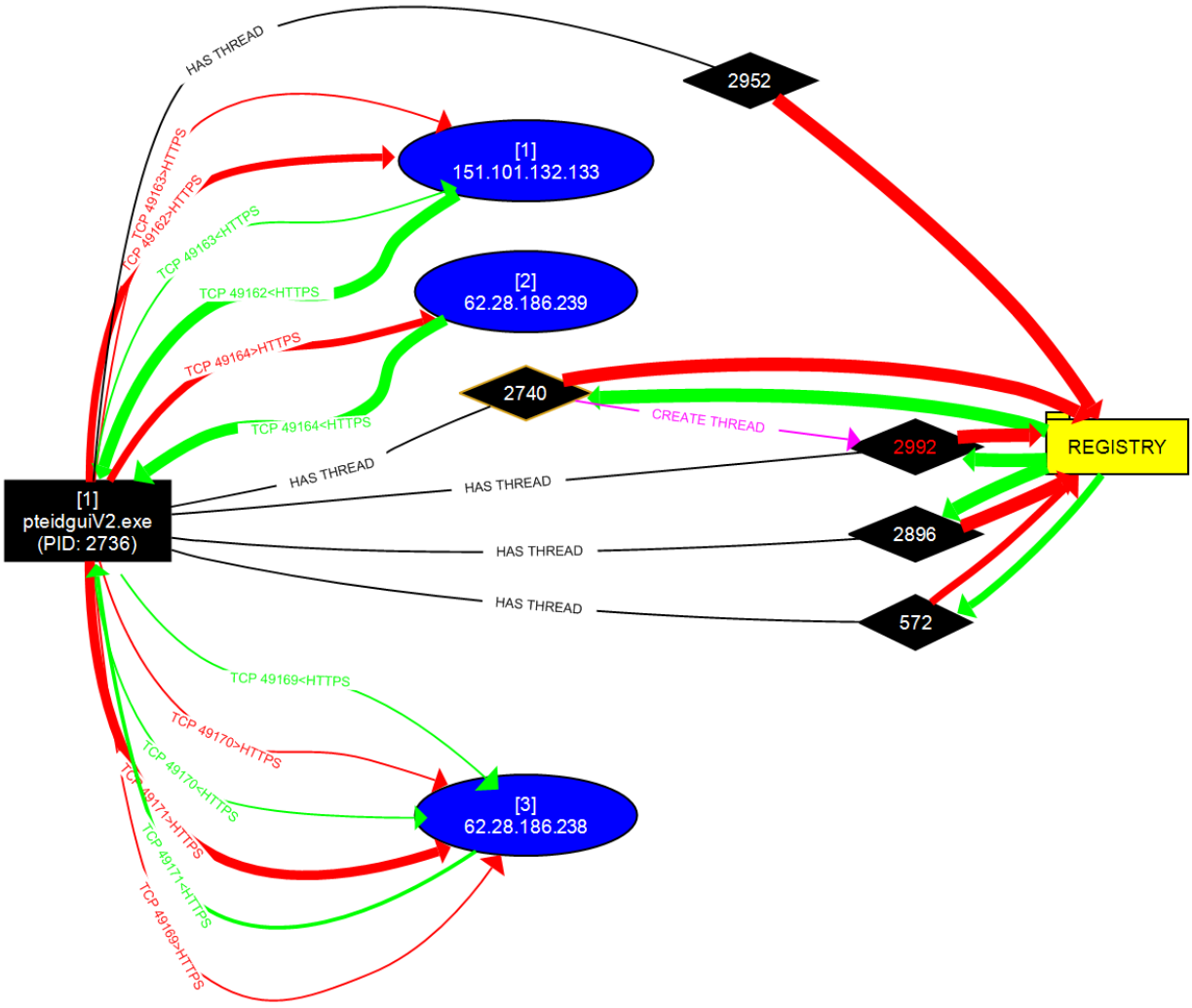


Figura 5-69 - Procdot registo alterações durante a assinatura com CMD

Se não existir acesso à Internet o processo para assinar um documento não é concluído apresentando mensagem de erro.

5.4. Resultados

Neste capítulo foram efetuados dois cenários de testes. O primeiro cenário foi elaborado com o objetivo de efetuar uma análise estática e dinâmica à aplicação Autenticação.gov para dispositivos móveis Android.

Na análise estática efetuada foram encontradas várias fragilidades na aplicação Autenticação.gov, identificadas através do software MOBSF. Na análise dinâmica não foram

encontradas fragilidades na aplicação, no entanto, todo o código fonte encontra-se acessível no ficheiro APK da aplicação como foi demonstrado.

O segundo cenário foi elaborado para validar a existência de comunicações externas e verificar todas as alterações no computador no momento do uso do programa Autenticação.gov.

O primeiro teste realizado teve o propósito de assinar um documento com a aplicação Cartão de Cidadão. Durante este processo todas as alterações registadas no computador relativas ao programa Autenticação.gov foram capturadas e tratadas no programa ProcDot. Neste verificamos a existência de comunicações não espectáveis a servidores externos. Através do certificado exportado verificou-se o número de série deste assim como o da chave pública.

No segundo teste realizado, o documento foi assinado através da Chave Móvel Digital. Como era espectável através do programa ProcDOT foram verificadas comunicações a servidores externos, no entanto, estas comunicações são diferenciadas das listadas no primeiro teste. Através do programa ProcDOT não foi possível detetar a localização de qualquer tipo de chaves usadas para efetuar a assinatura do documento.

Na Figura 5-68 é apresentado o número de série do certificado assim como os dados referentes a chave pública usada.

Estes, são diferentes dos dados registados do primeiro teste, o que nos leva a concluir que existirá servidores autorizados com um par de chaves (privada e pública) de um determinado individuo para uso no momento da assinatura de documentos através da Chave Móvel Digital.

A Tabela 5-3 resume as vulnerabilidades encontradas durante os testes efetuados.

Tabela 5-3 – Resumo das vulnerabilidades da aplicação autenticação.gov

	Tipo análise	**Ferramentas**	**Descrição**
Fase 1	Análise estática	MobSF	Foram encontradas diversas vulnerabilidades ao nível das permissões.
	Análise dinâmica	Superfície de ataque	Não foram encontradas vulnerabilidades nos diversos testes efetuados.

	Análise dinâmica	Ofuscação do código	Não sendo obrigatório a ofuscação de código, este encontra-se presente no ficheiro APK sem qualquer tipo de proteção.
Fase 2		Assinatura com o Cartão de Cidadão	Foram verificadas comunicações a servidores externos. Número série certificado e chave públicas usadas diferentes
		Assinatura com a Chave Móvel Digital	Foram verificadas comunicações a servidores externos. Número série certificado e chave públicas usadas diferentes

6. Conclusão

A assinatura digital surgiu da necessidade de transpor a assinatura manuscrita para os documentos eletrónicos. A evolução das tecnologias de informação e comunicação criou a necessidade por parte da Administração Pública, de uma restruturação dos serviços prestados aos cidadãos disponibilizando meios técnicos para que possa ser possível assinar um documento digitalmente.

Este trabalho descreve o funcionamento da Chave Móvel Digital. O estudo realizado de forma detalhada direcionou este projeto para a análise da aplicação Autenticação.gov desenvolvida pela Administração Pública para dispositivos móveis com o SO Android.

Como ponto de partida é abordado no capítulo 2 uma visão geral sobre o conceito de análise aplicacional com destaque para métodos e ferramentas usadas para efetuar essa análise. Foi ainda realizada uma síntese sobre as soluções propostas por outros investigadores visando a análise aplicacional.

No capítulo 3 é feita referência aos tipos de autenticação, analisando os métodos usados através da Chave Móvel Digital e da aplicação Autenticação.gov.

No capítulo 4 abordamos as tecnologias utilizadas no presente trabalho. Descreve-se de uma maneira geral a arquitetura do sistema operativo móvel Android que suporta a aplicação Autenticação.gov. No mesmo são apresentadas as ferramentas enumeradas nos capítulos anteriores para efetuar uma análise estática e dinâmica de uma aplicação móvel para o sistema operativo Android.

A análise à aplicação Autenticação.gov é realizada no capítulo 5 usando as ferramentas descritas nos capítulos 2 e 4. Foi realizada uma análise estática e dinâmica a aplicação com o objetivo de encontrar possíveis vulnerabilidades. Para perceber o funcionamento da Chave Móvel Digital, foi efetuado um estudo acerca do seu comportamento. Em tal estudo realizou-se a assinatura de um documento que foi comparada com a assinatura de um outro documento efetuada através do Cartão de Cidadão. Foram efetuados diversos testes visando encontrar diferenças nas assinaturas realizadas através de formas de diferentes.

Por fim foram realizados e discutidos os testes efetuados.

Os objetivos definidos para este trabalho eram:

- Estudar o funcionamento da Chave Móvel Digital;
- Realizar uma análise estática e dinâmica à aplicação móvel para o sistema Android Autenticação.gov;
- Comparar a assinatura efetuada através do Cartão de Cidadão com a assinatura praticada com a Chave Móvel Digital.

Os objetivos propostos na realização deste projeto foram todos alcançados. O estudo da Chave Móvel Digital, assim como a análise à aplicação Autenticação.gov permitiu perceber o seu comportamento e efetuar o levamento das vulnerabilidades identificadas.

6.1. Principais Contribuições

Tomando em consideração os objetivos iniciais e os resultados obtidos, as principais contribuições deste projeto foram:

- Criação de um estudo sobre as vulnerabilidades da aplicação Autenticação.gov para dispositivos móveis com o sistema operativo Android recorrendo a análise estática e dinâmica;
- Elaboração de um estudo comparativo entre assinatura eletrónica realizada através da Chave Móvel Digital e do Cartão de Cidadão.

6.2. Tópicos para Trabalho Futuro

Este trabalho de dissertação proporciona várias perspetivas que podem ser exploradas num futuro próximo, não só a nível da continuação e aperfeiçoamento da análise no momento de lançamento de novas versões da aplicação, como também alagar e estender a pesquisa ao sistema operativo iOS.

Para tal será necessário efetuar um estudo sobre o levamento das ferramentas de análise disponíveis para os dois sistemas operativos e realizar um estudo comparativo de métodos e técnicas de análise de vulnerabilidades paras os dois sistemas operativos.

Bibliografia

Abraham, A., & Superpoussin22. (2019). *Mobile-Security-Framework-MobSF @ github.com*. https://github.com/MobSF/Mobile-Security-Framework-MobSF

Almeida, D. T. N. P. (2009). *Assinatura Electrónica Qualificada*.

Ama, A. P. A. M. A. (2007). *Cartão de Cidadão O novo documento de identificação dos cidadãos portugueses*. 1–19.

Ama, A. P. A. M. A. (2019). *www.autenticacao.gov.pt*. https://www.autenticacao.gov.pt/cmd-pedido-chave

Anacom. (1999). *Decreto-Lei n.º 290-D/99, de 2 de agosto*. https://www.anacom.pt/render.jsp?contentId=957061

Android. (2019a). *abis @ developer.android.com*. https://developer.android.com/ndk/guides/abis.html

Android. (2019b). *Android Runtime (ART) and Dalvik.* https://source.android.com/devices/tech/dalvik/

Android. (2019c). *Fundamentos de aplicativos*. https://developer.android.com/guide/components/fundamentals

Apache. (2019). *apktool*. https://ibotpeaches.github.io/Apktool/

AV-TEST. (2019). *AV-TEST Security Report 2018/19*. https://www.av-test.org/fileadmin/pdf/security_report/AV-TEST_Security_Report_2018-2019.pdf

Avira. (2020). *What a bargain! Get a real app – and the Anubis Trojan – with cloned and repackaged apps on 3rd party markets*. https://www.avira.com/en/blog/what-a-bargain-get-a-real-app-and-the-anubis-trojan-with-cloned-and-repackaged-apps-on-3rd-party-markets

Bilić, D. G. (2019). *Segurança em dispositivos móveis: balanço do primeiro semestre de 2019*. https://www.welivesecurity.com/br/2019/09/04/seguranca-em-dispositivos-moveis-balanco-do-primeiro-semestre-de-2019/

Bilić, D. G. (2020). *Detecções de malware em dispositivos móveis diminuem no Android e crescem no iOS*. https://www.welivesecurity.com/br/2020/01/09/deteccoes-de-malware-em-dispositivos-moveis-diminuem-no-android-e-crescem-no-ios/

Chauhan, S. (2020). *Understanding Xamarin Android - Build Native Android App*. https://www.dotnettricks.com/learn/xamarin/understanding-xamarin-android-build-native-android-app

Chebyshev, V. (2019). *Mobile malware evolution 2018*. https://securelist.com/mobile-malware-evolution-2018/89689/

Chen, H., Yang, C. C., Chau, M., & Li, S. H. (2010). Open Research Problems in Network Security. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 5477). Springer.

Chen, S., Su, T., Fan, L., Meng, G., Xue, M., Liu, Y., & Xu, L. (2018). Are mobile banking apps secure? what can be improved? *ESEC/FSE 2018 - Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, *2018*, 797–802. https://doi.org/10.1145/3236024.3275523

ComsumerLab, E. (2020). *6.1B Smartphone Users Globally By 2020, Overtaking Basic Fixed Phone Subscriptions*. https://techcrunch.com/2015/06/02/6-1b-smartphone-users-globally-by-2020-overtaking-basic-fixed-phone-subscriptions/?guccounter=1&guce_referrer=aHR0cHM6Ly93d3cua2FzcGVyc2t5LmNvbS5ici9yZXNvdXJjZS1jZW50ZXIvdGhyZWF0cy9tb2JpbGU&guce_referrer_sig=AQAAAEWHUwIS1

Costa Marques, P. P. L. (2013). *Informática Forense Recolha e preservação da prova digital*.

Dang, B., Gazet, A., Bachaalany, E., & Josse, S. (2014). *Practical Reverse Engineering: x86, x64, ARM, Windows Kernel, Reversing Tools, and Obfuscation*. Wiley. https://goo.gl/H9a9HH

De Andrade, C. A. B., De Mello, C. G., & Duarte, J. C. (2013). Malware automatic analysis. In *Proceedings - 1st BRICS Countries Congress on Computational Intelligence, BRICS-CCI 2013*. https://doi.org/10.1109/BRICS-CCI-CBIC.2013.119

Donohue, B. (2014). *O que é a autenticação de dois fatores e como usá-la?* https://www.kaspersky.com.br/blog/o-que-e-a-autenticacao-de-dois-fatores-e-como-usa-la/3226/

DRE. (2003). *Decreto-Lei n.º 62/2003 de 3 de Abril do Ministério da Justiça*. www.dre.pt

DRE. (2019). *Decreto-Lei n.º 290-D/99*. 1–18.

Dupuy, E. (2019). *java-decompiler/jd-gui*. https://github.com/java-decompiler/jd-gui/releases

ETSI EN 319 142-1. (2016). *Electronic Signatures and Infrastructures (ESI); PAdES digital signatures; 1*, 1–23.

Fauskrud, J. (2019). *Hybrid analysis for Android malware family classification in a time-aware setting* (Issue June). Norwegian University of Science and Technology.

Franek, B. P. (2017). *Secure software development process and tools for Android applications*. Masaryk University Faculty of Informatics.

Gartner. (2019). *2019-02-21-gartner-says-global-smartphone-sales-stalled-in-the-fourth-quart @ www.gartner.com*. https://www.gartner.com/en/newsroom/press-releases/2019-02-21-gartner-says-global-smartphone-sales-stalled-in-the-fourth-quart

GlobalStats. (2020). *Mobile Operating System Market Share Worldwide - October 2020*. https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-201910-202010

Google. (2020). *Google Developers*. https://developer.android.com/reference/android/Manifest.permission?hl=pt#GET_ACCOUNTS

Google Developers. (2019a). *Android Debug Bridge*. https://developer.android.com/studio/command-line/adb

Google Developers. (2019b). *manifest-intro developer.android.com*. https://developer.android.com/guide/topics/manifest/manifest-intro?hl=pt-BR

Google Developers. (2020). *Content provider basics*. https://developer.android.com/guide/topics/providers/content-provider-basics

Granjal, J. (2017). *Segurança Prática em Sistemas e Redes com Linux*. FCA.

Greenberg, A. (2019). *The SIM Swap Fix That the US Isn't Using*. https://www.wired.com/story/sim-swap-fix-carriers-banks/

Gregg, M. (2015). The Network Security Test Lab. In *The Network Security Test Lab*. https://doi.org/10.1002/9781119183433

Hutchison, D., & Mitchell, J. C. (2006). *Computer Security – ESORICS 2006*.

IDC. (2020). *Quota de mercado do smartphone*. https://www.idc.com/promo/smartphone-market-share/os

INCM. (2020). *Controlo do n.º de Versão Cartão de Cidadão*. 1–9.

Infopedia. (2018). *malware*.

InfoSecurity. (2019). *MWR LABS*. https://labs.mwrinfosecurity.com/tools/drozer/

Inside, T. (2020). *Quase 30% das empresas já sofreram ataques a dispositivos móveis*. https://tiinside.com.br/25/05/2020/quase-30-das-empresas-ja-sofreram-ataques-a-dispositivos-moveis/

Junior, V. A. (2019). *Gestão De Identidade E Acesso Multifator Para Smart Grid*.

kaspersky. (2019). *encyclopedia.kaspersky.com*. https://encyclopedia.kaspersky.com/knowledge/risktool/

kaspersky. (2020). *Ameaças à segurança de dispositivos móveis Android*. https://www.kaspersky.com.br/resource-center/threats/mobile

Khandelwal, S. (2018). *Pre-Installed Malware Found On 5 Million Popular Android Phones*. https://thehackernews.com/2018/03/android-botnet-malware.html

Linuxforyou. (2008). *A Developer's First Look At Android*.

López, J. L. P. (2018). *Análisis de Seguridad de las Aplicationes Móviles basadas en Android, utilizadas en la banca Electrónica del Equador.* (Vol. 66). Escuela Politécnica Nacional Facultad de Ingeniería Elétrica y Electrónica.

MD5ONLINE. (2020). *www.md5online.org*. https://www.md5online.org/blog/why-md5-is-not-safe/

Mendonça, M. F., Souza, L. S., & Lima, I. (2017). *Automatização de Teste de Segurança Móvel com MobiSec*. *September*, 1–4. https://doi.org/10.5753/sbsi.2019.7429

Mengato, R., Santin, A., Abreu, V., & Borchardt, M. (2019). Método de Autenticação Multi-canal Baseado em Proximidade. *XIX Simpósio Brasileiro de Segurança Da Informação e Sistemas Computacionais*.

MICROCOSM. (2018). *HOTP vs TOTP: What's the Difference?* https://www.microcosm.com/blog/hotp-totp-what-is-the-difference

Microsoft.com. (2020). *Architecture Xamarin.Android*. https://docs.microsoft.com/en-us/xamarin/android/internals/architecture

Microsoft. (2019). *https://azure.microsoft.com/en-us/overview/what-is-middleware/*. https://azure.microsoft.com/en-us/overview/what-is-middleware/

Molnar, T., & Ko, A. (2016). Electronic Government and the Information Systems Perspective. In *9th International Conference, EGOVIS 2020, LNCS*.

Montgomery, A., & Mingis, K. (2020). *The evolution of Apple's iPhone*. Computerworld. http://www.computerworld.com/article/2604020/smartphones/the-evolution-of-apples-iphone.html#slide11

Naway, A., & Li, Y. (2019). *Android Malware Detection Using Autoencoder*. 1–9. http://arxiv.org/abs/1901.07315

Oktavianto, D., & Muhardianto, I. (2013). *Cuckoo Malware Analysis*. Packt Publishing.

Onelogin. (2020). *Qual é a diferença entre OTP, TOTP e HOTP?* https://www.onelogin.com/learn/otp-totp-hotp

Panda. (2018). *Malware clássico*. https://www.pandasecurity.com/brazil/homeusers/security-info/classic-malware/

Parlamento, O. (2000). *DIRECTIVA 1999/93/CE DO PARLAMENTO EUROPEU E DO CONSELHO de 13 de Dezembro de 1999 relativa a um quadro legal comunitário para as assinaturas electrónicas*. *95*, 12–20.

Paulo, U. D. S., & Pinheiro, M. R. (2004). *Análise de Artefatos Maliciosos em Ambiente Acadêmico*.

Piqueras Roger, J. (2020). Security Analysis of SMS. *International Journal for Digital Society*, *10*(4), 1556–1561. https://doi.org/10.20533/ijds.2040.2570.2019.0193

Plata, I. T., & Calpito, J. L. (2020). Application of time-based one time password (TOTP) algorithm for human resource e-leave tracking web app. *International Journal of Scientific and Technology Research*, *9*(3), 4070–4077.

Pohlmann, N., Reimer, H., Schneider, W., Electronic, S., & Processes, B. (2010). *ISSE 2009 Securing Electronic Business Processes*.

Polychronakis, M., & (Eds.), M. M. (2017). *Detection of Intrusions and Malware, and Vulnerability Assessment*.

Pxb1988. (2019). *dex2jar github.com*. https://github.com/pxb1988/dex2jar

Rahav, A. (2020). *Software Tokens Vs Hardware Tokens*. https://doubleoctopus.com/blog/tokens-hard-soft-and-whats-in-between/

Ranganath, V. (2019). *Are Free Android App Security Analysis Tools Effective in Detecting Known Vulnerabilities ?*

República Portuguesa. (2020). *DRE Diário da República Electrónico*. https://dre.pt/home/-/dre/69879391/details/maximized?p_auth=Ev8uUQ6y

Russo, T. (2019). *SIMulated Trust : How Malicious Actors Take Advantage of Cellular Carriers to Perform SIM Swapping Attacks*.

Santoku. (2019). *santoku-linux.com*. https://santoku-linux.com/

SCEE. (2020). *Sistema de Certificação Electrónica do Estado*. https://www.scee.gov.pt/

Sean, C. (2019). *The Most Expensive Lesson Of My Life: Details of SIM port hack*. https://medium.com/coinmonks/the-most-expensive-lesson-of-my-life-details-of-sim-port-hack-35de11517124

Security, H. (2018). *O que é um Payload e por que são usados por hackers?* https://hackersec.com/o-que-e-um-payload-e-por-que-sao-usados-por-hackers/

Shahriar, H. (2019). *An Exploratory Analysis of Mobile Security Tools*.

Sikorski, M., & Honig, A. (2012). *Practical Malware Analysis: A Hands-On Guide to*

Dissecting Malicious Software. No Starch Press.

Stackoverflow. (2017). *security-exception-write-use-app-feature-survey-in-samsung-phones*. https://stackoverflow.com/questions/38281449/security-exception-write-use-app-feature-survey-in-samsung-phones

Statista. (2020). *Number of apps available in leading app stores as of 2nd quarter 2020*. https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

Suciu, G., Istrate, C.-L., Raducanu, R. I., Ditu, M.-C., Fratu, O., & Vulpe, A. (2019). *Mobile devices forensic pla tform for malware detection*.

Tam, K., Feizollah, A., Anuar, N. B., Salleh, R., & Cavallaro, L. (2017). The evolution of android malware and android analysis techniques. *ACM Computing Surveys*, *49*(4), 1–41. https://doi.org/10.1145/3017427

Turner, D. M. (2017). *PAdES and Long Term archival (LTA)*. https://www.cryptomathic.com/news-events/blog/pades-and-long-term-archival-lta

Wei, F., Li, Y., Roy, S., Ou, X., & Zhou, W. (2017). Deep ground truth analysis of current android malware. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10327 LNCS*, 252–276. https://doi.org/10.1007/978-3-319-60876-1_12

Zhao, J. (2017). *Android LogCat And Logging Best Practice*. https://www.dev2qa.com/android-logcat-and-logging-best-practice/

Zhao, J. (2018). *Android Broadcast Overview*. https://www.dev2qa.com/android-broadcast-overview/

Zoe, N. (2017). *How Office Managers Can Make the Most of Smartphone Habits in the Workplace*. https://www.officespacesoftware.com/blog/how-office-managers-can-make-the-most-of-smartphone-habits-in-the-workplace

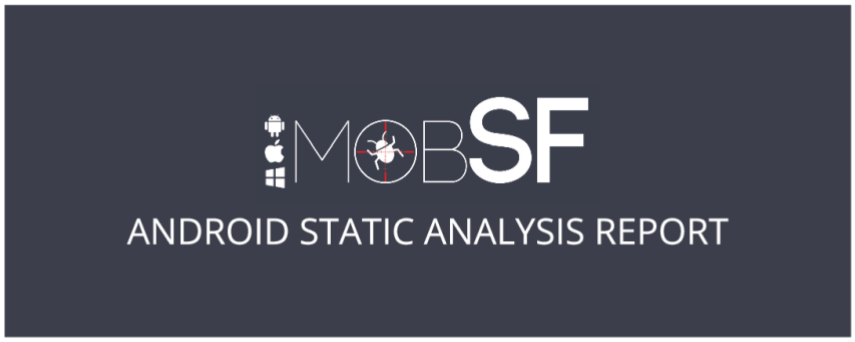
Zúquete, A. (2018). *Segurança em redes informáticas*. FCA.

Anexo A – MobSF

Apresenta-se a listagem efetuada pela ferramenta MobSF que lista as vulnerabilidades encontradas após análise dinâmica efetuada a aplicação autenticação.gov.

Link para análise da versão 2.0.0 : https://tinyurl.com/yxsvaczl

Link para análise da versão 3.2.1: https://tinyurl.com/y434bfro



Autenticação Gov (2.0.0)

File Name:	Autenticação_Gov_v2.0.0.apk
Package Name:	pt.ama.autenticacaogov
Average CVSS Score:	4.9
App Security Score:	75/100 (LOW RISK)

FILE INFORMATION

File Name: Autenticação_Gov_v2.0.0.apk
Size: 30.29MB
MD5: 0b6475db153d8a042a0f42fff906f487
SHA1: 38892c6e1088925a2dadc2d19b1543ea1a9e3cfc
SHA256: 6bd5e2eae4992f3d138c3911fe85ac185c554586cfba23512c415624c1347b87

APP INFORMATION

App Name: Autenticação Gov
Package Name: pt.ama.autenticacaogov
Main Activity: md5ba9292671f075a189d6b15232e1c2e4f.MainActivity
Target SDK: 17
Min SDK: 17
Max SDK:
Android Version Name: 2.0.0
Android Version Code: 23

APP COMPONENTS

Activities: 3
Services: 1
Receivers: 1
Providers: 1
Exported Activities: 0
Exported Services: 0
Exported Receivers: 0
Exported Providers: 0

CERTIFICATE INFORMATION

APK is signed
v1 signature: True
v2 signature: True
v3 signature: True
Found 1 unique certificates
Subject: C=US, ST=California, L=Mountain View, O=Google Inc., OU=Android, CN=Android
Signature Algorithm: rsassa_pkcs1v15
Valid From: 2017-10-01 20:37:46+00:00
Valid To: 2047-10-01 20:37:46+00:00
Issuer: C=US, ST=California, L=Mountain View, O=Google Inc., OU=Android, CN=Android
Serial Number: 0x63c55266add0058e82f3291b4d87a58dbf303585
Hash Algorithm: sha256
md5: 45519715d1bde3d9b95253b3bd56b3de
sha1: 4117982bf71c4a3e8df5346e38967d3a6b94f821
sha256: 07f2c5452493a70bf557ca68fea8aad62e6b8afcfc5e9d1184bed171e4676a97
sha512:
b9c0a94a008f151fb0ded6389f72965b2e2255050785ef4fddf4a6933f39bad9d388424887993923abc2a0eaaeca7bcf7419e5e3e20b9919a3bbd5814966922f

PublicKey Algorithm: rsa
Bit Size: 4096

Fingerprint: 484b9c9809802d02fc05eabd6b12f216274643ff89d3298d4ee1f1024cee6eda

Certificate Status: Good
Description: Certificate looks good.

APPLICATION PERMISSIONS

PERMISSION	STATUS	INFO	DESCRIPTION
android.permission.ACCESS_NETWORK_STATE	normal	view network status	Allows an application to view the status of all networks.
android.permission.USE_FINGERPRINT	normal	allow use of fingerprint	This constant was deprecated in API level 28. Applications should request USE_BIOMETRIC instead
android.permission.BIND_NOTIFICATION_LISTENER_SERVICE	signature		Must be required by an NotificationListenerService, to ensure that only the system can bind to it.
android.permission.INTERNET	dangerous	full Internet access	Allows an application to create network sockets.
android.permission.READ_EXTERNAL_STORAGE	dangerous	read SD card contents	Allows an application to read from SD Card.
android.permission.READ_LOGS	dangerous	read sensitive log data	Allows an application to read from the system's various log files. This allows it to discover general information about what you are doing with the phone, potentially including personal or private information.
android.permission.READ_PHONE_STATE	dangerous	read phone state and identity	Allows the application to access the phone features of the device. An application with this permission can determine the phone number and serial number of this phone, whether a call is active, the number that call is connected to and so on.
android.permission.VIBRATE	normal	control vibrator	Allows the application to control the vibrator.
android.permission.WAKE_LOCK	dangerous	prevent phone from sleeping	Allows an application to prevent the phone from going to sleep.
com.example.proj.permission.C2D_MESSAGE	signature	Allows cloud to device messaging	Allows the application to receive push notifications.
android.permission.WRITE_EXTERNAL_STORAGE	dangerous	read/modify/delete SD card contents	Allows an application to write to the SD card.
pt.ama.autenticacaogov.permission.C2D_MESSAGE	signature	Allows cloud to device messaging	Allows the application to receive push notifications.
com.google.android.c2dm.permission.RECEIVE	signature	C2DM permissions	Permission for cloud to device messaging.

android.permission.GET_ACCOUNTS	normal	discover known accounts	Allows an application to access the list of accounts known by the phone.

APKID ANALYSIS

FILE	DETAILS
classes.dex	FINDINGS: Compiler — DETAILS: dx

MANIFEST ANALYSIS

ISSUE	SEVERITY	DESCRIPTION

CODE ANALYSIS

ISSUE	SEVERITY	CVSS	CWE	OWASP	FILES
					md5a104545e4d19c4ffe9ec3d5074a3b979/WebAuthenticatorActivity.java md5a104545e4d19c4ffe9ec3d5074a3b979/AndroidAccountStore_SecretAccount.java md5a104545e4d19c4ffe9ec3d5074a3b979/WebAuthenticatorActivity_Client.java md5a104545e4d19c4ffe9ec3d5074a3b979/FormAuthenticatorActivity_State.java md5a104545e4d19c4ffe9ec3d5074a3b979/FormAuthenticatorActivity.java md5a104545e4d19c4ffe9ec3d5074a3b979/WebAuthenticatorActivity_State.java android/runtime/XmlReaderResourceParser.java android/runtime/JavaProxyThrowable.java android/runtime/XmlReaderPullParser.java android/runtime/UncaughtExceptionHandler.java android/app/ActivityTracker.java mono/java/lang/RunnableImplementor.java mono/java/lang/Runnable.java mono/java/util/EventListenerImplementor.java mono/android/TypeManager.java mono/android/sax/StartElementListenerImplementor.java mono/android/sax/EndElementListenerImplementor.java mono/android/sax/EndTextElementListenerImplementor.java mono/android/preference/Preference_OnPreferenceClickListenerImplementor.java

					mono/android/preference/PreferenceManager_OnActivityDestroyListenerImplementor.java mono/android/preference/Preference_OnPreferenceChangeListenerImplementor.java mono/android/preference/PreferenceManager_OnActivityResultListenerImplementor.java mono/android/preference/PreferenceManager_OnActivityStopListenerImplementor.java mono/android/location/GpsStatus_NmeaListenerImplementor.java mono/android/location/LocationListenerImplementor.java mono/android/location/GpsStatus_ListenerImplementor.java mono/android/location/OnNmeaMessageListenerImplementor.java mono/android/runtime/OutputStreamAdapter.java mono/android/runtime/InputStreamAdapter.java mono/android/runtime/JavaObject.java mono/android/app/AlarmManager_OnAlarmListenerImplementor.java mono/android/app/UiAutomation_OnAccessibilityEventListenerImplementor.java mono/android/app/TabEventDispatcher.java mono/android/app/DatePickerDialog_OnDateSetListenerImplementor.java mono/android/app/FragmentManager_OnBackStackChangedListenerImplementor.java mono/android/app/SharedElementCallback_OnSharedElementsReadyListenerImplementor.java mono/android/app/TimePickerDialog_OnTimeSetListenerImplementor.java mono/android/app/SearchManager_OnDismissListenerImplementor.java mono/android/app/FragmentBreadCrumbs_OnBreadCrumbClickListenerImplementor.java mono/android/app/SearchManager_OnCancelListenerImplementor.java mono/android/app/Application_OnProvideAssistDataListenerImplementor.java mono/android/app/ActionBar_TabListenerImplementor.java mono/android/app/AppOpsManager_OnOpChangedListenerImplementor.java mono/android/app/ActionBar_OnNavigationListenerImplementor.java mono/android/app/ActionBar_OnMenuVisibilityListenerImplementor.java mono/android/database/sqlite/SQLiteTransactionListenerImplementor.java mono/android/renderscript/Allocation_OnBufferAvailableListenerImplementor.java mono/android/animation/LayoutTransition_TransitionListenerImplementor.java mono/android/animation/Animator_AnimatorPauseListenerImplementor.java mono/android/animation/Animator_AnimatorListenerImplementor.java mono/android/animation/AnimatorEventDispatcher.java mono/android/animation/TimeAnimator_TimeListenerImplementor.java mono/android/animation/ValueAnimator_AnimatorUpdateListenerImplementor.java mono/android/inputmethodservice/KeyboardView_On

					KeyboardActionListenerImplementor.java mono/android/speech/RecognitionListenerImplementor.java mono/android/speech/tts/TextToSpeech_OnInitListenerImplementor.java mono/android/speech/tts/TextToSpeech_OnUtteranceCompletedListenerImplementor.java mono/android/widget/CompoundButton_OnCheckedChangeListenerImplementor.java mono/android/widget/AdapterView_OnItemClickListenerImplementor.java mono/android/widget/SearchView_OnQueryTextListenerImplementor.java mono/android/widget/ExpandableListView_OnGroupCollapseListenerImplementor.java mono/android/widget/AutoCompleteTextView_OnDismissListenerImplementor.java mono/android/widget/DatePicker_OnDateChangedListenerImplementor.java mono/android/widget/AbsListView_RecyclerListenerImplementor.java mono/android/widget/NumberPicker_OnValueChangeListenerImplementor.java mono/android/widget/SlidingDrawer_OnDrawerScrollListenerImplementor.java mono/android/widget/TextView_OnEditorActionListenerImplementor.java mono/android/widget/PopupWindow_OnDismissListenerImplementor.java mono/android/widget/Filter_FilterListenerImplementor.java mono/android/widget/ActionMenuView_OnMenuItemClickListenerImplementor.java mono/android/widget/ExpandableListView_OnChildClickListenerImplementor.java mono/android/widget/PopupMenu_OnMenuItemClickListenerImplementor.java mono/android/widget/TabHost_OnTabChangeListenerImplementor.java mono/android/widget/SlidingDrawer_OnDrawerCloseListenerImplementor.java mono/android/widget/Toolbar_OnMenuItemClickListenerImplementor.java mono/android/widget/CalendarView_OnDateChangeListenerImplementor.java mono/android/widget/ExpandableListView_OnGroupClickListenerImplementor.java mono/android/widget/ExpandableListView_OnGroupExpandListenerImplementor.java mono/android/widget/RadioGroup_OnCheckedChangeListenerImplementor.java mono/android/widget/SearchView_OnSuggestionListenerImplementor.java mono/android/widget/SlidingDrawer_OnDrawerOpenListenerImplementor.java mono/android/widget/SeekBar_OnSeekBarChangeListenerImplementor.java mono/android/widget/ShareActionProvider_OnShareTargetSelectedListenerImplementor.java mono/android/widget/ZoomButtonsController_OnZoomListenerImplementor.java mono/android/widget/SearchView_OnCloseListenerImplementor.java mono/android/widget/TimePicker_OnTimeChangedLis

					tenerImplementor.java mono/android/widget/Chronometer_OnChronometerTickListenerImplementor.java mono/android/widget/PopupMenu_OnDismissListenerImplementor.java mono/android/widget/AdapterView_OnItemSelectedListenerImplementor.java mono/android/widget/AbsListView_OnScrollListenerImplementor.java mono/android/widget/NumberPicker_OnScrollListenerImplementor.java mono/android/widget/AdapterView_OnItemLongClickListenerImplementor.java mono/android/widget/RatingBar_OnRatingBarChangeListenerImplementor.java mono/android/hardware/SensorEventListenerImplementor.java mono/android/hardware/Camera_FaceDetectionListenerImplementor.java mono/android/hardware/Camera_OnZoomChangeListenerImplementor.java mono/android/hardware/SensorListenerImplementor.java mono/android/hardware/input/InputManager_InputDeviceListenerImplementor.java mono/android/hardware/display/DisplayManager_DisplayListenerImplementor.java mono/android/accounts/OnAccountsUpdateListenerImplementor.java mono/android/net/ConnectivityManager_OnNetworkActiveListenerImplementor.java mono/android/net/sip/SipRegistrationListenerImplementor.java mono/android/net/nsd/NsdManager_DiscoveryListenerImplementor.java mono/android/net/nsd/NsdManager_ResolveListenerImplementor.java mono/android/net/nsd/NsdManager_RegistrationListenerImplementor.java mono/android/net/wifi/p2p/WifiP2pManager_DnsSdTxtRecordListenerImplementor.java mono/android/net/wifi/p2p/WifiP2pManager_PeerListListenerImplementor.java mono/android/net/wifi/p2p/WifiP2pManager_GroupInfoListenerImplementor.java mono/android/net/wifi/p2p/WifiP2pManager_ActionListenerImplementor.java mono/android/net/wifi/p2p/WifiP2pManager_ConnectionInfoListenerImplementor.java mono/android/net/wifi/p2p/WifiP2pManager_UpnpServiceResponseListenerImplementor.java mono/android/net/wifi/p2p/WifiP2pManager_ServiceResponseListenerImplementor.java mono/android/net/wifi/p2p/WifiP2pManager_ChannelListenerImplementor.java mono/android/net/wifi/p2p/WifiP2pManager_DnsSdServiceResponseListenerImplementor.java mono/android/os/MessageQueue_OnFileDescriptorEventListenerImplementor.java mono/android/os/ParcelFileDescriptor_OnCloseListenerImplementor.java mono/android/os/ActionHandlerCallback.java mono/android/os/RecoverySystem_ProgressListenerImplementor.java

					mono/android/os/CancellationSignal_OnCancelListenerImplementor.java mono/android/gesture/GestureOverlayView_OnGesturePerformedListenerImplementor.java mono/android/gesture/GestureOverlayView_OnGestureListenerImplementor.java mono/android/gesture/GestureOverlayView_OnGesturingListenerImplementor.java mono/android/text/TextWatcherImplementor.java mono/android/graphics/SurfaceTexture_OnFrameAvailableListenerImplementor.java mono/android/graphics/drawable/Icon_OnDrawableLoadedListenerImplementor.java mono/android/media/AudioTrack_OnRoutingChangedListenerImplementor.java mono/android/media/MediaPlayer_OnVideoSizeChangedListenerImplementor.java mono/android/media/MediaScannerConnection_OnScanCompletedListenerImplementor.java mono/android/media/SoundPool_OnLoadCompleteListenerImplementor.java mono/android/media/AudioRecord_OnRecordPositionUpdateListenerImplementor.java mono/android/media/MediaSync_OnErrorListenerImplementor.java mono/android/media/MediaPlayer_OnPreparedListenerImplementor.java mono/android/media/ImageReader_OnImageAvailableListenerImplementor.java mono/android/media/ImageWriter_OnImageReleasedListenerImplementor.java mono/android/media/AudioTrack_OnPlaybackPositionUpdateListenerImplementor.java mono/android/media/MediaPlayer_OnSeekCompleteListenerImplementor.java mono/android/media/MediaPlayer_OnTimedMetaDataAvailableListenerImplementor.java mono/android/media/AudioRecord_OnRoutingChangedListenerImplementor.java mono/android/media/AudioManager_OnAudioFocusChangeListenerImplementor.java mono/android/media/JetPlayer_OnJetEventListenerImplementor.java mono/android/media/MediaDrm_OnKeyStatusChangeListenerImplementor.java mono/android/media/MediaDrm_OnExpirationUpdateListenerImplementor.java mono/android/media/MediaPlayer_OnErrorListenerImplementor.java mono/android/media/MediaPlayer_OnBufferingUpdateListenerImplementor.java mono/android/media/MediaPlayer_OnCompletionListenerImplementor.java mono/android/media/RemoteControlClient_OnPlaybackPositionUpdateListenerImplementor.java mono/android/media/MediaPlayer_OnTimedTextListenerImplementor.java mono/android/media/MediaDrm_OnEventListenerImplementor.java mono/android/media/MediaPlayer_OnInfoListenerImplementor.java mono/android/media/RemoteControlClient_OnGetPlaybackPositionListenerImplementor.java mono/android/media/MediaCodec_OnFrameRenderedListenerImplementor.java

IP Address disclosure	warning	4.3 medium	CWE-200		mono/android/media/MediaRecorder_OnInfoListenerImplementor.java mono/android/media/RemoteController_OnClientUpdateListenerImplementor.java mono/android/media/MediaRecorder_OnErrorListenerImplementor.java mono/android/media/RemoteControlClient_OnMetadataUpdateListenerImplementor.java mono/android/media/session/MediaSessionManager_OnActiveSessionsChangedListenerImplementor.java mono/android/media/midi/MidiManager_OnDeviceOpenedListenerImplementor.java mono/android/media/tv/TvView_OnUnhandledInputEventListenerImplementor.java mono/android/media/effect/EffectUpdateListenerImplementor.java mono/android/media/audiofx/EnvironmentalReverb_OnParameterChangeListenerImplementor.java mono/android/media/audiofx/Virtualizer_OnParameterChangeListenerImplementor.java mono/android/media/audiofx/AudioEffect_OnEnableStatusChangeListenerImplementor.java mono/android/media/audiofx/BassBoost_OnParameterChangeListenerImplementor.java mono/android/media/audiofx/AudioEffect_OnControlStatusChangeListenerImplementor.java mono/android/media/audiofx/Visualizer_OnDataCaptureListenerImplementor.java mono/android/media/audiofx/Equalizer_OnParameterChangeListenerImplementor.java mono/android/media/audiofx/PresetReverb_OnParameterChangeListenerImplementor.java mono/android/drm/DrmManagerClient_OnInfoListenerImplementor.java mono/android/drm/DrmManagerClient_OnErrorListenerImplementor.java mono/android/drm/DrmManagerClient_OnEventListenerImplementor.java mono/android/bluetooth/BluetoothProfile_ServiceListenerImplementor.java mono/android/view/View_OnSystemUiVisibilityChangeListenerImplementor.java mono/android/view/ViewTreeObserver_OnWindowFocusChangeListenerImplementor.java mono/android/view/ViewTreeObserver_OnGlobalFocusChangeListenerImplementor.java mono/android/view/View_OnKeyListenerImplementor.java mono/android/view/ViewTreeObserver_OnScrollChangedListenerImplementor.java mono/android/view/View_OnApplyWindowInsetsListenerImplementor.java mono/android/view/MenuItem_OnMenuItemClickListenerImplementor.java mono/android/view/View_OnTouchListenerImplementor.java mono/android/view/Window_OnRestrictedCaptionAreaChangedListenerImplementor.java mono/android/view/View_OnAttachStateChangeListenerImplementor.java mono/android/view/View_OnDragListenerImplementor.java mono/android/view/ViewTreeObserver_OnTouchModeChangeListenerImplementor.java

					mono/android/view/GestureDetector_OnDoubleTapListenerImplementor.java mono/android/view/ViewGroup_OnHierarchyChangeListenerImplementor.java mono/android/view/View_OnLayoutChangeListenerImplementor.java mono/android/view/GestureDetector_OnGestureListenerImplementor.java mono/android/view/ScaleGestureDetector_OnScaleGestureListenerImplementor.java mono/android/view/MenuItem_OnActionExpandListenerImplementor.java mono/android/view/View_OnGenericMotionListenerImplementor.java mono/android/view/TextureView_SurfaceTextureListenerImplementor.java mono/android/view/ViewTreeObserver_OnWindowAttachListenerImplementor.java mono/android/view/View_OnContextClickListenerImplementor.java mono/android/view/GestureDetector_OnContextClickListenerImplementor.java mono/android/view/View_OnHoverListenerImplementor.java mono/android/view/View_OnClickListenerImplementor.java mono/android/view/ActionProvider_VisibilityListenerImplementor.java mono/android/view/View_OnFocusChangeListenerImplementor.java mono/android/view/ViewTreeObserver_OnPreDrawListenerImplementor.java mono/android/view/PixelCopy_OnPixelCopyFinishedListenerImplementor.java mono/android/view/ViewStub_OnInflateListenerImplementor.java mono/android/view/View_OnCreateContextMenuListenerImplementor.java mono/android/view/View_OnLongClickListenerImplementor.java mono/android/view/Window_OnFrameMetricsAvailableListenerImplementor.java mono/android/view/ViewTreeObserver_OnDrawListenerImplementor.java mono/android/view/ViewTreeObserver_OnGlobalLayoutListenerImplementor.java mono/android/view/View_OnScrollChangeListenerImplementor.java mono/android/view/accessibility/AccessibilityManager_AccessibilityStateChangeListenerImplementor.java mono/android/view/accessibility/AccessibilityManager_TouchExplorationStateChangeListenerImplementor.java mono/android/view/animation/Animation_AnimationListenerImplementor.java mono/android/view/textservice/SpellCheckerSession_SpellCheckerSessionListenerImplementor.java mono/android/accessibilityservice/AccessibilityService_SoftKeyboardController_OnShowModeChangedListenerImplementor.java mono/android/accessibilityservice/AccessibilityService_MagnificationController_OnMagnificationChangedListenerImplementor.java mono/android/transition/Transition_TransitionListene

					rImplementor.java mono/android/webkit/DownloadListenerImplementor.java mono/android/webkit/WebView_FindListenerImplementor.java mono/android/webkit/WebView_PictureListenerImplementor.java mono/android/webkit/WebIconDatabase_IconListenerImplementor.java mono/android/nfc/NfcAdapter_OnTagRemovedListenerImplementor.java mono/javax/xml/transform/ErrorListenerImplementor.java md5a7279444c0fe86aa930f169dd7d0faa5/GcmBroadcastReceiver.java md5a7279444c0fe86aa930f169dd7d0faa5/GcmService.java md5214eafb7e7b3b7fcc363a68a6358563f/GcmServiceBase.java md5214eafb7e7b3b7fcc363a68a6358563f/GcmBroadcastReceiverBase_1.java opentk/GameViewBase.java opentk/platform/android/AndroidGameView.java md5270abb39e60627f0f200893b490a1ade/FormsViewPager.java md5270abb39e60627f0f200893b490a1ade/SwitchRenderer.java md5270abb39e60627f0f200893b490a1ade/PickerRenderer_PickerListener.java md5270abb39e60627f0f200893b490a1ade/ViewRenderer_2.java md5270abb39e60627f0f200893b490a1ade/NavigationPageRenderer_DrawerMultiplexedListener.java md5270abb39e60627f0f200893b490a1ade/MasterDetailContainer.java md5270abb39e60627f0f200893b490a1ade/FrameRenderer.java md5270abb39e60627f0f200893b490a1ade/NavigationPageRenderer_ClickListener.java md5270abb39e60627f0f200893b490a1ade/ButtonRenderer_ButtonClickListener.java md5270abb39e60627f0f200893b490a1ade/ButtonRenderer.java md5270abb39e60627f0f200893b490a1ade/MasterDetailPageRenderer.java md5270abb39e60627f0f200893b490a1ade/Platform_ModalContainer.java md5270abb39e60627f0f200893b490a1ade/FragmentContainer.java md5270abb39e60627f0f200893b490a1ade/FormsFragmentPagerAdapter_1.java md5270abb39e60627f0f200893b490a1ade/NavigationPageRenderer.java md5270abb39e60627f0f200893b490a1ade/CarouselPageRenderer.java md5270abb39e60627f0f200893b490a1ade/PickerRenderer.java md5270abb39e60627f0f200893b490a1ade/TabbedPageRenderer.java opentk_1_0/GameViewBase.java opentk_1_0/platform/android/AndroidGameView.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/StepperRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/StepperRenderer_StepperListener.java

					md5b60ffeb829f638581ab2bb9b1a7f4f3f/SwitchRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/BoxRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/VisualElementTracker_AttachTracker.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/PickerRenderer_PickerListener.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/EntryCellView.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/ProgressBarRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/AndroidActivity.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/ActionSheetRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/EntryEditText.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/NativeViewWrapperRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/WebViewRenderer_WebClient.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/InnerScaleListener.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/NavigationMenuRenderer_MenuAdapter.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/ViewRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/EditorEditText.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/TableViewRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/DatePickerRenderer_TextFieldClickHandler.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/NavigationMenuRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/EntryRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/AHorizontalScrollView.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/FormsAppCompatActivity.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/InnerGestureListener.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/ViewRenderer_2.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/GenericAnimatorListener.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/BaseCellView.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/EditorRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/VisualElementRenderer_1.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/MasterDetailContainer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/ViewCellRenderer_ViewCellContainer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/FormsTextView.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/CellRenderer_RendererHolder.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/TimePickerRenderer_TimePickerListener.java

					md5b60ffeb829f638581ab2bb9b1a7f4f3f/OpenGLViewRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/TabbedRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/ScrollViewContainer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/CarouselPageAdapter.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/FrameRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/FormattedStringExtensions_FontSpan.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/SliderRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/SearchBarRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/ListViewRenderer_Container.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/ActivityIndicatorRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/SwitchCellView.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/ButtonRenderer_ButtonClickListener.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/ListViewRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/LabelRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/NavigationRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/TextCellRenderer_TextCellView.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/ToolbarButton.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/ButtonRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/ImageRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/CellAdapter.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/ButtonDrawable.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/FormsApplicationActivity.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/MasterDetailRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/ListViewAdapter.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/FrameRenderer_FrameDrawable.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/FormsSeekBar.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/PlatformRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/PageRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/ConditionalFocusLayout.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/OpenGLViewRenderer_Renderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/TimePickerRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/CarouselPageRenderer.java

					md5b60ffeb829f638581ab2bb9b1a7f4f3f/EntryCellEditText.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/FormsImageView.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/DatePickerRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/ObjectJavaBox_1.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/ToolbarImageButton.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/WebViewRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/TableViewModelRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/FormsWebChromeClient.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/GenericMenuClickListener.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/ScrollViewRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/PickerRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/PageContainer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/Platform_DefaultRenderer.java md5b60ffeb829f638581ab2bb9b1a7f4f3f/NavigationMenuRenderer_MenuElementView.java md5acbf8a2bb66a55366b8ef0c0e6ec6127/MyAuthCallback.java md5ba9292671f075a189d6b15232e1c2e4f/MainActivity.java
App can read/write to External Storage. Any App can read data written to External Storage.	high	5.5 medium	CWE-276	M2: Insecure Data Storage	mono/MonoPackageManager.java

PLAYSTORE INFORMATION

Title: Autenticação Gov

Score: 2.4 **Installs:** 100,000+ **Price:** 0 **Android Version Support:** 4.2 and up **Category:** Tools **Play Store URL:** pt.ama.autenticacaogov

Developer Details: AMA, IP, AMA,+IP, None, https://www.autenticacao.gov.pt/, info.portaldocidadao@ama.pt,

Release Date: Dec 15, 2017 **Privacy Policy:** Privacy link

Description:

A aplicação Chave Móvel Digital é uma alternativa ao envio do código de segurança por sms, email ou mensagem direta no Twitter, sendo este enviado através de notificação push para o smartphone que fica associado ao número de identificação civil. É igualmente possível gerar novos códigos e controlar o tempo de vida de cada um. Para mais informação sobre o pedido da chave móvel digital pode consultar a informação no site autenticação.gov.pt.

App Security Score Calculation

Every app is given an ideal score of 100 to begin with.
For every findings with severity high we reduce 15 from the score.
For every findings with severity warning we reduce 10 from the score.
For every findings with severity good we add 5 to the score.
If the calculated score is greater than 100, then the app security score is considered as 100.

And if the calculated score is less than 0, then the app security score is considered as 10.

Risk Calculation

APP SECURITY SCORE	RISK
0 - 15	CRITICAL
16 - 40	HIGH
41 - 70	MEDIUM
71 - 100	LOW

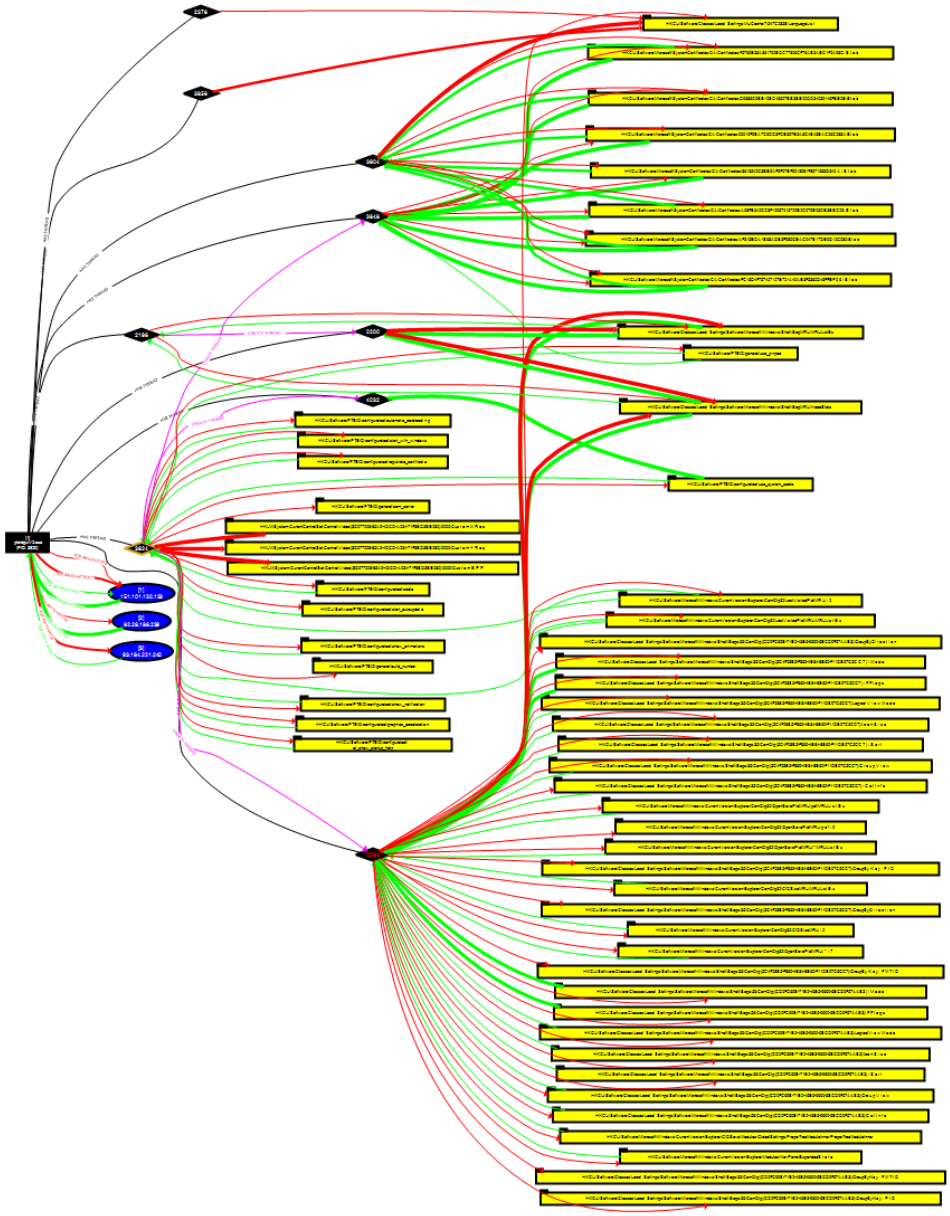
Report Generated by - MobSF v3.0.4 Beta

Mobile Security Framework (MobSF) is an automated, all-in-one mobile application (Android/iOS/Windows) pen-testing, malware analysis and security assessment framework capable of performing static and dynamic analysis.

Anexo B – ProcDot - Registo de alterações durante a assinatura com o Cartão de Cidadão

Apresenta-se a listagem efetuada pela ferramenta Procdot que onde é possível visual de uma forma global todos todas as ligações registadas no momento da assinatura com o Cartão de Cidadão.

link para o vídeo: https://tinyurl.com/y5ovoqll



Anexo C – ProcDot - Registo de alterações durante a assinatura com a CMD

Apresenta-se a listagem efetuada pela ferramenta Procdot que onde é possível visual de uma forma global todos todas as ligações registadas no momento da assinatura com a Chave Móvel Digital.

link para o vídeo: https://tinyurl.com/y4byzyba