Universidade Fernando Pessoa

Avaliação Comparativa do Modelo de Segurança do Android

Nelson Ricardo Santos Sachse

Porto, Dezembro de 2010

Universidade Fernando Pessoa
Praça 9 de Abril, 349
P-4249-004 Porto
Tel. +351-22550.82.70
Fax. +351-22550.82.69
geral@ufp.pt

Avaliação Comparativa do Modelo de Segurança do Android

Nelson Ricardo Santos Sachse

Licenciado em Engenharia Informática pela Universidade Fernando Pessoa

Dissertação apresentada à Universidade Fernando Pessoa como parte dos requisitos para obtenção do grau de Mestre em Computação Móvel, orientada pelo Professor Doutor Feliz Ribeiro Gouveia.

Avaliação Comparativa do Modelo de Segurança do Android

Por:

Nelson Ricardo Santos Sachse

_____

Orientador

Professor Doutor Feliz Ribeiro Gouveia

Universidade Fernando Pessoa

Faculdade de Ciência e Tecnologia

Praça 9 de Abril, 349, 4249-004 Porto, Portugal

Dezembro de 2010

Resumo

Nesta dissertação, pretende-se avaliar, compreender e analisar o modelo de segurança do sistema operativo *Android*, comparativamente a outros modelos de segurança implementados noutros sistemas operativos, como o caso do *iPhone* e *Symbian*.

Para tal, serão identificados os princípios básicos pelos quais os modelos de segurança se devem reger e a sua arquitectura. É descrito o funcionamento do sistema operativo *Android* assim como o seu modelo de segurança.

Serão avaliados os modelos de segurança dos sistemas *iPhone* e *Symbian* para ser possível obter uma base comparativa para o modelo *Android*.

No fim, é feita uma avaliação do modelo de segurança *Android*, na qual será verificado se haverá necessidade de implementação de outros mecanismos presentes noutros *smartphones*.

Abstract

This thesis, aims to assess, understand and analyze the security model of the Android operating system, compared to other security models as the iPhone and Symbian.

Therefore, we will identify what the basic principles by which security models are governed and its architecture. The workflow of the Android operating system and his security model, as well as the iPhone and Symbian.

In the end, we obtained an evaluation of Android security model, which will be checked whether there is need to implement other mechanisms that are present in other smartphones.

Agradecimentos

A elaboração desta dissertação foi um trabalho de empenho pessoal, dedicação e coragem, mas não o seria capaz de o realizar sem a ajuda de algumas pessoas.

Quero desde já agradecer ao meu orientador, professor Feliz Gouveia, por toda a ajuda e dedicação que disponibilizou mas acima de tudo por me fazer atingir níveis de exigência que eu não pensava ter.

A minha equipa, pelo indubitável incentivo diário, em particular a Dra. Ana Matos e ao Dr. João Alves que tornaram possível esta realidade.

Aos meus pais, que sempre me apoiaram em todo o meu percurso académico com bons conselhos e dedicação, sem isso não seria quem sou hoje. Não esquecendo também a Liberta, que no silêncio do momento sabia o que dizer nem que fosse com a sua presença.

A Cláudia, por todas as palavras certas nos momentos certos e por todas as vezes que me fez acreditar, sem isso, não teria sido possível.

Lista de Acrónimos

SMS	Short Message Service
API	Application Programming Interface
BSD	Berkeley Software Distribution
USB	Universal Serial Bus
SID	Secure Identifier
VID	Vendor Identifier
ARM	Advanced RISC Machine
ICC	Inter-component communication
MAC	Monitor Access Control
RVM	Reference Validation Mechanism
VM	Virtual Machine
XML	Extensible Markup Language
DARP	Data At Rest Protection
IOS	iPhone Operative System
VPN	Virtual Private Network
CMS	Cryptographic Message Syntax
DES	Data Encryption Standard
AES	Advanced Encryption Standard
IPSec	Internet Protocol Security
L2TP	Layer 2 Tunneling Protocol
PPTP	Point-to-Point Tunneling Protocol
SSL	Secure Sockets Layer
WPA	Wi-Fi Protected Access
PIN	Personal Identification Number
MMU	Memory Management Unit
EULA	End User License Agreement
SSL	Secure Sockets Layer
WPA	Wi-Fi Protected Access
PIN	Personal Identification Number

Lista de Ilustrações

Índice

1. Introdução

Com os avanços tecnológicos alcançados, os dispositivos móveis têm muito mais funcionalidade que os telefones móveis do passado. Cada vez mais são utilizados da mesma forma que computadores pessoais, tornando-os potencialmente susceptíveis às mesmas ameaças que afectam os computadores ligados à internet. No entanto para um utilizador comum a preocupação com a segurança do seu telefone será provavelmente uma das suas menores preocupações.

Visto isto, o desafio que nos surge é, será um modelo de segurança capaz de manter a integridade do seu sistema sem a interacção do utilizador?

1.1. Descrição do problema

Uma das dificuldades na construção de modelos de segurança reside em conter o seu sistema intacto em todas as situações; no entanto quando existe a necessidade de interagir com o utilizador deverá ser o mais conciso e explicativo possível. Por exemplo, é exequível mostrar ao utilizador o que vai permitir ou negar com uma determinada acção, seja ela a instalação de uma aplicação ou a permissão de um acesso.

Como tal, o modelo de segurança tem de ser capaz de manter o sistema íntegro assim como evitar a passagem de decisões críticas para o utilizador, já que se mal tomadas podem afectar a integridade do sistema.

No caso particular do *Android*, será avaliado que tipo de arquitectura implementou face a interacção com o utilizador, assim como a sua organização e o funcionamento dos seus mecanismos de segurança.

1.2. Objectivos do trabalho

O objectivo do trabalho consiste em analisar, compreender e propor possíveis alterações ao modelo de segurança do *Android*. Será para isso feito o estudo dos princípios de modelos de segurança e a investigação necessária dos modelos de segurança escolhidos (*Android*, *iPhone* e *Symbian*) para obter uma sólida base de comparação com o modelo de segurança do *Android*. Após a investigação dos documentos técnicos, será então possível propor possíveis melhoramentos ao modelo de segurança do *Android*.

1.3. Motivação

A motivação para o estudo de modelos de segurança recai no facto da segurança ser um elemento que passa muitas das vezes completamente despercebida aos utilizadores. Vezes sem contas a questão da segurança fica para segundo plano para utilizadores comuns e como tal não se conseguem aperceber da sua importância até ser tarde de mais.

Visto isto, a responsabilidade de manter a integridade dos sistemas operativos usados fica a cabo das empresas que os desenvolvem. Desse modo, a estruturação de um modelo de segurança é muito importante, pois na maior parte das vezes são eles que tomam as decisão mais criticas que evitam perda de dados ou a integridade do dispositivo.

Nesse caso, se um modelo de segurança não estiver a funcionar correctamente ou com determinadas lacunas, será o utilizador que vai sofrer as suas consequências no fim.

1.4. Metodologia

A estruturação da metodologia foi baseada na investigação dos princípios de modelos de segurança, passando pela aplicação de modelos de segurança em *smartphones*, a examinação de documentos técnicos dos respectivos sistemas operativos : *Android*, *iPhone* e *Symbian*. Esta análise da literatura já feita permite obter uma base sólida para uma comparação entre modelos de segurança com o modelo do *Android*.

1.5. Estrutura do documento

A estrutura do documento é apresentado de modo a que seja possível partir das definições de *smarphones*, para os conceitos de modelos de segurança e terminar com uma avaliação do modelo do *Android*.

No segundo capítulo, será feita uma contextualização em relação ao uso de *smartphones* tendo em atenção as suas tendências perante sistemas operativos assim como ao seu uso no mercado Português.

No terceiro capítulo, serão identificados os princípios de segurança pelos quais os modelos de segurança se regem mas também a estrutura e conceitos base na concepção de modelos de segurança.

No quarto capítulo, é identificado e explicado ao pormenor o sistema operativo *Android*, passando pela sua arquitectura até ao funcionamento de uma aplicação no seu ambiente.

No quinto capítulo, são analisados os modelos de segurança dos sistemas operativos previamente identificados, respectivamente, o modelo do *Android*, *iPhone* e *Symbian*.

No sexto capítulo, é realizada uma comparação entre os modelos de segurança do *iPhone* e da *Symbian* com o modelo do *Android*.

No sétimo capítulo, são apresentadas as conclusões do trabalho.

2. Contextualização

Cada vez mais usamos os telemóveis não apenas para fazer chamadas ou receber mensagens, mas também como dispositivos móveis que nos permitem atingir os mais diversos fins, sejam eles utilizar caixas de correio electrónico, aceder a redes sociais ou até mesmo consultar contas bancárias.

Hoje em dia, os *smartphones* são como pequenos computadores portáteis, capazes de conter toda a informação necessária de um utilizador para tarefas diárias que ultrapassam largamente o uso de um telemóvel.

Desde o calendário até a agenda de tarefas, o dia-a-dia é guardado num único dispositivo. A questão que se coloca é, será seguro guardar todo este tipo de informação num dispositivo móvel como um *smartphone*? Darão o sigilo e a segurança necessária para guardar todos os contactos disponíveis na agenda assim como os documentos de trabalho importantes?

Só no ano passado em Londres foram deixados mais de dez mil dispositivos móveis por mês em táxis, segundo foi noticiado pela revista *Newslite* (newslite, 2009). É por isso que verificamos que a segurança é importante, imprescindível, pois se hoje em dia damos valor aos dados que temos no telemóvel, então é necessário proteger do mesmo modo que protegemos todos os outros bens materiais.

2.1. Smartphones em Portugal

O crescimento da utilização de *smartphones* também se verificou em Portugal, e como tal houve um mercado emergente que se tornou bastante acentuado.

No segundo trimestre de 2010 tinham sido vendidos 1,7 milhões de telemóveis. Comparativamente com o ano de 2009 houve um crescimento de 16% nas vendas, segundo o estudo realizado pela "IDC Mobile Phone Tracker". Houve um maior crescimento na área de *smartphones* com um aumento de 79% em relação ao segundo trimestre de 2009, pois houve mais de 270 mil unidades vendidas representado 16% das vendas de telemóveis no trimestre.

"*Nos dois primeiros trimestres de 2010 o mercado Português registou um forte crescimento das vendas no segmento dos smartphones, o que denota que estes telefones estão claramente a entrar nas preferências de compra da maioria dos consumidores, atraídos pelas novas funcionalidades, e pelo facto do preço médio destes terminais continuar a descer, tornando-os bastante competitivos em comparação com os restantes telefones*", segundo Francisco Jerónimo, Responsável Europeu de pesquisa da área de telefones móveis da IDC.

De salientar também o facto de o *Android* se ter tornado o segundo sistema operativo mais vendido em Portugal depois da *Symbian*, visto que o seu crescimento foi de 675%, segundo as vendas de terminais com este sistema operativo, respectivamente ao período de 2009.

Tal facto, deve-se principalmente a popularidade dos terminais terem uma experiência em muito semelhante ao telemóvel da *Apple,* o *iPhone*, principalmente na interacção com o utilizador e da parceria com a *Samsung*¸ como é possível identificar na tabela seguinte.

Fabricante	**Vendas 2T 2010 (milhares)**	**Mercado 2T 2010**	**Vendas 2T 2009 (milhares)**	**Mercado 2T 2009**	**Variação Vendas 2T 2010 vs 2T 2009**
1. Nokia	671	40%	656	45%	2%
2. Samsung	491	29%	524	36%	-6%
3. LG	181	11%	97	7%	87%
4. Vodafone*	168	10%	88	6%	91%
5. Sony Ericsson	58	3%	26	2%	123%
6. Outros	123	7%	71	5%	73%
Total	**1,692**	**100%**	**1,462**	**100%**	**16%**

Tabela 1- Principais Fabricantes e Quotas de Mercado em Portugal - 2º Trimestre 2010 - Smartphones (Fonte : IDC, Setembro 2010)

Relativamente ao mercado internacional é possível verificar que o sistema Android subiu consideravelmente em relação a passagem do ano de 2008 para o ano de 2009. Como é possível constatar na tabela seguinte.

Empresa	2009 Unidades	(%)	2008 Unidades	(%)
1. Symbian	80,878.6	46.9	72,933.5	52.4
2. Research In Motion	34,346.6	19.9	23,149.0	16.6
3. iPhone OS	24,889.8	14.4	11,417.5	8.2
4. Microsoft Windows Mobile	15,027.6	8.7	16,498.1	11.8
5. Linux	8,126.5	4.7	10,622.4	7.6
6. Android	6,798.4	3.9	640.5	0.5
7. WebOS	1,193.2	0.7	NA	NA
8. Other OSs	1,112.4	0.6	4,026.9	2.9
Total	**172,373.1**	**100.0**	**139,287.9**	**100.0**

Tabela 2 - Vendas mundiais de Smartphones por Sistema Operativo em 2009 (Fonte: Gartner, Fevereiro 2010)

Da análise da tabela é evidente que o mercado do *Android* esteve em crescimento, no entanto ainda precisará de algum tempo até marcar a sua presença no mercado internacional como um dos principais sistemas de dispositivos móveis.

2.2. Malware em Smartphones

Segundo um estudo recente (US-Cert, 2010), a tecnologia móvel tem sido a tecnologia mais rapidamente adoptada na história, com um valor estimado 4,6 bilhões de assinaturas de telemóveis a nível mundial no final de 2009. Além disso, os avanços tecnológicos têm contribuído para uma capacidade de computação em dispositivos portáteis sem precedentes. A crescente dependência do utilizador em dispositivos móveis, a nível mundial, assim como assinaturas de banda larga móvel cresceu mais de 850% em 2008, excedendo o número de assinantes de banda larga fixa. Os dispositivos móveis têm-se tornado parte integrante da sociedade e, para alguns, uma ferramenta essencial. No entanto, a funcionalidade e concepções complexas destes dispositivos introduziram vulnerabilidades adicionais. Essas vulnerabilidades, juntamente com a quota de mercado em expansão tornam a tecnologia móvel um alvo atraentemente viável e gratificante para aqueles interessados em explorá-las.

2.2.1. Taxionomia de Malware e Ataques

Visto serem cada vez mais comuns, os vírus para dispositivos móveis são semelhantes aos previamente desenvolvidos para computadores pessoais e hoje já são possíveis de classificar assim como de agrupar por tipo de ataques ou vulnerabilidade.

Segundo (Dunham, K., et al. 2009), estes são as definições actuais de algumas ameaças que existem em dispositivos móveis.

- **Ad/Spyware**

  Programas potencialmente indesejados, que podem incluir uma EULA (*End User License Agreement*). Este tipo de programa executa uma série de acções, por norma sem o conhecimento por parte do utilizador. Geralmente estudam o perfil do utilizador, adquirindo as suas preferências e gostos. Podem também envolver o aparecimento de anúncios (*pop-up´s*) normalmente sobre algum tipo de informação que tenha conseguido apurar.

- **Bluebug**

  Explora uma vulnerabilidade na segurança do Bluetooth para criar chamadas telefónicas para números com valor acrescentado. É possível também tirarem partido da ligação à internet.

- **Buffer-overflow**

  É recorrente quando o tamanho de um *buffer* ultrapassa a sua capacidade máxima de armazenamento. Caso o programa não tenha sido adequadamente desenvolvido, esse excesso de dados pode acabar por ser armazenado em áreas de memória próximas, corrompendo dados ou terminando o programa.

- **Brute Force**

  Algoritmo que consiste em enumerar todo os possíveis resultados de uma solução e verificar se cada um satisfaz o problema. Utilizado para atacar nomes de utilizadores e senhas de acesso.

- **Denial-of-Service (DoS)**

  Um ataque destinado a prejudicar e / ou negar o uso de um dispositivo, serviço ou rede.

- **Exploit**

  Software ou acções que tentam aproveitar uma vulnerabilidade para executar acções indesejadas.

- **Hacking Defaults**

  Uma técnica usada para aceder a dispositivos por software no qual são usadas senhas padrão, as quais permitem alterar as configurações do sistema.

- **Mobile Malware**

  Software com acções maliciosas, designado para afectar dispositivos móveis e software. Também conhecido como *malware*, vírus e código malicioso.

- **Rogue Software**

  Software ilegal arquitectado para induzir o utilizador a comprar um software de modo a eliminar potenciais vírus. Este tipo de programas inclui frequentemente uso limitado, erros em resultados após análises do sistema, avisos e outros tipos de simulações que forcem o utilizador a comprar um produto desnecessário.

- **Payload**

  A carga útil de um código malicioso. Exemplo: um Trojan pode ser usado para instalar *Rogue Software*, onde o *Rogue Software* é o *payload* do ataque.

- **Trojan**

  Um cavalo de Tróia é um *software* malicioso que se disfarça em algo que não é, no entanto não se replica.

- **Virus**

  O *software* malicioso que infecta um ficheiro para se espalhar pelo sistema.

- **Worm**

  Software malicioso que cria uma cópia de si mesmo.

A maior partes das ameaças descritas fizeram a sua passagem dos computadores pessoais para os dispositivos móveis. Embora muitas delas possam ser variantes das originais, os seus objectivos de obter informações do utilizador ou danificar o dispositivo continuam a ser os mesmos.

2.3. Segurança em Smartphones

Dispositivos móveis são críticos na sociedade de hoje. Com cada vez menos barreiras tecnológicas, empresas e indivíduos necessitam deste tipo de dispositivos para se tornarem comunicáveis em qualquer altura. E embora todos eles disponibilizem uma série de recursos existem riscos, desde problemas com a perda de dispositivos, até *malware* ou acessos externos não autorizados. Sendo por isso necessário a implementação de uma gestão de risco e de prevenção na eventualidade de algum tipo de falha acontecer em tais dispositivos. Analisando a situação verificamos que estas falhas se devem ao seu maior problema e à sua maior vantagem: a portabilidade.

Com a portabilidade aumenta a possibilidade de comunicação em qualquer altura, mas ao mesmo tempo é possível a utilização de redes sem fios sem uma transmissão segura. Isto verifica-se em ligações sem fios como o *wi-fi* ou *bluetooth* que por vezes permitem aos *smartphones* enviarem e receberem informações sem o conhecimento dos utilizadores.

Visto que as redes sem fios são usadas para transmitir pacotes de dados sem a necessidade do uso de cabos são mais facilmente interceptadas, podendo infectar o *smartphone* com *malware*, que pode activar facilmente funcionalidades físicas nativas do *smartphone*, como o caso de câmaras, microfones ou até mesmo comprometer a capacidade de armazenamento de dados que existe no dispositivo. Este tipo de ataques são perpetrados através do uso de ferramentas de acesso remoto, que podem ser instaladas no dispositivo permitindo aceder a sua informação. Eventualmente é possível restringir este tipo de ataques através do uso de cifras na comunicação com o exterior e com o uso de aplicações de segurança de terceiros.

Segundo a ISACA (2010), os dispositivos móveis podem ter várias vulnerabilidades que podem ser aproveitadas por código malicioso, assim como outros tipos de ameaças. É possível identificar alguns desses ataques nos quadros seguintes.

Ameaça	Intrusões maliciosas que podem danificar dispositivos
Vulnerabilidade	Informação viaja através de redes sem fios, as quais podem ser menos seguras que as redes físicas.
Risco	Intercepção de informação que pode resultar numa captura de dados, comprometer a integridade de uma empresa, acções legais

Ameaça	A passagem de dispositivos por diversas redes cria a possibilidade de introduzir *malware* em outras redes.
Vulnerabilidade	Portabilidade do dispositivo pode criar o abandono da segurança imposta pela empresa e os seus protocolos de segurança.
Risco	Propagação de *malware*, a qual pode resultar em fugas de informação, corrupção de dados e indisponibilidade dos mesmos.

Ameaça	Atacantes podem descobrir a disponibilidade do acesso por *bluetooth* e tentar ataques.
Vulnerabilidade	Utilização indevida do *bluetooth*, que por vezes se pode encontrar disponível sem o conhecimento do utilizador.
Risco	Corrupção de dados, perda de dados, intercepção de chamadas e exposição de informação.

Ameaça	Em caso de ocorrer uma intercepção maliciosa em que informação é adquirida, os dados podem ser interpretados e usados.
Vulnerabilidade	Informação não cifrada guardada no dispositivo.
Risco	Exposição a dados sensíveis que podem danificar a imagem de uma empresa, colaboradores e clientes.

Ameaça	Dispositivos móveis podem ser perdidos ou furtados no entanto os seus dados nem sempre são recuperados.
Vulnerabilidade	Perda de dados pode prejudicar a produtividade de um colaborador.
Risco	Colaboradores dependem de dispositivos móveis, caso tenham perdido as suas informações não podem executar as suas tarefas.

Ameaça	Na eventualidade de furto ou perda, desconhecidos podem aceder ao dispositivo e aos seus dados.
Vulnerabilidade	Os dispositivos não tem métodos de autenticação aplicáveis.
Risco	Exposição de dados, resultante em denegrir a imagem da empresa assim com o a sua fiabilidade.

Ameaça	Se não existir politicas de acesso com dispositivos móveis, os colaboradores podem utilizar os seus próprios dispositivos que podem eventualmente aceder a documentos sensíveis, como o caso de correio electrónico.
Vulnerabilidade	A empresa não controla o dispositivo. A gestão é feita pelo utilizador.
Risco	Propagação de *malware*, fuga de informação.

Ameaça	O dispositivo permite a instalação de aplicações de terceiros que não tenham assinaturas digitais.
Vulnerabilidade	A empresa não controla o dispositivo. A gestão é feita pelo utilizador.
Risco	Propagação de *malware*, fuga de informação, intrusão na rede da empresa.

Segundo a própria ISACA, estes tipos de ameaças são as mais comuns e aquelas que ocorrem com maior frequência nos dispositivos móveis.

2.4. Segurança em Smarphone : Visão do Utilizador

Para o utilizador comum de *smartphone*, a segurança é o mínimo das suas preocupações, pois assume que é um telemóvel e que ainda não existem tantos perigos como os existentes num computador. No entanto, na realidade, os factores de perigo a que estão expostos são bastantes mais frequentes do que podem pensar e quando acontecem podem não perceber a razão de tal.

Um exemplo recente disso aconteceu com o *iPhone,* o qual foi bloqueado por um software quando sentiu falta de segurança no sistema (o que visto do ponto de segurança foi uma boa medida), no entanto, para o utilizador não é aceitável (por razões de usabilidade) e como tal forçou a *Apple* a desbloquear todos os aparelhos diminuindo assim a sua segurança.

Segundo (Becher, M., et al. 2007), o utilizador comum não tem conhecimento profundo de segurança e mesmo que a sua consciência sobre segurança possa ter sido aumentada através de incidentes (*worms* ou *vírus*), continua a ser questionável, se será capaz de diferenciar entre os diferentes produtos de segurança e de como usá-los correctamente. Acreditando que se trata de um utilizador consciente dos perigos e se preocupa com a segurança, é lhe difícil perceber quais os programas ou documentação sobre a qual estar a par, pois as mais simples técnicas de engenharia social podem ser aplicadas com sucesso por um período indeterminado de tempo.

Podemos assumir ainda, que para um utilizador comum o dispositivo móvel é considerado um item mais descartável que um computador pessoal, logo a solução de segurança deve ser de custo mais reduzido. Devido a estes factos, é de salientar que os utilizadores precisam de uma solução que se encontre incorporada no manuseamento normal do dispositivo, em vez de uma solução separada. Mesmo que existam utilizadores que não queiram perder tempo com a segurança pode haver outros que sintam que os seus dados são extremamente importantes e não podem arriscar perdê-los.

Desse modo, um modelo de segurança que possa estar integrado no sistema e funcione em segundo plano, com a necessidade de intervenção mínima por parte do utilizador é uma necessidade real e imprescindível em dispositivos móveis.

3. Princípios de Modelos de Segurança

Etimologicamente a palavra segurança (segurar+-ança) significa confiança, certificação, garantia, certeza, por outras palavras, o que não se vai perder ou danificar. No conceito de computação, segundo (Ware W., 1967) o termo segurança descreve técnicas que controlam quem pode usar ou modificar o computador ou então a informação que eles contêm. Já para Bruce Schneier (especialista de segurança), existe na nossa sociedade a ideia que a segurança é a ordem natural das coisas, e que devemos estar complemente seguros constantemente, o que se avaliarmos correctamente, verificamos que é falso, a vida é risco e não existe algo como segurança total. Schneier afirma ainda que não podemos apenas ver os seus benefícios, é necessário também ver os custos, visto ser importante avaliar o que estamos dispostos a sacrificar em prol de atingir níveis significantes de segurança (Schneier B., 2009).

Mas quando falamos de protecção de dados informáticos temos de nos precaver de muitas outras situações que ficam para lá de um mero conceito fechado de segurança. É preciso pensar que nos dias de hoje há cada vez mais furtos electrónicos, tentativas de intrusão em sistemas, código malicioso (*malware* e *spyware*), entre muitos outros problemas. Segundo (Antonelli, C., 2009) existem as mais diversas ameaças aos dados, sejam elas, por corrupção, por perda de dispositivos, roubo do dispositivo móvel, roubo de dados, perda da chave de cifra, ou qualquer outro tipo de compromisso. É por isso que preparamos os nossos sistemas de modo a proteger de algum ataque ou eventualidade que possa acontecer. Desse modo precisamos que o sistema forneça o tipo de segurança necessário para evitar este tipo de situação.

Conhecendo os perigos que enfrentam os utilizadores de sistemas operativos dos dispositivos móveis, os fabricantes seguem alguns princípios de modelos de segurança, que embora tenham sido já elaborados há algum tempo ainda se mantém actuais, princípios como os de Saltzer e Schroeder (Saltzer et al. 1975) e os princípios de Dennis e Van Horn (Dennis et al. 1972). Princípios estes que são utilizados em muitos dos modelos de segurança de hoje. Com a utilização destes princípios e a concepção de novas soluções, é possível eliminar as mais conhecidas ameaças assim como obter um resistente modelo de segurança.

Este capítulo identifica a taxionomia utilizada por um modelo de segurança, identificar os primeiros modelos de segurança elaborados para os sistemas distribuídos e define princípios base na criação de modelos de segurança.

3.1. Taxionomia dos conceitos de modelos de segurança

De modo a ser mais compreensível os conceitos inerentes a um modelo de segurança foi elaborada uma taxinomia dos elementos contribuintes. Como tal, vamos identificar cada um dos elementos que fazem parte de um sistema seguido de uma descrição da sua função.

- **Componente**

  Qualquer parte do sistema, que por si própria, representa o todo ou parte de uma funcionalidade do sistema.

- **Mecanismos de segurança**

  Sistema usado para a utilização de políticas de segurança.

- **Modelo de referência**

  Um conceito de controlo de acessos que tem a função de mediador entre todos os acessos a objectos por entidades activas.

- **Modulo**

  Uma unidade de computação na qual se encapsula uma base de dados e uma interface para a inicialização, devolução ou retorno de informação.

- **Processo**

  Programa em execução.

- **Serviço**

  Processamento ou protecção derivada de um componente que serve utilizadores ou componentes.

- **Sistema**

  É composto por um ou mais componentes, os quais podem estar ligados entre si.

Com este tipo de organização estão identificados os elementos que constroem um sistema.

Deste modo, os modelos de segurança podem verificar quais os elementos que se podem tornar mais expostos ou aqueles aos quais se devem restringir os acessos.

3.2. Modelos de Segurança

Os modelos de segurança que se seguem, são denominados como a base de muitos dos modelos que são criados hoje em dia. Eles contem os princípios básicos do comportamento de um sistema, assim como a identificação da estrutura a seguir para a construção de um modelo de segurança mais completo.

3.2.1. Princípios de Saltzer e Schroeder

Segundo (Saltzer et al. 1975), a base de um modelo de segurança deve seguir oito princípios de modo a poder proteger correctamente tanto o sistema como toda a informação disponibilizada do utilizador. Os seus princípios são descritos a seguir.

Economia dos mecanismos

Este princípio diz que se deve manter a concepção simples e a mais pequena quanto possível. Este princípio bem conhecido aplica-se a qualquer aspecto de um sistema, mas merece um destaque para os mecanismos de protecção por uma razão: erros de implementação. Estes erros de implementação, que normalmente resultam em acessos indesejados, não são observados durante o uso normal de um utilizador (visto que ele não vai tentar incluir tentativas para utilizar caminhos de acessos indevido). No entanto podem ser identificados por outro tipo de utilizador mais experiente ou que esteja à procura de eventuais falhas do sistema. Como resultado, as técnicas de inspecção linha a linha do *software* e exames físicos ao *hardware* são necessárias.

Falta de acesso

Este princípio indica-nos que o sistema por defeito deve ter falta de acesso. Ou seja, nunca deve permitir acesso a nenhum componente ou funcionalidade, a não ser que lhe sejam indicadas. Deste modo mantendo a sua integridade.

Mediação

No princípio da mediação o acesso a um objecto protegido deve ser sempre verificado e qualquer tipo de alteração deve ser analisada metodicamente. Quando aplicada sistematicamente, é um dos fundamentos do sistema de protecção. Força um controle de acesso em todo o sistema, que, além do funcionamento normal inclui a inicialização, recuperação, terminação e manutenção do sistema em causa.

Concepção Aberta

Neste princípio assimila-se que a segurança de um sistema não deve depender de manter um mecanismo secreto. Os sistemas abertos baseiam-se na protecção através de chaves ou palavras passe e não através da ocultação da sua arquitectura.

Separação de Privilégios

Este princípio afirma que um mecanismo de protecção que requer a utilização de duas chaves é melhor do que um mecanismo que apenas precise de uma. A razão é que, uma vez que o mecanismo é bloqueado, as duas chaves podem ser fisicamente separadas e programas, organizações ou indivíduos distintos ficam responsáveis por elas. A partir deste pressuposto, nenhum acidente único, engano, ou quebra de confiança é suficiente para comprometer as informações protegidas.

Menor Privilegio

Este princípio diz que uma aplicação deve ser executada com a uma pequena lista de permissões de modo a reduzir o risco de falhas do sistema. Deste modo, limita os danos que podem resultar de um acidente ou erro e reduz também o número de possíveis interacções entre programas privilegiados para o mínimo.

Redução de Mecanismos Comuns

Este princípio assume que o número de serviços comuns usados por múltiplos programas deve ser reduzido a um mínimo, de modo a prevenir uma potencial fuga de informação entre programas. Cada mecanismo comum (especialmente aqueles que envolvem variáveis partilhadas) representa um caminho de informação entre utilizadores e como tal deve-se desenvolver com muito cuidado para ter a certeza de que não há risco de comprometer a segurança.

Aceitação Psicológica

No princípio de aceitação psicológica, a interacção do utilizador com o interface deve ser fácil e ir de encontro ao que ele acha que são os mínimos de segurança. Se a imagem mental projectada pelo utilizador dos seus mecanismos de segurança, for de encontro aos objectivos de protecção que deve usar então os erros serão minimizados. Por exemplo no acesso a páginas de internet no qual um certificado não está validado deve ser identificado ao utilizador, mas a partir do momento que está autorizado pelo mesmo, não deverá ser mais feita essa questão durante a sua utilização.

3.2.2. Princípios de Dennis e Van Horn

Segundo (Dennis et al. 1972), a criação de um mecanismo de validação de referência, RVM (*Reference Validation Mechanism*), reforça o sistema.

O RVM consiste em um mecanismo no qual uma base de computação confiável (*hardware*, *firmare* ou *software* responsáveis por implementar políticas de segurança) tem a função de controlar acessos entre sujeitos ou objectos e cuja correcta implementação é essencial para a protecção de dados no sistema. Sendo que os seus principais conceitos são os seguintes:

O mecanismo de validação de referência deve ser inviolável: se o mecanismo de validação de referência pode ser alterado programaticamente ou manualmente, então a sua integridade não pode ser garantida, e como tal, nenhum certificado de segurança pode ser emitido.

O mecanismo de validação de referência deve ser sempre invocado: este mecanismo de referência deve ser utilizado em todos os programas incluindo o sistema operativo.

O mecanismo de validação de referência de ser pequeno o suficiente para ser sujeito a análise, testes e tudo o que for necessário para que possa ser assegurado: o mecanismo deve ser pequeno o suficiente para que se possa demonstrar logicamente que é completo, então não só é fiel ao seu modelo, e correctamente implementado, mas também é capaz de provar que é correcto.

A utilização destes princípios cria um elo muito forte na troca de parâmetros entre os componentes, já que o sistema assume que não haverá problemas após ter sido feita a verificação através de mecanismos de validação.

3.3. Segurança na Concepção de Soluções

Manter os princípios de segurança durante o desenvolvimento de uma solução de um programa ou aplicação é um objectivo que deverá visar a consistência e a integridade. Segundo (Levin, T. et al. 2007) a fase de concepção de uma solução que envolve princípios de segurança e deve passar obrigatoriamente por três fases: estrutura, lógica (e funções) e ciclo de vida do sistema. Cada uma dessas fases vai ser descrita e explicada de modo a identificar a sua importância durante o desenvolvimento de uma nova implementação de segurança. Na seguinte figura é possível identificar a esquematização de cada uma dessas fases:

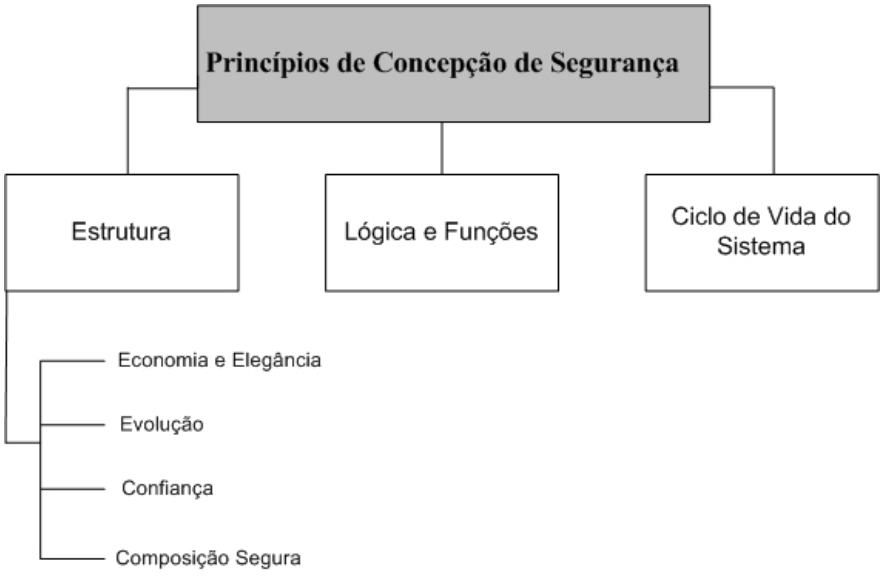

Ilustração 1 - Esquema dos princípios de concepção de segurança

Verifica-se na imagem que a fase da estrutura pode ainda ser subdividida em mais quatro grupos no entanto para as outras fases não será necessário.

3.3.1. Estrutura - Como é efectuada a organização

A concepção da estrutura afecta os princípios fundamentais do sistema, como os componentes se relacionam entre si ou com as interfaces. É possível subdividir em quatro grupos: economia e elegância, evolução, confiança e composição segura.

Economia e Elegância

O tema da economia e elegância recai em fundamentos da arquitectura do sistema na qual a simplicidade é a principal regra a seguir. Definir e desenvolver interfaces e funções de modo consistente e intuitivo ajuda a obter a elegância desejada, promovendo assim uma mais fácil análise, inspecção e rigor na utilização de depuradores. Seguindo técnicas como a de limitar as funções a ter apenas um ponto de entrada e um de saída, utilizar valores estipulados para as funções em vez de apontadores ajuda não só a manter a economia do sistema como também a sua elegância. Tal reflecte-se na manutenção do sistema, já que modificações em funções comuns obtêm um impacto mais reduzido e podem ser percebidas. A utilização de camadas, um mecanismo de gestão de memória no hardware, a limitação de acessos entre componentes e objectos (processos e funções), utilização de virtualização (permitindo a criação de componentes em espaços privados de memória sem acessos indevidos) e a redução de complexidade do sistema aumentam a segurança mas também conseguem criar um sistema arquitectonicamente harmonioso e de fácil utilização.

Evolução

Este princípio assume que um sistema é capaz de ser actualizado com a passagem do tempo. Visto que um sistema tem de ser actualizado e reconfigurado durante o seu ciclo de vida, é por vezes necessário que esteja preparado para ser modificado para novas versões. Como tal, é necessário que o sistema esteja preparado para o caso de ser necessário algum tipo de intervenção.

Confiança

Os princípios de confiança afectam todas as relações entre os componentes assim como as suas dependências. Por exemplo, no mecanismo de certificados encadeados, só teremos acesso ao certificado de nível inferior se o anterior tiver a confiança do sistema. Numa hierarquia de certificados, o certificado raiz é aquele no qual existe maior confiança, o mesmo acontece quando se verifica quando se tenta aceder ao *kernel* de um sistema operativo. Como tal, cada componente deve ter privilégios suficientes para executar as suas funções mas nada mais, verificando assim que alguma falha por corrupção será minimizada.

Composição Segura

Relativamente a um sistema distribuído, se existir uma composição segura, na qual os componentes distribuídos aplicam as mesmas políticas de segurança que os componentes individuais usam, então a comunicação entre protocolos e os mais variados mecanismos de distribuição de dados asseguram que a consistência do sistema distribuído é mantida. O uso de técnicas, como restringir o acesso através de um mecanismo de controlo apropriado, uso de um monitor de referência em cada um dos componentes, ou usar canais cifrados elimina assim vulnerabilidades em comunicação de canais inseguros.

3.3.2. Lógica e Funções

Relativamente à fase de lógica e funções, são aplicados tanto ao sistema como ao nível do componente diferentes conceitos que fazem com que o sistema atinja um nível de segurança elevado. Esses conceitos são identificados em seguida.

Como a protecção da informação necessita de um sistema de políticas de acesso, é necessário existirem pressupostos na arquitectura do sistema que façam com que a integridade, confidencialidade e privacidade não será deixada sem protecção durante o controlo do sistema. Assim, todos os pedidos devem ser validados e o próprio mecanismo de referência deve-se proteger a si mesmo, e no caso de haver um falha total do sistema, deverá existir um mecanismo de recuperação de dados (*rollback*).

O uso de meta dados é outro tipo de protecção de informação, que assegura que dados sensíveis são disfarçados através de outro tipo de denominação. Assim, a renomeação de directórios ou ficheiros pode ser uma técnica de simples uso mas extremamente eficaz.

Outro princípio que usa lógica e o uso de funções é a utilização de uma auto-analise por parte do sistema, a qual permite criar uma cadeia de confiança. Quando tal acontece, sabemos que quando o sistema autoriza um determinado nível ou componente é porque já verificou a integridade do anterior, como no exemplo de um *boot* de um computador na qual desde o nível mais inferior ao superior é testada a confiança em cada um dos componentes.

Existe também a necessidade de identificar possíveis falhas e desta maneira recapitular todo o processo até à sua origem, por isso, a utilização de auditores permite que seja possível ao programador identificar mais rapidamente o problema. Para que não entrem em conflito com a politica de segurança do sistema, pois se tal aconteceu e não foi contemplado um plano de contenção poderá causar danos irremediáveis. Durante a elaboração da arquitectura do sistema é obrigatório que sejam criadas configurações padrão, para que se consiga proteger o sistema desde o primeiro instante.

Os mecanismos de segurança devem ser construídos de modo a não degradar a performance do sistema. Como um dos métodos mais usados para a protecção de dados é o uso de cifras, então, a atenção recai sobre qual a melhor cifra a usar para proteger o respectivo modelo. Pois se não for necessário usar uma cifra bastante forte pode causar questões de performance muito críticas. Se necessário, então é possível adaptar este tipo de desempenho para os mecanismos de *hardware* obtendo melhor processamento.

Manter um sistema ergonómico e seguro ajuda o utilizador a ter um interface assim como serviços de suporte de fácil utilização. Do mesmo modo, se for possível não pedir autenticações e autorizações desnecessárias ao utilizador então o sistema torna-se mais intuitivo e menos intrusivo. No entanto, nestes casos será necessário adaptar a arquitectura do sistema para este tipo de ambiente e sempre que possível guardar a informação do utilizador na memória do dispositivo (ou em *cache*) para que seja possível aplicar politicas de segurança do sistema.

3.3.3. Ciclo de Vida do Sistema

Com o ciclo de vida do sistema, é possível tornar o sistema de fácil manutenção assim melhorar a sua elegância, já que usa princípios que se baseiam no *Common Criteria* (2010). O sistema deve fornecer a identificação de quais são os seus objectivos assim como o conceito do seu funcionamento. A documentação, como toda a respectiva informação, deverá ser fornecida para o utilizador, de modo a poder contribuir para um sistema mais seguro com o conhecimento a ser transmitido para quem o necessita.

Na eventualidade de haver alguma actualização do sistema, deverá haver a atenção para que nenhuma alteração da estrutura seja feita. Um dos cuidados que se deverá ter na actualização do sistema, para um nova versão, será de não perder nenhuma das suas funcionalidades existentes e não mudar a arquitectura do sistema, pois, caso contrário poderia contribuir para a deterioração do sistema.

Após analisar neste capítulo os princípios de segurança, é possível identificar qual a taxionomia pela qual um modelo de segurança identifica o seu sistema. Foram analisados os princípios de modelos de segurança, passando pelos fundamentos básicos de modelos de segurança até a concepção de soluções para um sistema. No próximo capítulo será introduzido o modelo *Android*, visto que o nosso estudo recai na análise do modelo de segurança será portanto necessário a compreensão do sistema antes do estudo do seu modelo.

4. Sistema Operativo Android

Antes de focar o estudo do modelo de segurança é necessário compreender o sistema operativo em questão. Como tal, neste capítulo será identificada a constituição do sistema *Android*, como foi definida a sua arquitectura, a compreensão e utilização do manifesto, o seu modelo de permissões e um caso prático de como as aplicações são executadas no seu ambiente.

4.1. Enquadramento

O *Android* é um sistema operativo desenvolvido pela *Google*, baseado em uma versão alterada do *kernel* do *Linux*. Inicialmente desenvolvido por uma empresa chamada *Android Inc* passado mais tarde para a OHA (*Open Handset Alliance*, 2010). De referir também o facto que o sistema operativo *Google* é distribuído em código livre (*open source*), visto que usa a licença da *Apache*.

Neste momento, segundo a Google, as aplicações desenvolvidas já ultrapassaram 10,000 disponíveis. Este facto deve-se principalmente a uma grande comunidade de programadores de aplicações para dispositivos móveis. Visto que é um mercado emergente no qual a maior parte dos interessados pode desenvolver aplicações e mais tarde disponibilizar na internet através do *Market* (local oficial das aplicações *Android*).

4.2. Arquitectura Android

Neste tópico, será identificada a arquitectura do *Android*, na qual podemos ver como o sistema operativo se organiza, realiza as suas operações e como subdivide a sua plataforma. Podemos ainda referir, que este sistema tem a capacidade de correr multi-processos, na qual cada uma das suas aplicações executa o seu próprio processo, incluindo aplicações nativas.

Segundo (B. Speckmann, 2008) e a *Google* (2010), a arquitectura da plataforma *Android* é dividida em cinco conjuntos de estruturas: as aplicações, o quadro de aplicações, as bibliotecas, o *Android runtime* e *kernel* do *Linux.*

4.2.1. Aplicações

Neste grupo são identificadas as aplicações nativas do sistema *Android*, como é possível identificar na figura seguinte.

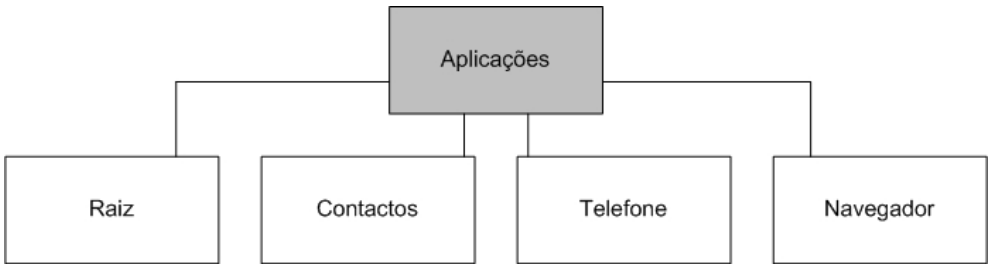

Ilustração 2 - Aplicações nativas da arquitectura Android

O *Android* contem um conjunto de aplicações centrais como um cliente de e-mail, programa para SMS (Short Message Service), calendário, mapas, navegador e gestor de contactos. Todas elas foram desenvolvidas com a linguagem Java e disponibilizam uma série de API desenvolvidas previamente pela *Google*.

4.2.2. Quadro de Aplicações

O quadro de aplicações contém todas os componentes que podem ser usados internamente no desenvolvimento de aplicações. A estrutura que se segue identifica como o quadro de aplicações é constituído.

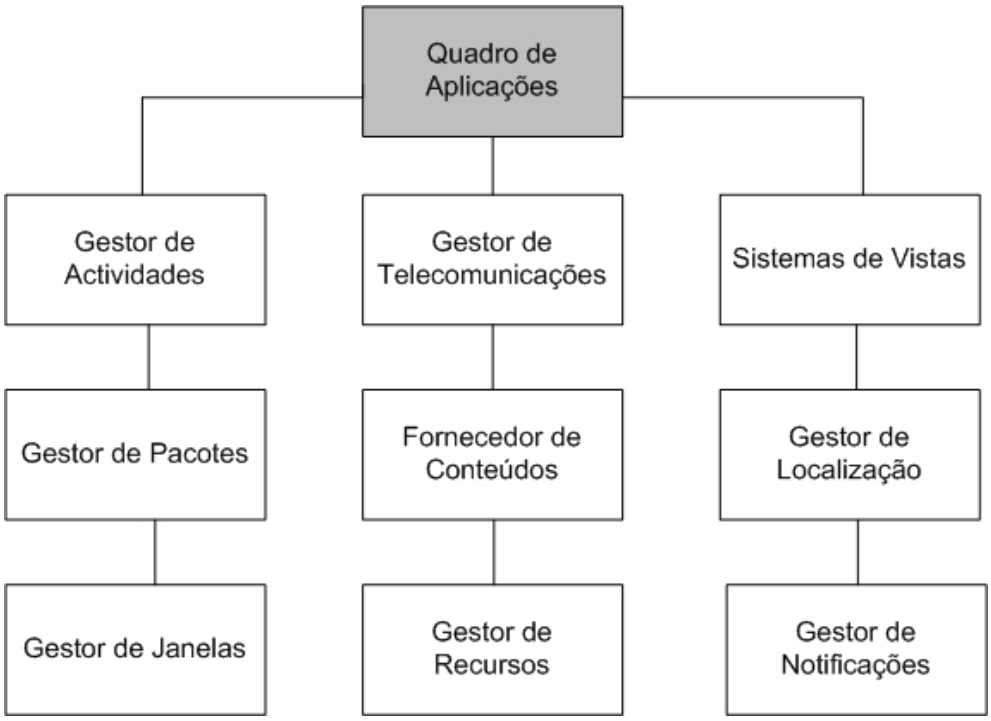

Ilustração 3 - Quadro de aplicações da arquitectura Android

A arquitectura do quadro de aplicações foi desenvolvida de modo a simplificar a reutilização dos componentes. Desta forma, qualquer programador pode construir uma aplicação e permitir que elas sejam utilizadas por outros programas nativos e vice-versa. É de referir que o programador tem acesso total ao sistema assim como toda a estrutura de APIs usadas em aplicações centrais, podendo, desta forma, aproveitá-las conforme achar mais conveniente. É de salientar que todas as aplicações são um conjunto de sistemas e serviços que incluem:

- Sistemas de Vistas (*Views*), painéis de navegação já desenvolvidos, que podem ser usados para construir aplicações, incluindo listas, grelham, botões, e ainda um navegador.
- Aceder a conteúdos de outras aplicações como por exemplo os contactos, através de Fornecedores de Conteúdos (*Content Providers)*
- Um Gestor de Recursos (*Resource Manager)*, que permite aceder a conteúdos como textos ou ficheiros de *Layout*
- Um Gestor de Eventos, que permite alterar mensagens de alerta
- Um Gestor de Actividades, que permite controlar e gerir o ciclo de vida de uma aplicação

4.2.3. Bibliotecas

As bibliotecas do sistema *Android* são um conjunto de funcionalidades disponibilizadas para executar as mais diversas aplicações. É possível identifica-las na figura seguinte:

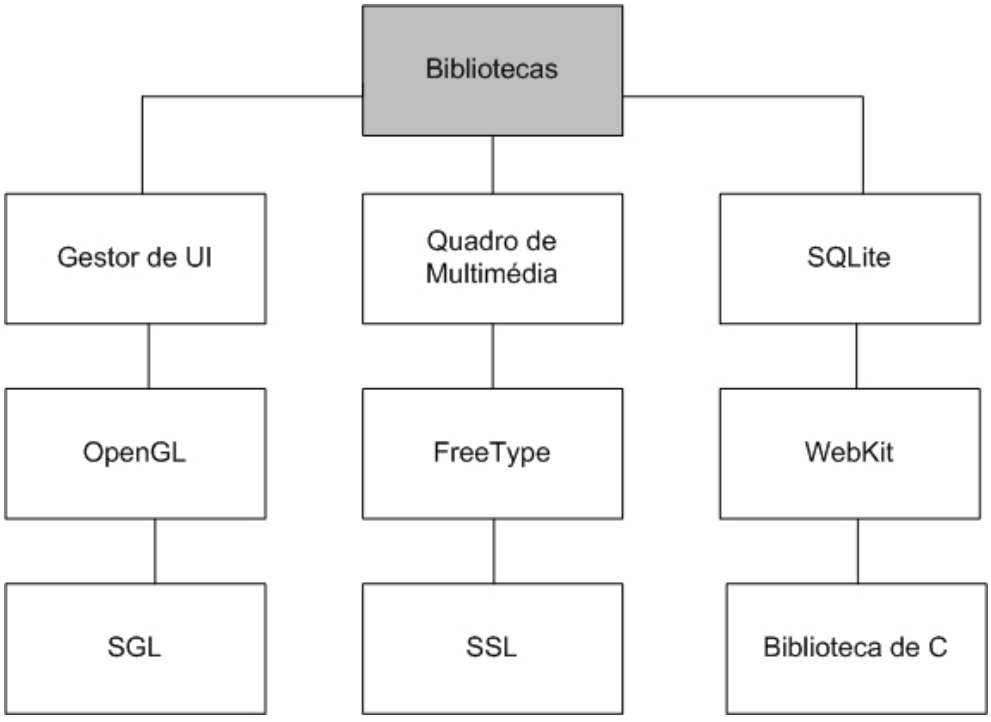

Ilustração 4 - Bibliotecas existentes na arquitectura Android

O sistema inclui um conjunto de bibliotecas criadas em C/C++ e usadas por diversos componentes do *Android*, algumas das funcionalidades estão aqui identificadas:

- O quadro de multimédia suporta a reprodução e gravação de formatos de áudio e vídeo, incluindo MPEG4, H.264, MP3, AAC, AMR, JPG, e PNG
- Gestor de superfície gere o acesso ao *display* de muitos subsistemas desde 2D a 3D
- *Webkit* é um navegador embebido que facilita a comunicação do *Android* com o sistema de vistas implementado
- SGL representa o motor 2D utilizado
- Bibliotecas 3D são usadas tanto com aceleração por *hardware*, se disponível, como também por software
- FreeType serve para a construção de *bitmaps* e fontes vectoriais.
- SQLite é uma base de dados relacional de pequeno tamanho disponível a todas as aplicações

4.2.4. Android Runtime

Durante a execução de aplicações o sistema operativo *Android*, necessita de dois grupos fundamentais, as suas bibliotecas centrais e o Dalvik (a sua maquina virtual), como é possível verificar no quadro em baixo.

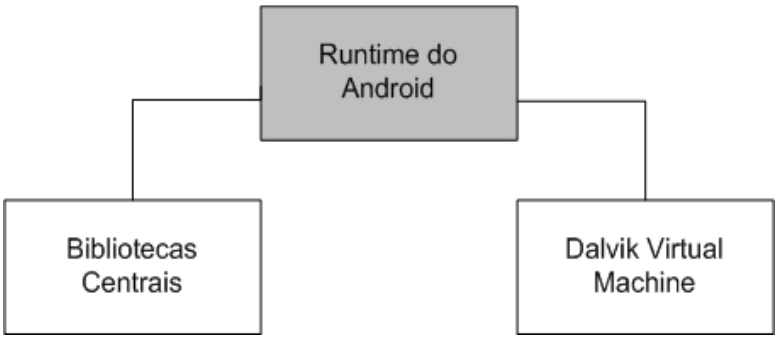

Ilustração 5 - Execução na arquitectura do Android

Cada aplicação do *Android* corre o seu próprio processo o qual é uma instância da máquina virtual *Dalvik* a qual foi criada para que cada dispositivo possa correr múltiplas VM (*Virtual Machines*).

A VM do *Dalvik* executa ficheiros em modo executável *Dalvik* (.dex), formato que foi optimizado para o utilização mínima de memória. E ao contrário de outras VM o *Dalvik* usa uma arquitectura *register-based*, o que faz com que seja mais rápido do que as tradicionais *stack-based* VM, sendo as principais diferenças:

- A VM do *Android* foi aperfeiçoada de modo a usar menos espaço
- Usa o seu próprio *bytecode* e não o Java *bytecode*

Assim sendo, o *Dalvik* corre classes compiladas por uma linguagem Java compilada mas que foi transformada no formato .dex, usando ainda o *kernel* do *Linux* para aumentar as suas funcionalidades como o uso de *threads* e gestão de memória de baixo nível.

4.2.5. Kernel do Linux

No *kernel* do sistema *Android* encontram-se todos os drivers necessários para que o sistema possa tirar partido de todas as funcionalidades inerentes do dispositivo móvel. A figura seguinte retrata os diferentes *drivers* presentes no sistema.

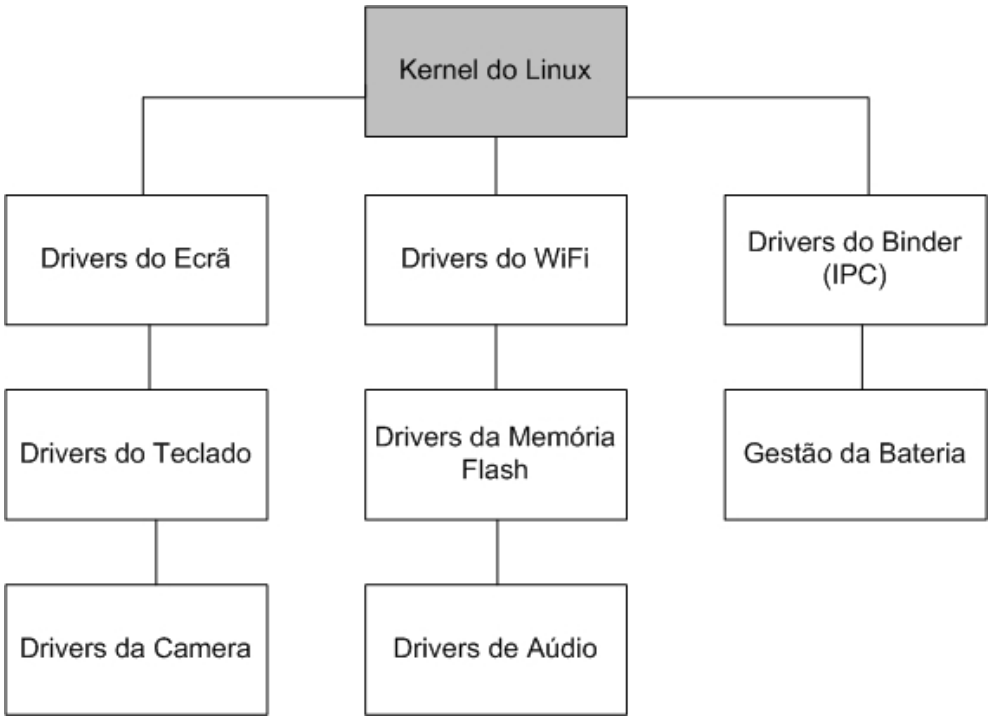

Ilustração 6 - Kernel do Linux na arquitectura Android

O *Android* usa o *Linux* na versão 2.6 para serviços essenciais do sistema, como segurança, gestão de memória, gestão de processos, utilização de rede e drivers. O *kernel* do *Linux* também funciona como uma camada de abstracção entre o *hardware* do dispositivo e o resto do conjunto de *software* que são desenvolvidos em paralelo.

4.3. Manisfesto

O manifesto é um ficheiro XML (*eXtensible Markup Language*), intitulado de *AndroidManifest.xml*, o qual existe obrigatoriamente na raiz de todas as aplicações. Neste ficheiro é apresentada toda a informação essencial sobre a aplicação necessária para o sistema *Android* saber antes de executar o seu código. Entre outros, o manifesto tem como principais funções:

- Dar o nome do *package* de Java para a aplicação (este nome serve como identificador único para a aplicação)
- Descrever os componentes da aplicação, sejam eles actividades (*activities*), serviços (*services*), alertas e eventos da rede (*Broadcast Receivers*) e conteúdos (*Content Providers*)
- Identificar as classes que vão implementar cada um dos componentes
- Determinar quais os processos que são necessários para os componentes da aplicação
- Contabilizar o número mínimo de APIs que o *Android* vai necessitar
- Listar as bibliotecas necessárias para a aplicação
- Declarar quais as permissões que a aplicação pode ter, de modo a aceder a partes protegidas das APIs e interagir com outras aplicações
- Declarar as permissões que outras aplicações têm de modo a interagirem com os componentes

Em baixo, mostra-se um exemplo de um manifesto do *Android*:

```
<application android:icon="@drawable/icon" android:label="@string/app_name">
  <activity android:name=".GreenHat" android:label="@string/app_name">
    <intent-filter>
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
  </activity>
</application>

<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
```

Após analisar o exemplo anterior de um manifesto *Android*, verificamos que, sempre que estiver definido user-permission o sistema vai dar permissão ao acesso requisitado. Neste caso concreto, o intuito é disponibilizar o acesso a internet assim como verificar o estado da ligação sem fios (wifi).

4.4. Permissões

Relativamente ao campo das permissões, verificamos que aqui as restrições do código e dos dados do dispositivo são efectuadas. A limitação é imposta de modo a proteger dados críticos e código que podia ser indevidamente usado ou danificado durante a sua utilização. Deste modo, cada permissão é identificada com uma etiqueta única, a qual identifica o tipo de acção que está a ser permitido, vejamos o exemplo:

```
android.permission.CALL_EMERGENCY_NUMBERS
android.permission.READ_OWNER_DATA
android.permission.SET_WALLPAPER
android.permission.DEVICE_POWER
```

Pode ainda existir a possibilidade de aceder a uma aplicação protegida declarando uma permissão com <uses-permission> no manifesto, assim, quando a aplicação é instalada no dispositivo o instalador determina se deve ou não dar acesso a outra aplicação. Neste caso pergunta ao utilizador se quer dar permissão, caso seja permitido, então aí poderá aceder a dados previamente protegidos, caso contrário simplesmente falha nem chegando a notificar o utilizador. Segue-se uma ilustração de um ecrã que identifica ao utilizador as permissões de uma aplicação.

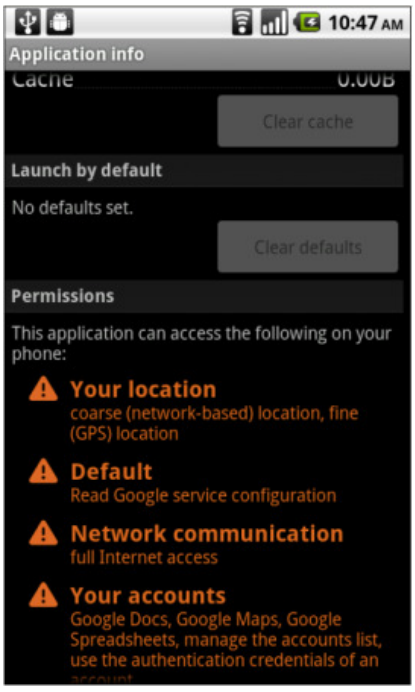

Ilustração 7 - Exemplo de sistema Android com a listagem de permissões de uma aplicação

4.4.1. Níveis de permissões

Com mais de cem tipos de permissões que controlam os acessos das aplicações foram criados diferentes níveis de permissões para melhor as caracterizar:

- **Normais**

  Permissões a nível da aplicação que não correm risco ao serem executadas, como é o caso de um jogo que precisa de acesso a raiz do cartão de memória.

- **Perigosas**

  Permissões de alto risco que permitem acesso a dados privados ou funcionalidades que podem alterar o sistema, tais permissões não são permitidas sem o consentimento do utilizador

- **Assinatura**

  Permissão que permite aceder a outros pacotes assinados com a mesma assinatura que aquele que esta a ser invocada

- **Assinatura ou sistema**

  Um tipo especial de permissão de assinatura que permite o acesso a pacotes instalados dentro do sistema

Com a divisão destas permissões nestes quatro grupos torna-se o modelo de segurança mais estruturado relativamente a níveis de permissões. Com o factor acrescentado de utilizar um dos princípios de modelos de segurança que afirma que se deve manter um nível de aceitação psicológica aceitável para o utilizador.

4.5. Ciclo de vida de uma aplicação

Antes de perceber como é que uma aplicação é executada no sistema *Android*, vejamos quais são os fundamentos de uma aplicação.

Todas as aplicações são escritas com a linguagem de programação Java, depois de compiladas, juntamente com todos os dados necessários serão transformados em um ficheiro designado como *apk (Android Package).* Este formato vai ser posteriormente distribuído como uma aplicação e instalado nos aparelhos móveis. Vejamos alguns pontos fulcrais da sua execução:

- Cada aplicação corre o seu processo Linux; o *Android* inicia o processo quando o código da aplicação precisa de ser executado e desliga-o quando não é mais necessário.

- Por defeito, cada processo tem a sua própria máquina virtual (VM) e como tal o código da aplicação corre isolada do código de outras aplicações.
- Cada aplicação tem um identificador único do *Linux user ID*. As permissões estão parametrizadas para que apenas esse utilizador possa ver os ficheiros da aplicação assim como a aplicação em si, embora existam maneiras de as fazer passar para outras aplicações desde que ambas aplicações usem o mesmo ID do utilizador.

Em seguida é analisado um exemplo de um caso prático de aplicações a serem executadas no ambiente *Android*.

4.5.1. Caso prático

No exemplo seguinte vamos compreender como o processo do ciclo de vida de aplicações decorre, para obtermos uma ideia de como o sistema se comporta. Assim sendo, temos a seguinte situação: um utilizador encontra-se ao telefone com uma pessoa, o qual lhe pede para aceder a uma página na internet, encontrar uma fotografia, enviar-lha e por fim termina a chamada. Na figura seguinte é possível verificar a ilustração do exemplo descrito.

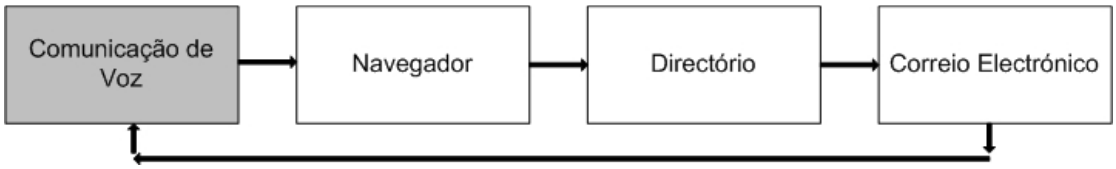

Ilustração 8 - Ciclo de vida de chamada com aplicações a decorrerem

Neste contexto verificamos que existem quatro aplicações diferentes e quatro processos diferentes a correr, mas do ponto de vista do utilizador nenhum deles é mais importante que o outro, visto que é o *Android* que vai gerir todo o esforço do processamento, assim como, a memória usada. Como estes processos são transparentes ao utilizador, não se tem de preocupar com os processos que estão a decorrer ou quanta memória esta a ser usada.

Ao analisar a situação com mais detalhe verificamos que inicialmente o utilizador está em comunicação com outro utilizador através de uma conversa telefónica, desse modo a aplicação para a conversa é inicializada, a qual contem o seu próprio gestor da actividade. Na imagem seguinte, vemos como a pilha contem dois processos a serem executados, o processo que gere o sistema (*System Process*) e a aplicação de comunicação de voz. Antes da pilha de programas carregar a aplicação do navegador, ela vai guardar um estado V relativo a voz para ser possível se lembrar mais tarde do processo.

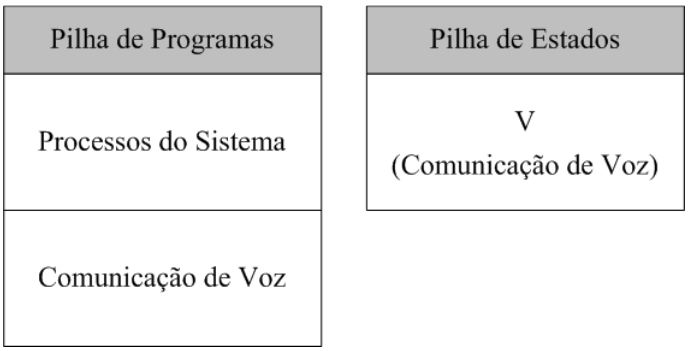

Ilustração 9 - Pilhas no estado inicial

Ao chegar a este ponto, o utilizador coloca a comunicação de voz em "espera" e abre o navegador, o sistema cria então um novo processo, e uma nova actividade é criada para esse evento. De novo, o estado da última actividade é guardado (N):

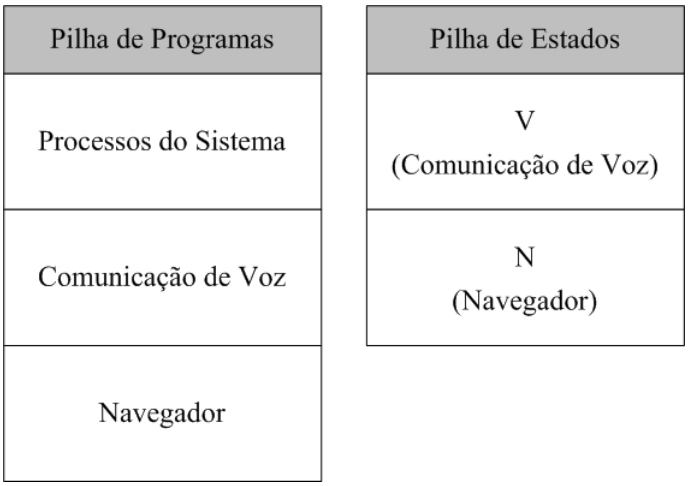

Ilustração 10 - Pilhas com a aplicação de navegador a funcionar

Após estas acções, o utilizador usando o navegador, acede à página pretendida na internet para adquirir a imagem que pretende e em seguida guarda-a para um directório específico. No entanto, ele não chega a encerrar o navegador, e em vez disso abre o directório onde guardou a imagem. A actividade para este processo particular é iniciada (D):

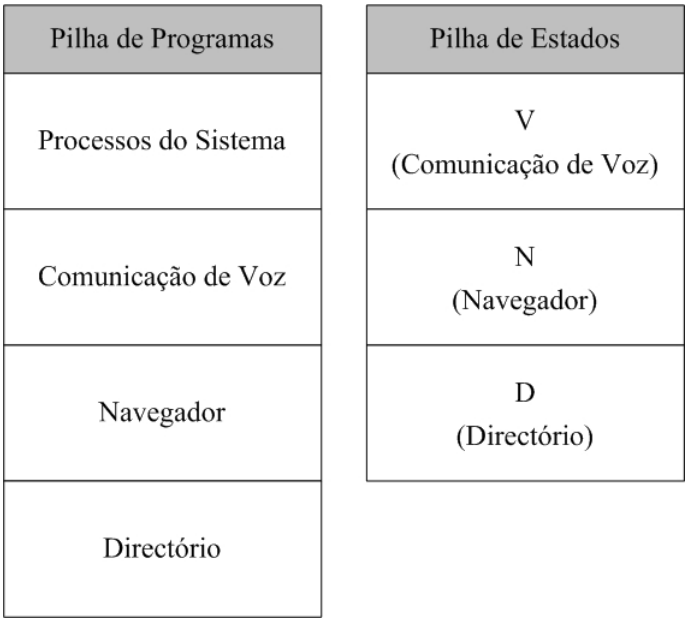

Ilustração 11 - Pilhas com a aplicação de directório adicionada

Neste ponto, o utilizador já tem a imagem pretendida e pode abrir a aplicação de correio electrónico. O último estado (D) é guardado.

No mesmo momento, aloca o processo do correio electrónico simultaneamente (C) na pilha de programas como na pilha de estados e remove o navegador:

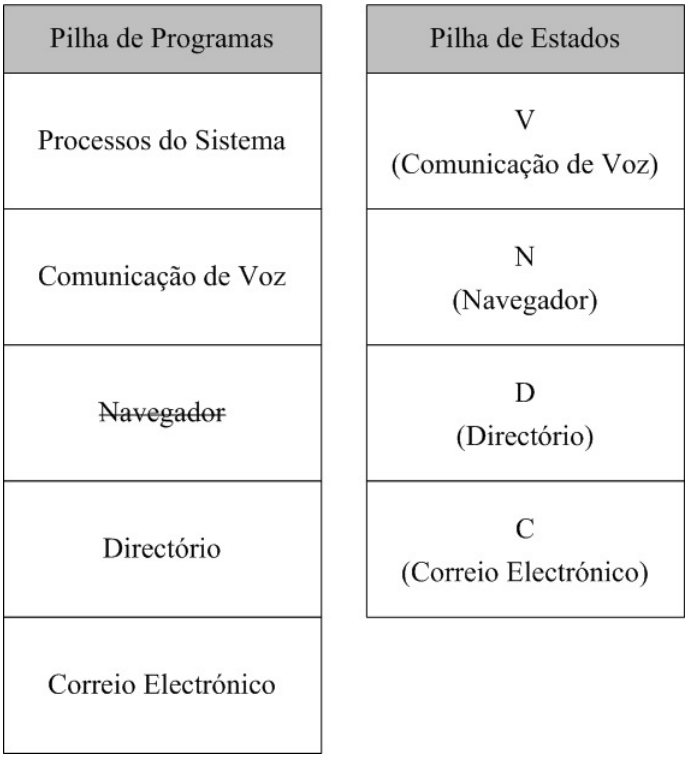

Ilustração 12 - Pilhas com a aplicação de correio electrónico adicionada e navegador removido

Imaginemos que nesta fase que o *smartphone* não tem mais memória. Nesse caso, o procedimento lógico do sistema *Android* será o de destruir um processo, mas a questão é qual deles? Visto que o sistema não pode destruir o processo do directório, pois é a última referência na pilha de estados, o sistema ira optar por descontinuar o processo do navegador.

Como é possível identificar, na pilha de programas o navegador foi eliminado, mas a sua referência na pilha de estados continua presente. Em seguida, o utilizador abre o correio electrónico e envia a imagem e para cada uma das pilhas é carregada a aplicação de correio electrónico. A acção lógica do utilizador após será de voltar a conversa inicial e retomar a chamada.

Pilha de Programas	Pilha de Estados
Processos do Sistema	V (Comunicação de Voz)
Comunicação de Voz	N (Navegador)
Directório	D (Directório)
~~Correio Electrónico~~	~~C~~ ~~(Correio Electrónico)~~

Ilustração 13 - Pilhas com a aplicação de correio electrónica em remoção

Com a acção do correio electrónico terminada, será feita a remoção da pilha de programas e de estados. Durante o seu retrocesso, o utilizador passa por todos os estados que estiverem presentes na pilha de estados. Sempre que voltar atrás elimina a aplicação que estava a usar previamente, removendo-a de ambas as pilhas assim como as suas referências. No entanto, quando a pilha de estados atinge o estado do navegador é carregado novamente a aplicação na pilha de programas, como identificado no esquema seguinte:

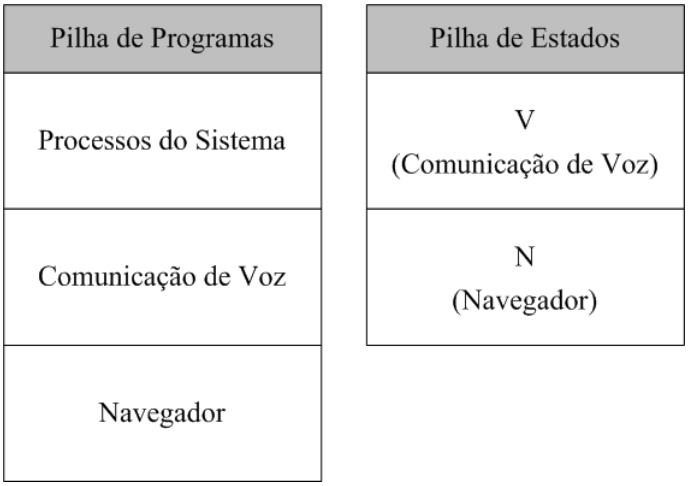

Ilustração 14 - Pilhas voltam ao estado inicial com navegador em funcionamento

Para finalizar, o utilizador volta à aplicação de conversa e resume a sua conversa com o outro utilizador. Devido a terem sido guardados os estados das aplicações previamente utilizadas é relativamente simples voltar atrás assim como proveitoso, pois ele não só recorda quais as actividades previamente criadas, como também as vistas usadas. Neste exemplo, foi possível verificar que sistema Android consegue fazer a gestão da memória das suas aplicações assim como consegue voltar atrás devido a guardar as referências na pilha de estados.

Concluído a analise da arquitectura do sistema *Android*, é neste momento possível identificar como os seus componentes estão divididos e para que são utilizados. Foi compreendido como o manifesto e o mecanismo de permissões funcionam, assim como as suas utilidades. Interpretou-se o funcionamento do sistema perante um caso de teste e o funcionamento internos do sistema.

Neste momento, é necessário compreender como é que o modelos de segurança funcionam em conjugação dos sistemas operativos, como tal, o próximo capítulo identificará os modelos de segurança dos sistemas operativos *Android*, *iPhone* e *Symbian* funcionam.

5. Modelos de Segurança dos Smartphones

O objectivo deste capítulo é identificar e explicar como é implementado o modelo de segurança nos sistemas operativos do *Android*, *iPhone* e *Symbian*. A selecção destes modelos deve-se ao facto de serem três dos sistemas com mais impacto no mercado nacional de dispositivos móveis. O *Symbian* é o sistema operativo com o maior número de dispositivos vendidos. O *iPhone* já conquistou sua marca no mercado e o sistema operativo adoptado tem uma boa aceitação devido a sua interface de fácil utilização e o *Android*, embora ainda recente, já contem uma sólida base de utilizadores.

5.1. Modelo de segurança do Android

Relativamente ao modelo de segurança do *Android*, podemos identificar que a constituição da sua arquitectura tem mecanismos do sistema de segurança do Linux, como é o caso do ID do utilizador (*user* ID) e do ID de grupos (*group* ID). Contém mecanismos especificamente desenvolvidos para o seu ambiente e utiliza também um mecanismo de validação de referência. É de salientar, como visto anteriormente, que também reforça as restrições do acesso às aplicação através do sistema de permissões.

5.1.1. Arquitectura de Segurança

O ponto fulcral da arquitectura de segurança do *Android* baseia-se que, por defeito, nenhuma aplicação tem permissão de executar qualquer operação que possa entrar em conflito com outra aplicação, o sistema operativo ou o utilizador. Como tal, isto refere-se à possibilidade de ler ou escrever nos dados pessoais do utilizador, aceder a ficheiros de outras aplicações e a configurações de rede. É possível estruturar em três pontos principais o funcionamento da sua arquitectura: *sandbox*, assinatura da aplicação, identificadores dos utilizadores e acesso a ficheiros.

Sandbox

Cada processo de uma aplicação é executado numa *sandbox.* De modo a compreender o conceito de *sandbox,* é necessário imaginar uma "caixa" na qual se pode utilizar o que estiver lá dentro mas não se pode utilizar nada que esteja no seu exterior, protegendo assim todo o ambiente que envolve a "caixa". Com o uso de uma *sandbox* é possível evitar aplicações de se poderem danificar ou acederem a ficheiros de outras, a menos que exista uma declaração (*permission*) no ficheiro do manifesto (*AndroidManifest*) da aplicação.

O modelo de segurança aceita pedidos de acessos das mais variadas maneiras, mas antes verifica se existe permissão e caso não haja pode questionar o utilizador. Deste modo a aplicação quando é instalada pela primeira vez notifica o utilizador dos acessos que ela vai poder efectuar, os quais nunca mais vão poder ser alterados.

Assinatura da Aplicação

Todas as aplicações no *Android* têm de ser assinadas com uma assinatura digital, a qual é associada através de uma chave privada que é guardada pelo programador, identificando deste modo o autor da aplicação. Deste modo, serve-se o propósito principal de criar uma base de confiança entre aplicações, pois existe uma assinatura em todas as aplicações que identificam o seu criador.

Identificadores dos Utilizadores e Acesso a Ficheiros

Em cada instalação de uma aplicação, no dispositivo, é associada um identificador único de utilizador do Linux, UID (user ID*)*. A partir deste momento, sempre que for executada será criado uma *sandbox* específica para a aplicação com aquele identificador único. Deste modo, impede-a de interagir com outras aplicações, da mesma maneira que evita interacção de outras aplicações. Este UID que lhe foi associado fica com esta aplicação até ao momento da sua desinstalação no dispositivo.

Visto que o reforço de segurança acontece ao nível do processo, o código de duas aplicações distintas não pode ser executado normalmente no mesmo processo pois contém UID diferentes. Para tal acontecer, é necessário usarem o *sharedUserId* no manifesto do *Android* (*AndroidManifest*) para que cada aplicação consiga usar o mesmo UID. Se tal acontecer, ambas aplicações vão ser tratadas como sendo a mesma, com o mesmo UID e como tal assim como as mesmas permissões.

No entanto segundo Google Developers (2010), para não se tornar vulnerável, apenas duas, ou mais, aplicações com a mesma assinatura (e com o pedido do mesmo *sharedUserId*) é que vão ser fornecidas com o mesmo UID. Qualquer tipo de dados guardados por uma aplicação vai ser associado e identificado com o UID dessa mesma aplicação, e normalmente não lhe deverá ser fornecido o acesso a outras aplicações.

5.1.2. Mecanismos de segurança

Os autores (Shabtai et al. 2009) propuseram que o modelo de segurança do *Android* tinha de incorporar mecanismos de segurança para prevenir eventuais vulnerabilidades e dividiram-nos em três grupos distintos: mecanismos do *Linux*, recursos do ambiente e mecanismos específicos do *Android.*

Mecanismos do Linux

Os mecanismos existentes na camada do Linux são divididos em duas partes: o POSIX (*Portable Operating System Interface*) e o acesso a ficheiros. O elemento base é o utilizador (a entidade, "*user*") e cada objecto (processo ou ficheiro) pertence a um utilizador (o qual é representado pelo UID)

- **POSIX**

  Cada ficheiro de pacotes (package .apk) é instalado com um único identificador Linux UID como posteriormente tínhamos visto, de modo a que não seja possível correr dois processos com o mesmo ID, a menos que existam permissões para duas aplicações partilharem o mesmo ID (utilização do sharedUserID).

- **Acesso a ficheiros**

  Como cada ficheiro está associado com o user ID e com o group ID assim como a tupla de permissões de leitura, escrita e execução (rwx). O facto de existir um identificador único para cada aplicação aumenta o nível de segurança ao acesso a ficheiros, tanto aos que pertencem ao sistema operativo como aos da própria aplicação. Os ficheiros de uma determinada aplicação não vão estar disponíveis a outra aplicação a menos que usem sharedUserID no código, ou estejam definidas globalmente as propriedades de leitura e de escrita desse mesmo programa.

Recursos do Ambiente

Os recursos do ambiente que estão disponíveis ao *Android* (por exemplo, hardware, infra-estrutura da operadora móvel) são também capazes de fornecer mecanismos de segurança. Ou seja, independentemente do dispositivo móvel ao qual o modelo de segurança *Android* esteja incorporado, existem mecanismos que estão presentes e poderão ser utilizados.

- **Memory management unit**

  É considerado um pré-requisito para muitos sistemas operativos, como é o caso do *Linux*.

  A unidade de gestão de memória, MMU (*Memory Management Unit*), um componente de hardware que facilita a separação de processos em diferentes espaços de memória virtual. Deste modo, um processo não tem a capacidade de "ler" a memoria de outra aplicação, ou seja, existe isolamento de informação.

- **Type Safety**

  Trata-se de uma propriedade de linguagem de programação que força as variáveis a serem atribuídas a determinados tipos e formatos, o qual acontece com a linguagem Java mas não é obrigatório em linguagens como o C, prevenindo erros e utilizações não desejadas. A falta deste tipo de linguagem pode levar a corrupção de memória, ataques de *buffer-overflow*. E mesmo usando mecanismos de validação de referência, a comunicação entre as aplicações torna-se mais segura.

- **Segurança da operadora móvel**

  Os atributos base de segurança de uma operadora móvel recaem no pressuposto do AAA (*authentication, authorization, e accounting*) e o *Android* usa a autenticação através do uso do cartão SIM.

- **Mecanismos específicos do Android**

  Os mecanismos específicos apresentados pelo *Android*, recaem sobre o encapsulamente de componentes e assinaturas de aplicações.

- **Encapsulação de Componentes**

  Uma aplicação do *Android* pode encapsular os seus componentes dentro dos conteúdos da sua aplicação, prevenindo-se contra o acesso de outras aplicações que tenham um UID diferente do seu.

- **Assinatura de aplicações**

  Cada aplicação no *Android* é arquivada num pacote (apk) para a instalação e o sistema operativo obriga que cada um dos ficheiros a instalar estejam assinados digitalmente, independentemente de serem recursos com código executável ou não.

  O pacote assinado é válido enquanto o seu certificado também o for e enquanto a chave pública que o programador utilizou para assinar a aplicação se mantiver identica. Só assim é possível verificar correctamente se a aplicação em questão é do mesmo autor (*"same-origin" verification*).

Estes mecanismos podem ser apresentados por tópicos como é possível ver no seguinte quadro:

	Mecanismo	Descrição	Factor de Segurança
Mecanismos do Linux	POSIX	Cada aplicação é associada com um UID (*user* ID) diferente	Previne que uma aplicação aceda a outra
	Acesso a Ficheiros	O directório da aplicação só é disponibilizado a aplicação	Previne que uma aplicação aceda a ficheiros de outra
Recursos do Ambiente	Memory Management Unit	Cada processo é executado no seu próprio espaço de memória	Prevenção de escalonamento de privilégios, negação de serviço
	Type Safety	Procedimento que reforça conteúdo de variáveis para um determinado formato	Prevenção contra buffer overflows
	Segurança da Operadora Móvel	Uso de cartões SIM para autorizar e autenticar a identidade do utilizador	Prevenção contra roubo do dispositivo
Mecanismos Específicos do Android	Encapsulamento de componentes	Cada componente na aplicação pode ofuscar o seu acesso através de outra aplicação	Prevenção contra acesso não autorizado entre aplicações ou componentes
	Assinatura de aplicações	O programador assina as suas aplicações e o gestor de pacotes verifica a autenticidade	Verifica que duas ou mais aplicações vêm da mesma fonte

Tabela 3 - Esquema dos mecanismos de segurança do Android

Visto os mecanismos de segurança usados pelo sistema Android na sua arquitectura foi possível identificar para que tipo de ameaças é que o sistema se preparou e qual a solução apresentada.

Será agora identificado o funcionamento do seu monitor de referências.

5.1.3.Inter-component Communication

Um dos reforços de segurança existentes no *Android* é através do ICC (*Inter-component communication*) o qual suporta uma mediação no nível mais interno da *framework* de segurança do sistema operativo. Como a política de segurança do Android é obrigatória, todas os rótulos de permissões que são definidos durante a instalação não podem ser alterados até que a aplicação seja desinstalada. Segundo (Patrick McDaniel et al. 2009), este controlo ocorre via I/O num nodo especial, em /dev/binder, e garante o reforço da segurança na camada de *middleware*. Pois observamos que é o ICC o responsável por interpretar os rótulos (*labels*) que foram associados às aplicações, como é verificado na figura seguinte.

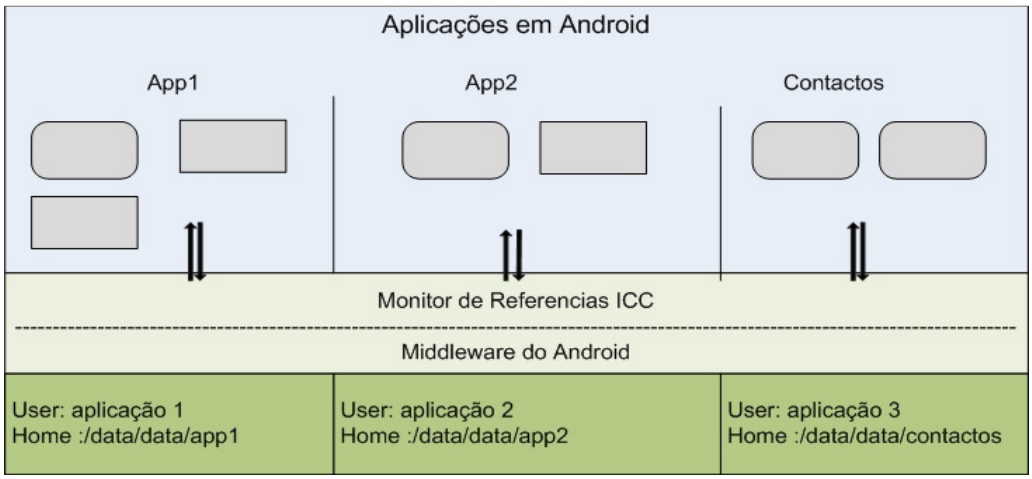

Ilustração 15 - Funcionamento do Inter-component Communication

Deste modo, um monitor de referência é criado para obter um controlo de acessos necessário na troca de dados, MAC (*Monitor Access Control*). Assim é possível verificar se cada componente de uma determinada aplicação pode ser utilizado por outra aplicação.

No desenvolvimento de aplicações, é necessário fazer associações de rótulos de permissões caso exista a necessidade de partilhar dados entre aplicações. Deste modo, quando o ICC se iniciar é possível verificar se tem permissão para aceder a determinadas propriedades ou dados de aplicações. Caso contrário, não é estabelecida ligação mesmo que o componente esteja na mesma aplicação, como é visto na seguinte figura.

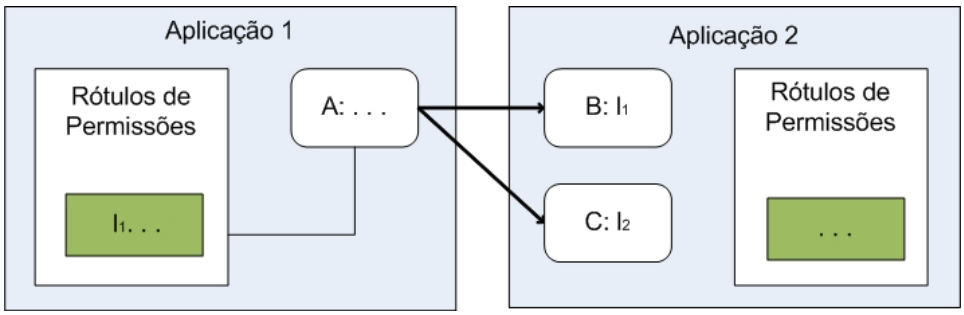

Ilustração 16 - Exemplo de partilha de permissões entre aplicações

A aplicação 1 consegue aceder à aplicação 2 mas apenas tem permissão para utilizar o componente B, visto que é isto que está listado nas permissões.

Após esta análise, verificamos que no desenvolvimento de aplicações é necessário associar permissões com o *AndroidManifest* pois assim consegue-se definir qual a política de segurança que se pretende implementar na aplicação. Se é uma política de segurança ao domínio ou se apenas se restringe a um determinado componente.

Como referido anteriormente, a política de segurança do *Android* é obrigatória, como tal todas os rótulos de permissões que foram definidos na instalação não podem ser alterados até que a aplicação seja eliminada.

5.2. Modelo de Segurança do iPhone

O *iPhone*, na sua versão 3GS, aplica uma visão mais empresarial usando alguns pressupostos essenciais no seu modelo de segurança, de modo a construir acessos seguros a serviços e protecção de dados do dispositivo. O *iPhone* usa uma versão do sistema operativo do Mac OS X, normalmente utilizada em computadores da *Apple*, sendo que a maior diferença recai no uso de um processador ARM (*Advanced RISC Machine*) em vez do x86.

Segundo a (*Apple Security Overview,* 2010) e (*IOS Reference Library,* 2010) o modelo concentra-se nos seguintes pontos fulcrais:

- Métodos que previnem o acesso indevido ao dispositivo.
- Protecção de dados DARP (*Data at rest protection*), incluindo a situação de perda ou roubo do aparelho.
- Protocolos de segurança de rede e utilização de cifras para a transmissão de dados.
- Consistência da plataforma do IOS (*iPhone Operative System*).

5.2.1. Controlo de dispositivo e protecção

No *iPhone* o controlo e protecção é feito através de configurações de senhas de acesso, políticas de acesso e uma configuração do dipositivo.

Configuração de senhas

O uso de senhas e a sua configuração é a primeira barreira imposta pelo *iPhone*. Com especial atenção à personalização das senhas, a qual permite o utilizador escolher um extensivo leque de opções para impedir o acesso a dados guardados no telemóvel. Dentro dessas opções estão incluídos definir períodos de tempo (*timeouts*), a exigência da senha (número máximo ou tipo de caracteres) e frequência da alteração da senha.

Politicas de Acesso

As políticas de acesso do *iPhone* recaem no uso perfis, os quais podem ser definidos previamente com diferentes níveis de impacto no sistema. Por exemplo, apagar uma conta de correio electrónico só poderá ser feita através de uma senha de administrador (caso o perfil em questão esteja previamente definido). É ainda possível associar este perfil ao dispositivo, não permitindo outro utilizador apagar a conta sem que remova todos os dados do aparelho.

Configuração Segura do Dispositivo

A configuração dos perfis é criado em ficheiros XML, os quais contêm as políticas de segurança, as restrições, configuração de VPN (*Virtual Private Network*), configurações de rede, correio electrónico, calendário e credenciais de autenticação para permitir que o *iPhone* trabalhe com os sistemas de uma empresa. A configuração do dispositivo, com um determinado perfil, pode ser uma funcionalidade que permita à empresa implementar as suas normas de segurança no telemóvel. Os perfis de configuração podem ser cifrados, usando CMS (*Cryptographic Message Syntax,* RFC 3852), suportando 3DES (*Triple Data Encryption Standard*) e AES128 (*Advanced Encryption Standard*).

Restrições do Dispositivo

Embora a restrição do dispositivo seja mais usada por empresas, ou no caso de haver mais do que um utilizador, é uma funcionalidade implementada no modelo de segurança do *iPhone*. Através das restrições é possível determinar quais as aplicações que os utilizadores podem aceder no *iPhone,* sejam elas o *Safari, YouTube, iTunes* entre outras.

Execução de Aplicações

Não é permitindo o acesso de execução de aplicações de terceiros que não tenham sido autenticadas pela *Apple* e introduzidas no *iPhone* através do seu *software* oficial, o *iTunes*. Sendo negado inclusive o acesso ao directório de ficheiros através de ligação por USB (*Universal Serial Bus*).

No entanto, segundo (Charlie Miller et al. 2007), todos os processos manipulados que utilizem dados de rede são executados com *user id* de valor 0, o que equivale ao utilizador máximo (*superuser*) e o com o maior número de permissões. O que faz com que qualquer aplicação que seja comprometida neste contexto vai ser executada com o maior nível de privilégio.

5.2.2. Protecção de Dados

De modo a proteger os dados dos seus utilizadores existe a possibilidade de cifrar toda a informação que está guardada no dispositivo através de *software* e por *hardware*, mas, se o mesmo é roubado ou perdido, é importante desactiva-lo ou apagar todos os dados.

Tal funcionalidade pode ser activada se o *iPhone* estiver configurado para tal. Se após um certo número de tentativas a senha de acesso do aparelho (que se encontra bloqueado), os dados podem ser apagados para que este tipo de ameaça não seja concretizado.

Cifrar

Como foi dito posteriormente é possível cifrar a informação baseado em *hardware*, o qual usa o AES 256 *bit,* sendo possível também adicionalmente fazer uma cópia de segurança da informação através do *iTunes*.

Eliminação dos Dados Remotamente

O *iPhone* pode ser apagado remotamente, se existir o caso de se perder ou o telemóvel ser furtado, o administrador ou o proprietário pode executar remotamente esta acção (*remote wipe*) que faz com que todos os seus dados sejam eliminados e que seja desactivado.

Caso o aparelho esteja configurado com uma Exchange *account*, o administrador pode iniciar a eliminação usando a consola do *Exchange Management (Exchange Server 2007)* ou o *Exchange ActiveSync Mobile Administration Web Tool (Exchange Server 2003 or 2007).*

Eliminação dos Dados Localmente

Na eliminação dos dados localmente é possível configurar o dispositivo para apagar os seus dados após a falha da senha de acesso ter passado o seu número máximo de tentativas. Este tipo de solução existe de modo a prevenir ataques de tentativas para ganhar acesso ao *iPhone*, por defeito ele automaticamente apaga todos os dados apôs dez tentativas falhadas. No entanto é possível configurar o número máximo de tentativas no perfil do utilizador.

5.2.3. Comunicações Seguras

Como é prioritário proteger toda essa comunicação que é feita no acesso a empresas, o modelo do *iPhone* permite o uso de canais seguros, assim como a utilização de protocolos de segurança para manter a integridade da comunicação, seja através de *Wi-Fi* ou da rede da operadora móvel.

VPN

Desde que a empresa em questão já tenha este tipo de rede virtual estabelecida, o *iPhone* pode aceder facilmente pois já tem integrado tecnologia VPN suportada pelo Cisco *IPSec* (*Internet Protocol Security*) e L2TP (*Layer 2 Tunneling Protocol*). Adicionalmente é possível autenticação através de VPN *On Demand*, implementado segurança no acesso aos mais variados serviços.

SSL/TLS

O SSL v3 (*Secure Sockets Layer*), assim como TLS v1 (*Transport Layer Security*) são mecanismos utilizados automaticamente por aplicações sempre que é necessário comunicar em canais seguros. Assim, sempre que for necessário uma aplicação como o caso do navegador, correio electrónico, calendário ou qualquer outro necessite utilizar canais cifrados para fazer a sua comunicação, pode usar estes mecanismos.

WPA/WPA2

Através do suporte de WPA2 (*Wi-Fi Protected Access*), e usando cifras de 128-bitAES, é possível existir uma comunicação em ligações sem fios entre o *iPhone* e o serviço que está a ser acedido.

5.2.4. A plataforma iPhone

A plataforma do *iPhone* foi pensada de modo a conter a segurança dentro do seu núcleo, como tal é implementado a utilização do método *sandbox* para as aplicações. Existe também ainda a necessidade de ter uma assinatura associada a cada aplicação de modo a preservar a sua integridade. Permite também que os dados da aplicação e credenciais de rede possam ser guardados com uma chave cifrada.

Protecção em Execução (Runtime)

Visto que é usado o método *sandbox* não é possível aceder a dados de outras aplicações, assim como ficheiros do sistema, recursos, e acesso ao núcleo (*Kernel*) do sistema operativo. Caso seja necessário uma aplicação aceder a dados de outra aplicação pode-o fazer se usar as APIs e serviços disponibilizados pelo sistema.

Obrigatoriedade de Assinatura

Todas as aplicações do *iPhone* têm obrigatoriamente uma assinatura digital, incluindo as aplicações nativas que já tem a assinatura da Apple. As aplicações não nativas têm de ser assinadas pelo programador usando um certificado emitido pela Apple, no qual vai ser garantido que não houve qualquer tipo de manuseamento no código. Adicionalmente, sempre que uma aplicação é executada é feita uma verificação para determinar se houve alteração desde a sua última utilização, tornando-se assim não confiável. Caso exista uma situação destas, a aplicação será terminada de imediato e não será possível executa-la novamente.

Autenticação Segura da Plataforma

Através de uma chave cifrada é possível guardar identidades digitais, nomes de utilizadores e palavras passes de um modo seguro. A chave da cifra será posteriormente guarda no sistema mas não poderá ser acedida por outro utilizador.

Arquitectura de Cifras

Os programadores tem acesso a APIs que podem usar para cifrar e proteger assim os seus dados, usando algoritmos como AES, 3DES. O *iPhone* pode utilizar aceleração de hardware para maximizar a performance da aplicação caso seja necessário o calculo de alguma cifra que necessite maior poder de cálculo.

5.3. Modelo de Segurança da Symbian

O sistema operativo *Symbian*, na sua versão 9.5, é actualmente o mais vendido em todo o mundo relativamente a modelos de *smartphones*, e segundo a *IEEE Spectrum:*

"Globalmente, quase metade dos 175 milhões de smartphones vendidos em 2009 tinham o sistema operativo Symbian a ser executado [...]" o que faz com que este sistema, que começou aproximadamente há 30 anos, seja ainda hoje um dos mais bem sucedidos e com maior cota do mercado.

Como tal, é de salientar que este sistema foi dos que mais evoluí com o seu modelo de segurança, visto que foi sujeito aos mais variados tipos de ataques e exploração de vulnerabilidades. Portanto, foi gradualmente evoluindo o seu modelo até aos dias de hoje, contando assim com as aprendizagens passadas, reutilização de código, e pressupostos baseados em concepções de modelos de segurança de dados como os que foram previamente descritos. Como tal o modelo da *Symbian* assenta nos seguintes pontos: processo de confiança, capacidade de determinar privilégios e encapsulamento de dados.

5.3.1. Processo de Confiança

O sistema operativo da *Symbian*, como a maior parte dos outros sistemas operativos, é concebido para ser operado por um único utilizador, não é necessário utilizar um nome de utilizador e uma palavra-chave para iniciar uma sessão no telemóvel, basta utilizar o PIN (*Personal Identification Number*) para ter acesso ao sistema e as aplicações. A partir deste momento a *Symbian* cria uma base de confiança para o utilizador.

Níveis de Confiança

A arquitectura de segurança da *Symbian* foi concebida de modo a controlar o que um processo pode ou não fazer, ou seja, só é capaz de executar uma actividade(s) caso tenha os privilégios correctos senão não vai ser executada. Pois não tem um nível de confiança suficiente.

Existem dois identificadores associados com cada ficheiro binário executável, o *secure identifier* (SID) e o *vendor identifier* (VID). Os SID são obrigatórios e únicos, podendo ser apenas atribuídos a um ficheiro executável se vierem com uma assinatura de entidade de confiança ao contrário dos VID que não precisam de ser únicos. Se todos os ficheiros executáveis são do mesmo fabricante partilham o mesmo VID.

Um processo tem pelo menos uma thread de execução e tem os seus recursos, particularmente os blocos físicos da memória controlados por um programa de gestão no hardware, o MMU. Caso haja um acesso indevido a um endereço virtual que não esteja no processo, o hardware vai criar um erro de processamento. Este tipo de funcionalidade é onde reside a base do modelo de segurança, pois o sistema operativo pode confiar num processo sabendo que ele não vai aceder a nenhum endereço de memória que não possa, pois o hardware não o permite. Para tal, ele verifica se o SID que tenta aceder é o mesmo que foi combinado com o endereço de memória.

Assim é possível dividir os níveis de confiança da plataforma necessários para poder correr os processos desejados em quatro níveis, o TCB (*Trusted Computing Base*), o TCE (*Trusted Computing Environment*), software assinado e o resto da plataforma. Estes níveis vão de totalmente confiáveis a totalmente não confiáveis, como é visto na figura seguinte.

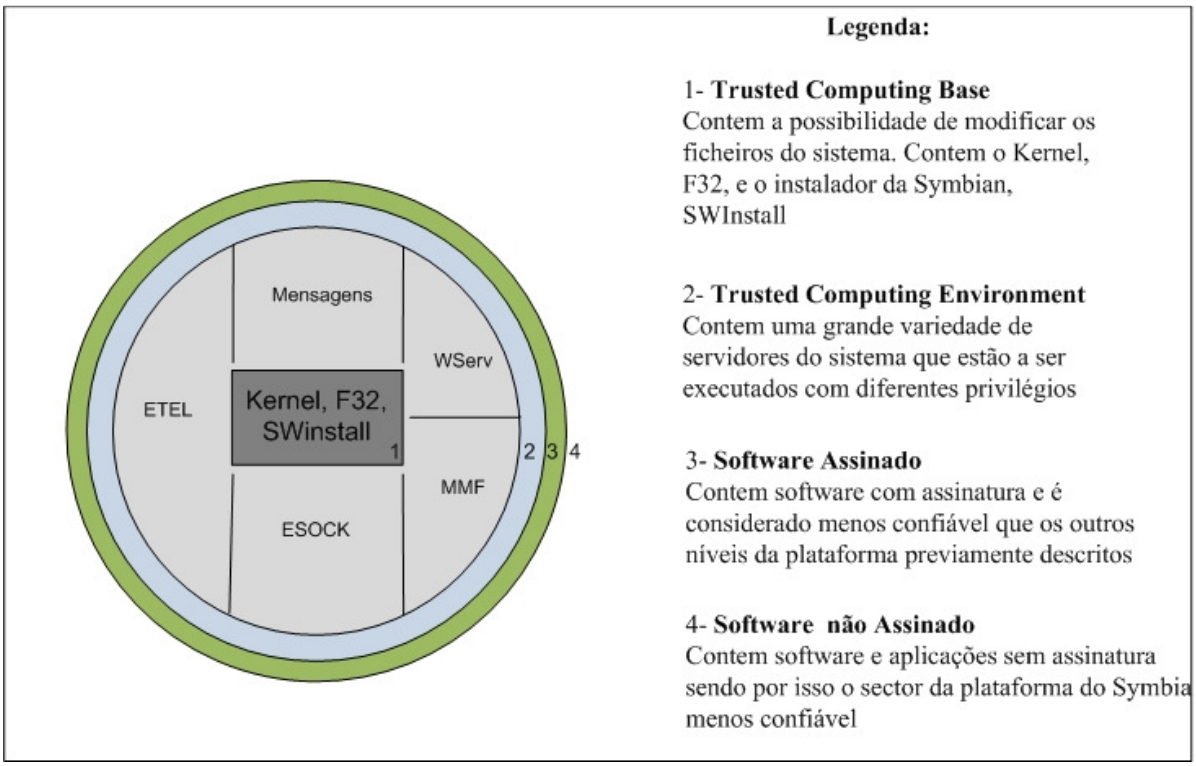

Ilustração 17 - Os quatro níveis de confiança do sistema Symbian

Trusted Computing Base

No TCB (*Trusted Computing Base*) é atingindo o nível mais baixo dos mecanismos de segurança e onde a responsabilidade de manter a integridade do núcleo do sistema atinge o nível mais elevado. O TCB avalia os detalhes de cada processo, incluindo a lista de privilégios, para deste modo obter uma correcta interpretação do que o processo tem acesso.

Num *smartphone* que suporte instalação nativa de programas, o instalador *SWInstall* que por sua vez pertence ao grupo dos mais confiáveis do sistema, extrai os ficheiros dos pacotes de instalação e valida os privilégios requisitados pelo programa com a assinatura digital que vem no pacote de instalação.

Trusted Computing Envionment

O TCE (*Trusted Computing Envionment*) implementa um sistema de servidores que correm processos e cada servidor tem privilégios limitados para executar um leque de serviços.

Por exemplo, o servidor de telecomunicações tem privilégios para aceder ao *driver* de comunicações mas não tem acesso para aceder ao servidor de UI (*User Interface*); deste modo, é possível limitar o acesso a operações de baixo nível a determinados servidores conseguindo limitar a exposição dos ataques.

Software assinado

É possível instalar software que adicione ou modifique componentes no TCB ou no TCE, no entanto só é permitido se estiver com assinatura de uma entidade confiável. Essa entidade tem de ter um certificado previamente aprovado pela *Symbian*, permitindo esse tipo de privilégio. Vejamos o seguinte caso, o acesso a rede só é disponibilizado caso a aplicação tenha o privilégio adequado e com uma assinatura de uma entidade confiável. Caso contrário o sistema operativo da *Symbian* vai recusar-se a disponibilizar acessos a software não identificado.

Software não assinado

Caso o software não esteja assinado por nenhuma entidade, não significa que não vai ter assinatura. Vai obter uma auto-assinatura e o nível mais baixo de confiança. Mas isso não significa que se trate de software corrompido ou malicioso (por exemplo, um jogo de Solitário), simplesmente não existe a necessidade de obter outro tipo de acessos. No entanto, este software é executado em modo *sandbox,* como não é confiável não pode executar nenhuma operação de segurança relevante.

5.3.2. Capacidade de Determinar Privilégios

O segundo conceito no modelo de segurança da *Symbian* é o modelo de privilégios, no qual os processos são capazes de determinar que tipos de operações podem executar.

Definição de Capacidade

A capacidade que está aqui a ser retratada é um *token* (valor que permite ao seu utilizador o direito a aceder a recursos do sistema, provando desta maneira ao sistema que está autorizado), o qual é necessário ser apresentado para ter acesso a recursos do sistema. As capacidades são incorporadas nativamente nos projectos da Symbian e são desenvolvidas em C++.

Este tipo de permissões para o modelo da *Symbian* são serviços providenciados via uma API e que em vez do nome de permissões são definidos como capacidades. Segue-se um exemplo de uma capacidade:

```
//sem capacidades
CAPABILITY         NONE
//usa a capacidade de LocalServices
CAPABILITY         LocalServices
```

O núcleo (*kernel*) do sistema guarda uma listagem de todas as capacidades dos processos que estão a serem executados, e pode ser questionado por um processo de modo a conhecer o nível de capacidade de um processo antes de lhe disponibilizar o serviço.

Como foi dito posteriormente, existem quatro níveis de confiança. No entanto, para cada um deles são necessárias três capacidades diferentes para serem identificar entre os diferentes níveis. Uma listagem de capacidades para o TCB, uma para as capacidades do sistema e outra para as capacidades do utilizador, como é possível ver na figura seguinte.

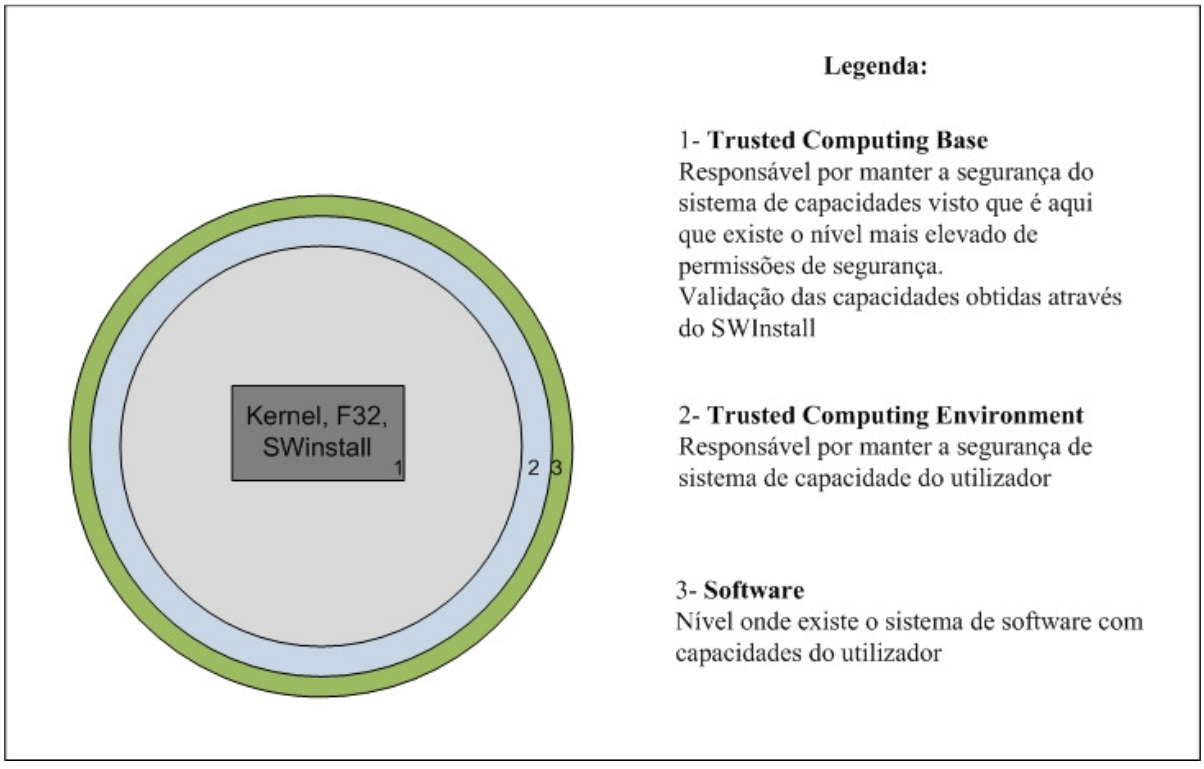

Ilustração 18 - Definições das capacidades do sistema Symbian

Capacidades de TCB

Como visto previamente, o TCB é executado com os privilégios máximos no modelo de segurança da *Symbian*, e como tal são concedidas todas as capacidades. Uma das capacidades que o TCB pode executar é a possibilidade de criar novos ficheiros executáveis e atribuir uma nova série de capacidades. Portanto, é importante que o TCB esteja sempre devidamente protegido e limitado ao máximo para evitar exploração de vulnerabilidades.

Capacidades do Sistema

Neste grupo de capacidades, permite-se aos processos executar operações sensíveis. Todo o software que eventualmente tenha necessidade de aceder a capacidades do sistema deve ter uma assinatura digital de uma entidade autorizada ou então deve ficar registada na ROM (*Read Only Memory*) do *smartphone* durante a instalação da aplicação.

Capacidades do Utilizador

Ao atribuir a capacidade de utilizador a um processo não deverá ser possível afectar a integridade do sistema operativo do *smartphone*. Como a lista de capacidades do utilizador é a mais reduzida que existe, deve ser de simples compreensão para o utilizador visto que vai ser ele que vai decidir as opções.

5.3.3. Encapsulamento de dados

No último conceito do modelo de segurança da *Symbian* vamos ver o acesso a ficheiros e de como é mantida a sua integridade e confidencialidade.

Princípios do Encapsulamento de dados

O encapsulamento dos dados (*Data Caging*, segundo a *Symbian*) é utilizado para proteger ficheiros importantes, seja código executável ou dados e independentemente de serem dados do sistema ou do utilizador. Muitos dos ficheiros do sistema são críticos para a integridade do mesmo, como tal, é importante a sua protecção. Através do encapsulamento dos dados é feito a restrição a directórios especiais. Este directórios "especiais" estão divididos em três directórios: \sys, \resource, \private e toda a restrição imposta em cada um deles é passada para os seus subdirectórios, caso existam.

Todos os outros directórios continuam públicos, fazendo com que os controlos de acesso a um ficheiro sejam determinados pelo seu directório e não por um mecanismo de controlo de acessos. Caso contrário, seria necessária a utilização do mecanismo sempre que houvesse necessidade de aceder a algum tipo de ficheiro. Portanto a utilização deste processo é extremamente simples e sem impacto na performance do dispositivo.

Directórios de Ficheiros

\sys

Relativamente ao directório \sys, o qual só é possível ser acedido através da verificação prévia do TCB, é subdividido em dois subdirectórios: \sys\bin e \sys\hash

No \sys\bin são guardados todos os ficheiros binários de uma aplicação, o que faz com que todos os ficheiros que tentem ser executados a partir de outro tipo de directório vão ser recusados pelo TCB, pois podem ser *malware*.

O \sys\hash é usado para verificar se existe alguma alteração com os executáveis que estão guardados em disco externos ao sistema (por exemplo, smartcards).

\resource

Este directório serve para a interpretação de ficheiros que são estritamente de leitura e que não vão sofrer nenhuma alteração após a sua instalação (por exemplo, bitmaps, fontes, ou ficheiros de ajuda). Só o TCB é que pode utilizar a permissão de escrita neste directório assegurando que os dados dos recursos vão ser dificilmente corrompidos.

\private

Cada ficheiro executável tem o seu próprio subdirectório encapsulado no directório \private, sendo identificado pelo seu SID. Caso aconteça que dois processos vão ser carregados dentro do mesmo executável então eles vão partilhar o mesmo subdirectório.

Analisando cada um dos três directórios de encapsulamento obtemos um quadro semelhante ao seguinte:

Directório	Necessidade de permissão de Leitura	Necessidade de permissão de Escrita
\sys	Todos os Ficheiros	TCB
\resource	Nenhuma	TCB
\private\<o_proprio_SID>	Nenhuma	Nenhuma
\private\<outro_SID>	Todos os Ficheiros	Todos os Ficheiros
\<other>	Nenhuma	Nenhuma

Tabela 4 - Listagem dos directórios e das suas permissões

Encapsulamento de dados e Capacidades

Os conceitos de capacidades e encapsulamento de dados são capazes de providenciar opções flexíveis para controlar o acesso a dados de aplicações ou do sistema, mas as capacidades que permitem processos acederem a dados encapsulados são as seguintes:

- **TCB**

  Permite escrita a todos os ficheiros executáveis e os ficheiros partilhados dos recursos

- **Todos os Ficheiros**

  Permite acesso a leitura de todos os ficheiros do sistema e permite o acesso a escrita em directórios privados de outros processos

Este tipo de capacidades são extremamente perigosas devido a nível de controlo que tem em todos os ficheiros do sistema. Como tal, depende dos programadores verificarem que tipo de acesso dar a uma aplicação que apenas precisa de ler dados ou escrever um tipo específico de informação. Para este tipo de situação o melhor é usar API´s que consigam limitar o tipo de privilégios de aplicações como é o caso de ReadUserData, WriteUserData, ReadDeviceData e WriteDeviceData.

Após analisar os modelos de segurança dos *smartphones* escolhidos, é necessário estabelecer uma comparação de cada um deles com o modelo do *Android*, permitindo assim identificar possíveis alterações e melhoramentos ao modelo actual.

6. Avaliação do Modelo de Segurança Android

Neste capítulo o objectivo, será de avaliar os modelos de segurança previamente descritos em função do modelo de segurança do *Android*. Deste modo, é possível identificar quais as implementações de segurança do *iPhone* e *Symbian* que fazem a diferença em relação ao modelo de segurança do sistema operativo *Android*.

A avaliação dos modelos de segurança dos respectivos *smartphones* com o modelo de segurança do *Android* vai permitir uma melhor compreensão de cada um deles, quais as semelhanças, em que pontos divergem e acima de tudo identificar quais as funcionalidades ou alterações que poderiam ser implementas em prol do melhoramento do modelo do *Android.*

6.1. Comparação do Modelo Android com o Modelo iPhone

O modelo de segurança do *iPhone* objectiva-se em manter o sistema inflexível quando se trata de utilizadores não autorizados, eliminação de dados e ligações seguras para os mais diversos serviços.

Na avaliação comparativa feita aos modelos de segurança do *Android* e do *iPhone* são identificados variadas disparidades. As quais foram divididas em três tópicos: perfis e acessos, aplicações e comunicação segura e eliminação de dados.

6.1.1. Perfis e Acessos

Quando se trata de utilizadores não autorizados o modelo de segurança do *iPhone* supera largamente o modelo do *Android*, visto que implementa um controlo na utilização do dispositivo móvel com senhas de acessos, as quais podem ser configuradas (desde o número de caracteres utilizados até ao tamanho das mesmas), na utilização de fortes políticas de acesso (que podem impedir o dispositivo de ser acedido por todos os utilizadores menos o administrador do mesmo) e uma configuração de perfis, os quais podem até ser cifrados, de inúmeras maneiras.

6.1.2. Aplicações

Em relação a utilização de aplicações o modelo do *iPhone* obriga que todas as instalações sejam feitas via o *software* autorizado (*iTunes*), que todas as aplicações tenham sido aprovadas previamente pela equipa da *Apple* (de modo a obterem um certificado de autenticidade) e ainda restringe o acesso a qualquer tipo de ficheiros no seu sistema através do acesso por USB. Em contrapartida, o modelo *Android* permite a instalação de aplicações através do seu *software* (*Market*) mas a única restrição que impõe é que a aplicação em questão esteja assinada, com uma chave que foi partilhada com o programador, assegurando que sabe quem é que desenvolveu a aplicação (*same-origin*). Porém, a aplicação não foi validada por nenhum elemento nem nenhuma equipa, deste modo não verificando a veracidade da aplicação e colocando ao utilizador a decisão final da sua instalação com a identificação dos acesso que a aplicação tem através do manifesto. De salientar também que todo o sistema é acedido pela utilização de um cabo USB, na qual os ficheiros não se encontram cifrados. Contudo, é usado o mesmo método de *sandbox* nos dois modelos diminuindo a possibilidade duma aplicação poder danificar ou afectar outras durante a sua execução.

6.1.3. Comunicação Segura e Eliminação de Dados

Uma das vertentes do modelo de segurança do *iPhone* acentua-se na utilização de acessos a serviços ou servidores externos de forma segura e cifrada, visto que todas as aplicações (caso seja necessário) podem usar SSL e TLS durante comunicações. Em caso de haver necessidade de restringir o acesso a dados do *smartphone*, na eventualidade de ter sido extraviado ou perdido, é possível utilizar um *Local Wipe*, apagando todos os dados existentes caso a palavra passe falhe um determinado número de vezes (o qual pode ser configurado com a utilização de um perfil de utilização) ou utilizar um *Remote Wipe*, o qual apaga todos os dados e desactiva o uso do *smartphone*. Na utilização de APIs é possível, os programadores, usarem métodos de cifrar como o caso do AES ou 3DES, para obterem uma maior segurança durante acessos em canais não seguros. O modelo *Android*, contudo, não disponibiliza a possibilidade de remotamente ou localmente eliminar os dados no caso de alguma eventualidade, é possível a utilização de *software* de terceiros para obter tais funcionalidades mas cabe ao utilizador saber se tem essa necessidade e de como trabalhar com tais ferramentas. Relativamente ao acesso de serviços ou servidores, o modelo pode usar cifras caso estejam definidas (por exemplo, o acesso ao uma página que requisite um protocolo de segurança como https) mas para ofuscar acessos e o uso de APIs, é utilizado o encapsulamento em vez de cifrar as conexões, as quais devem ser definido previamente pelo programador.

6.2. Comparação do Modelo Android com o Modelo Symbian

A *Symbian* já se encontra no mercado aproximadamente há trinta anos, deste modo, é expectável que o seu modelo, devido à longevidade do sistema operativo no mercado e à experiencia acumulada, será mais completo e robusto que do sistema operativo *Android*. A comparação entre os dois modelos foi possível dividir três capítulos: níveis de confiança, aplicações e acesso e modificação de ficheiros

6.2.1. Níveis de Confiança

Realizando uma analise mais cuidada é possível identificar que o modelo do *Symbian* é muito diferente do modelo *Android*, em primeira instancia reparamos que a estrutura é constituída por níveis de confiança que são atribuídas aos processos. Esses níveis, com diferentes tipos de permissões e acessos, são responsáveis por validar os privilégios requisitados pela aplicação, validação da assinatura digital necessária, restrição ao acesso a ficheiros ou alterações, e ainda disponibiliza servidores internos com os mais diferentes serviços. Já no modelo *Android* o cenário é bem diferente, visto que a sua estrutura base assenta no sistema POSIX, tornando as suas aplicações diferenciadas através do seu identificador, impedindo-as de entrar em contacto com qualquer outro elemento caso não esteja definido. Embora seja uma implementação segura, não é capaz de atribuir um valor de importância como no modelo *Symbian*, identificando o perigo ou o cuidado que é necessário ter com uma determinada aplicação. É de salientar, que todas as aplicações usam o método *sandbox* protegendo o sistema assim como outras aplicações, no entanto é impossível de determinar qual é o seu impacto no *smartphone* durante a sua execução.

6.2.2. Aplicações

Na utilização de aplicações, o modelo da *Symbian*, obriga que todas que necessitem de permissões críticas têm de conter uma assinatura de uma entidade confiável, a qual obteve um certificado aprovado pela *Symbian*. Caso contrário, será negado aceder a qualquer tipo de serviço ou funcionalidade dos níveis mais restritos. Embora exista interacção com o utilizador quando existe a necessidade de confirmar alguma permissão ou pedido da aplicação, o modelo da *Symbian* assume a maior parte das decisões críticas em relação ao sistema deixando assim de parte o utilizador de tomar uma decisão que se possa verificar como danificadora para o *smartphone*.

Para o modelo *Android*, a utilização de assinaturas é obrigatória mas, como já foi identificado na comparação com o modelo do *iPhone*, não é validado por nenhuma entidade ou equipa se o objectivo da aplicação está a ser cumprido, verificando a sua integridade. Qualquer tipo de acessos ou serviços será disponibilizado pela utilização do manifesto, o qual é apresentado ao utilizador, mas no entanto será ele a tomar a decisão em vez do modelo, correndo assim o risco de aceitar algum acesso indevido.

6.2.3. Acesso e Modificação de Ficheiros

Respectivamente ao acesso e modificação de ficheiros, é possível verificar que o modelo da *Symbian* foi construído de dentro para fora. Preocupando-se com os acessos aos níveis mais baixos (os que têm maior impacto no sistema operativo) e com as suas ligações interna entre componentes. Como tal, a utilização de encapsulação em ficheiros faz com que o modelo obtenha um nível superior de integridade e confidencialidade. O processo de encapsulamento é feito através da colocação de ficheiros, normalmente durante a instalação de uma aplicação, em directórios predefinidos, os quais são associados com diferentes níveis de confiança. Por exemplo, qualquer ficheiro que não esteja no directório \sys\bin não será executado, tornando impeditivo ao sistema utilizar ficheiros em directórios não autorizados. Com o uso desta estrutura, o sistema operativo reduz a necessidade de ter um mecanismo de verificação de acessos, visto que verifica a pasta e não o tipo de acessos (ou passagem de argumentos), aumentando a performance do sistema operativo.

Na instalação de aplicações em *smartphones*, o modelo *Android*, cria um identificador único (UID) para cada aplicação. Este identificador permite não só identificar qual é aplicação que está a ser executada, mas também qual o leque de permissões e acessos que têm direito e com o uso do mecanismo de *sandbox* impede aplicações de interagirem entre elas. Se houver necessidade de aceder a ficheiros ou componentes, o manifesto da aplicação terá de conter a permissão *sharedUserId* para que possa obter o mesmo identificador que outra aplicação. Assim é possível haver partilha de recursos, visto que ambas aplicações usam o mesmo UID. No entanto, para obter o *shareUserId* é necessário que ambas as aplicações tenham a mesma assinatura (a qual não é verificada por nenhuma identidade) e ao contrário da *Symbian*, tirando o uso do mecanismo do *sandbox*, não existe qualquer tipo de restrição para executar aplicações. A outra solução que o modelo *Android* oferece para o acesso de componentes entre aplicações é o ICC. O qual funciona correctamente visto que só permite o acesso aos componentes que o programador definiu previamente, mas do ponto de vista do desempenho do sistema é mais um mecanismo de validação a ser executado.

6.3. Possíveis Alterações ao Modelo de Segurança do Android

Após a análise comparativa dos modelos de segurança escolhidos com o modelo de segurança do *Android*, é possível propor alterações ao modelo de segurança. Estas alterações tentam identificar possíveis ameaças ou vulnerabilidades e apresentar as respectivas soluções. As possíveis alterações serão apresentadas através de dois quadros, respectivamente, alterações comparativamente ao modelo do *iPhone* e alterações comparativamente ao modelo da *Symbian*.

6.3.1. Alterações Comparativamente ao Modelo do iPhone

Segue-se o quadro de alterações relativamente ao modelo de segurança implementado no *iPhone*:

Modelo de Segurança do iPhone	Alterações
Perfis e Acessos	- Criação de perfis de utilização - Configuração de senhas de acessos
Aplicações	- Aplicações têm de ser aprovadas - Restringir o acesso a ficheiros através do USB
Comunicação Segura e Eliminação de Dados	- Uso de SSL e TLS durante comunicações - Eliminar dados localmente - Eliminar dados remotamente

Neste quadro foi possível identificar que o modelo de segurança do *Android* poderia modificar as suas políticas de acesso para permitir um maior controlo por parte do utilizador ou empresa. Poderia reavaliar a possibilidade de criar uma validação de aplicações mais coesa do que a existente, assim como poderia implementar uso de SSL e TLS nas comunicações e a possibilidade de eliminação de dados.

6.3.2. Alterações Comparativamente ao Modelo da Symbian

Segue-se o quadro de alterações relativamente ao modelo de segurança implementado no *Symbian*:

Modelo de Segurança do Symbian	Alterações
Níveis de Confiança	- Diferentes tipos de permissões e acessos - Validação de privilégios - Validação da assinatura digital
Aplicações	- Assinatura de uma entidade confiável - Modelo de segurança assume as decisões
Acesso e Modificações de Ficheiros	- Directórios pré-definidos - Eliminação de um mecanismo de validação

Relativamente a comparação com o modelo de segurança da *Symbian* é possível identificar algumas alterações ao nível da estruturação do acesso ao sistema e a serviços. É de salientar no entanto que este tipo de modificações teriam um forte impacto na arquitectura do sistema o que poderia não ser de fácil implementação com o modelo actual. Como identificado no quadro anterior, o uso de uma validação prévia de aplicações seria acréscimo que resultaria em instalações mais seguras por parte do utilizador, assim como uma maior independência do modelo de segurança com o utilizador. No acesso e alterações de ficheiros a utilização de directórios pré-definidos não só aumentaria a o desempenho do sistema como poderia eliminar do sistema um mecanismo de validação.

7. Conclusão

O objectivo desta dissertação era analisar, compreender e identificar possíveis alterações no modelo de segurança do sistema operativo *Android*. Para tal foi elaborada uma estrutura na qual se explicou a importância da segurança dos *smartphones*, os perigos que estão expostos, uma retrospectiva dos princípios de modelos de segurança, o funcionamento da arquitectura do sistema *Android*, os diferentes modelos de segurança implementados em outros sistemas operativos móveis e a obtenção de uma avaliação do modelo de segurança *Android* comparativamente a outros modelos de segurança.

Numa avaliação geral do sistema operativo *Android*, podemos concluir que o modelo de segurança implementado é bem construído. Desde a utilização de métodos que não permitem aplicações de interagirem com partes do sistema, até ao isolamento de aplicações de modo a prevenir qualquer tipo de corrupção do sistema operativo. O modelo de segurança do *Android* não só consegue conciliar mecanismos de segurança de sistemas exigentes, como o caso do *Linux*, como também a utilização de mecanismos de validação de referência.

Comparativamente com outros modelos de segurança, o modelo do *Android* é permeável em alguns aspectos pontuais.

Na análise com o modelo de segurança do *iPhone*, foi possível identificar a lacuna da existência de políticas de acesso, a possibilidade da eliminação dos dados do *smartphone* na eventualidade de um furto ou excesso de tentativas, a validação de aplicações e a utilização de comunicações seguras. Deste modo, é possível verificar que o modelo do *iPhone* se preocupa em definir diversos perfis de utilização, nos quais garantem a veracidade das aplicações e no caso de perda do dispositivo manter a sua confidencialidade.

Relativamente ao modelo da *Symbian*, foram também identificadas disparidades, como a utilização de diferentes níveis de acessos no sistema, o cuidado na validação de aplicações, decisões críticas executadas pelo modelo de segurança e utilização de directórios pré-definidos. É de salientar, que o modelo da *Symbian* é um modelo mais trabalhado e no qual é possível verificar que se consegue precaver de muitas ameaças devido a sua arquitectura exigente. No entanto o mais importante é manter a integridade do sistema sem importunar o utilizador, ficando para a responsabilidade do modelo de segurança as decisões mais críticas.

Levando em consideração as diferenças encontradas nos modelos de segurança com o modelo do *Android*, é possível melhorar alguns aspectos do modelo em prol do utilizador. Visto que o modelo do *Android* deixa o utilizador deferir decisões críticas que podem criar alterações do sistema, consente que as mais diversas aplicações sejam instaladas no sistema operativo sem qualquer tipo de validação e não obtêm qualquer tipo de solução nativa para o infortúnio da perda do *smartphone* e consequente perda de dados.

Em trabalhos futuros, é possível fazer a passagem daquilo que foi descrito em teoria para a prática, deste modo, averiguando a eficácia de um novo modelo de segurança do *Android* em situações reais.

Para finalizar, podemos afirmar que os modelos de segurança são uma necessidade cada vez mais indispensável. Não só para ajudar o utilizador a obter um dispositivo mais seguro, evitando possíveis ameaças, mas também para o auxiliar a decidir em caso de dúvida.

8. Bibliografia

Anderson, J. (1972). *Computer Security Technology Planning Study*. [Em linha]. Disponível em http://seclab.cs.ucdavis.edu/projects/history/papers/ande72.pdf [Consultado em 5/09/2010].

Android Development. (2010). [Em linha] Disponível em http://code.google.com/android [Consultado em 10/11/2009].

Antonelli, C., Mobile Device Security (2009), Information Technology Security Services, University of Michigan, 4-6.

Apple. (2009). *iPhone in Business: Security Overview*. [Em linha]. Disponível em http://images.apple.com/iphone/business/docs/iPhone_Security_Overview.pdf [Consultado em 15/09/2010].

Apple. (2009). *Security Overview*. [Em linha]. Disponível em http://developer.apple.com/library/ios/documentation/Security/Conceptual/Security_Overview/Security_Overview.pdf [Consultado em 15/09/2010].

Apple. (2010). *IOS Reference Library*. [Em linha]. Disponível em http://developer.apple.com/library/ios/#documentation/Security/Conceptual/Security_Overview/Architecture/Architecture.html#//apple_ref/doc/uid/TP30000976-CH202-TPXREF101 [Consultado em 15/09/2010].

Becher, M., Freiling, F., Towards Dynamic Malware Analysis to Increase Mobile Device Security (2007), University of Mannheim, 1-4.

Common Criteria. (2010). [Em linha]. Disponível em http://www.commoncriteriaportal.org/ [Consultado em 15/12/2010].

Dunham, K., Abu-Nimeh, S., Becher M., Seth, F.,Hernaclti, H., Morales, J.,Wright, C., Mobile Malware Attacks and Defense (2009), Elsevier Inc. ISBN 13: 978-1-59749-298-0, 3-28.

Gartner [Em linha]. Disponível em http://www.gartner.com/it/page.jsp?id=1306513 [Consultado em 25/06/2010].

Google Developers, Google [Em linha]. Disponível em http://developer.android.com/guide/topics/manifest/manifest-element.html#uid [Consultado em 25/10/2010].

Heath, C. (2006). *Symbian OS Platform Security: Software Development Using the Symbian OS Security Architecture*, Wiley**.**

IDC Mobile Phone Tracker Portugal. (2010). [Em linha]. Disponível em http://www.idc.pt/press/pr_2010-09-13.jsp [Consultado em 15/09/2010].

ISACA. (2010). *Securing Mobile Devices* . Rolling Meadows, USA.

Levin, T., Irvine, C., Benzel, T., Bhaskara, G., Clark, P., Nguyen, T. Design Principles and Guidelines for Security(2007), Naval Postgraduate School, Monterey, California, 3-21.

McDaniel, P., Ongtang, M., Enck, W. (2009). *Understanding Android Security*. IEEE Security & Privacy.

Miller, C., Honoroff, J., Mason, J. (2007). *Security Evaluation of Apple's iPhone, Independent Security Evaluators*. [Em linha]. Disponível em http://securityevaluators.com/files/papers/exploitingiphone.pdf [Consultado em 1/09/2010].

Newslite. (2009). [Em linha]. Disponível em http://newslite.tv/2009/12/01/10000-mobiles-lost-in-london-t.html [Consultado em 26/12/2010].

Open Handset Alliance. (2010). [Em linha]. Disponível em http://www.openhandsetalliance.com/ [Consultado em 12/07/2010].

Saltzer, J., Schroeder, M. (1975). The Protection of Information in Computer Systems. [Em linha]. Disponível em http://www.cs.virginia.edu/~evans/cs551/saltzer/ [Consultado em 1/09/2010].

Schneier, B., What is security? (2009). [Em linha]. http://etspictures.com/video_popup_bruce_2.html[Consultado em 04/11/2010].

Shabtai, A., Fledel, Y., Kanonov, U., Elovici. Y., Dolev S., Glezer. C. (2009). Google Android: A Comprehensive Security Assessment, IEEE Security & Privacy, vol, nº, 6-10.

Speckmann, B. (2008). The Android mobile platform. [Em linha]. Disponível em http://www.emich.edu/compsci/projects/Master_Thesis_-_Benjamin_Speckmann.pdf, 5-16.

Spectrum IEEE. [Em linha]. Disponível em http://spectrum.ieee.org/geek-life/tools-toys/crossplatform-smartphone-apps-still-difficult [Consultado em 04/9/2010].

State of Illinois, Mobile Device Security Policy (2009), Department of Central Management Services Bureau of Communication and Computer Services, 4-5.

US-Cert, Cyber Threats to Mobile Devices (2010), Technical Information Paper [Em linha]. Disponível em http://www.us-cert.gov/reading_room/TIP10-105-01.pdf, 1-6 [Consultado em 10/11/2010].

Ware,W., Security and privacy in computer systems (1967), AFIPS Cont. Proc., vol. 30, 287-290.