ISSN: 1984-3151

UMA ABORDAGEM SOBRE PROBLEMAS DE SEGURANÇA DA INFORMAÇÃO POR MEIO DO DESENVOLVIMENTO DE APLICAÇÕES MALICIOSAS PARA ANDROID

AN APPROACH TO PROBLEMS OF INFORMATION SECURITY THROUGH THE DEVELOPMENT OF MALICIOUS APPLICATIONS FOR ANDROID

Leandro Rodrigues Inácio[1]; Antônio Ricardo Leocádio Gomes[2]

1 Analista de Planejamento. UniBH, 2014. JN Serviços Industriais. Belo Horizonte, MG. leandro.inaccio@gmail.com.

2 Especialista em Novas Tecnologias em Educação e Treinamento. UniBH, 2014. Professor. Centro Universitário de Belo Horizonte. Belo Horizonte, MG. antonio.gomes@prof.unibh.br.

Recebido em: 25/09/2014 - Aprovado em: 20/11/2014 - Disponibilizado em: 30/11/2014

RESUMO: Este artigo discute o desenvolvimento de aplicações maliciosas e o nível de conhecimento do usuário sobre segurança do Sistema Operacional móvel Android. Este trabalho visa a demonstrar o nível de dificuldade envolvido no desenvolvimento de uma aplicação maliciosa até a fase de disseminação.
PALAVRAS-CHAVE: Android. Malware. Segurança da Informação.

ABSTRACT: This article discusses the development of malicious applications and the level of user knowledge about security of mobile operating system Android. This paper demonstrates the level of difficulty involved in the development of a malicious application to the dissemination phase.
KEYWORDS: Android. Malware. Information Security.

---

1 INTRODUÇÃO

A comunicação é uma necessidade básica dos seres humanos. O ato de comunicar está relacionado diretamente a duas palavras: transmissão e contato. Seja através da fala, de gestos ou expressão corporal, é possível expressar sentimentos. Em um mundo cada vez mais conectado, o ser humano tem a necessidade de trocar, conectar e compartilhar as mais diversas experiências de vida com outras pessoas, de acordo com o psicólogo norte-americano Abraham Maslow (CATHO, 2012). As novas tecnologias permitem cada vez mais a troca de informações em praticamente qualquer parte do mundo. Os novos dispositivos denominados s*martphones* tem se despontado neste quesito de integração entre os humanos.

Os *smartphones* são dispositivos híbridos que combinam a funcionalidade de um celular e um PC (*Personal Computer*). Os *smartphones* englobam as principais tecnologias de comunicação atuais em um único dispositivo como: chamada de áudio e vídeo, SMS, Email, GPS, Comunicadores instantâneos, Redes Sociais etc. Outra característica importante que possibilitou tais funcionalidades e o gerenciamento de inúmeras aplicações em um único dispositivo foi o amadurecimento do Sistema Operacional (SO) móvel. Atualmente três Sistemas Operacionais disputam o

mercado móvel mundial: o IOS da Apple, o Android da Google e o Windows Phone da Microsoft. No primeiro trimestre de 2013, houve um aumento de 86% nas vendas de *smartphones* em relação ao mesmo período do ano passado no Brasil (ZERO HORA, 2013). A expectativa é de que mais de 950 milhões de *smartphones* sejam vendidos somente este ano no mundo, de acordo com os estudos da IDC.

Um importante fator que contribuiu para o crescimento do mercado de *smartphones* no Brasil é a geração Y ou *millennials*, jovens com idade entre 18 e 30 anos (IDGNOW!, 2013a). Estes jovens lideram o crescimento de *smartphones* no mercado brasileiro e são também importantes motores para o crescimento da oferta de dados móveis. Mesmo com uma renda mais baixa, os *millennials* representam 25% da receita da Telefônica Brasil, segundo o executivo Santiago Fernández Valbuena.

No segundo trimestre de 2013, pela primeira vez, os *smartphones* superaram a venda de celulares no Brasil, segundo pesquisa realizada pela IDC Brasil. Dos 15 milhões de celulares vendidos de abril a junho, deste total 54% foram de *smartphones*, contra 46% de celulares tradicionais. Enquanto as vendas de *smartphones* cresceram 110%, os celulares tradicionais tiveram uma retração de 35%. Este aumento na venda de *smartphones* ocorreu por vários motivos, dentre os quais se destacam o aumento do portfólio de produtos dos fabricantes em diferentes faixas de preço, aumento das promoções e o marketing dos produtos. Outro ponto importante que impulsionou este crescimento foi o início da aplicação da desoneração fiscal (MP do bem) para produtores locais de *smartphones*. Um *smartphone* que custava em média $316 no semestre anterior passou a custar $240 no segundo semestre. Outro número que se destaca é o de *smartphones* com o SO Android, do total de 8,3 milhões de *smartphones* vendidos no segundo semestre, 90% são *smartphones* com o SO da Google (IDC, 2013).

"O Android é a primeira plataforma para aplicações móveis completamente livre e de código aberto [...]". (LECHETA, 2013, p.27). Foi inicialmente desenvolvido pela Android Inc. e comprada pela Google em agosto de 2005. Foram lançadas diversas versões do Android, uma curiosidade é que as versões recebem o nome de uma sobremesa, com exceção do Android 1.0 e 1.1, lançadas em 2008 e 2009. A primeira versão do Android que recebeu o nome de uma sobremesa foi a 1.5 lançada em 2009, chamada de *Cupcake* e, a partir dessa versão, tornou-se padrão as versões do Android receberem este tipo de nomeação. A última versão lançada, a 4.4, recebeu o nome de *Kitkat*. Em março de 2013, Larry Page, cofundador e atual presidente da Google, postou no blog oficial da empresa que o número de aparelhos com Android ultrapassa os 750 milhões e que 25 bilhões de aplicativos já foram baixados do serviço Google Play, a loja oficial de aplicativos para Android (TECMUNDO, 2013a). Outro número que chama a atenção é a quantidade de *malware* desenvolvida para o Android em um curto espaço de tempo.

Em um relatório divulgado pela empresa F-Secure, o Android foi alvo de 79% de todo o *malware* móvel em 2012, contra 66,7% em 2011 e apenas 11,25% em 2010 (PORTAL TERRA, 2013). O Android demorou apenas 3 anos para alcançar o mesmo volume de ameaças que computadores demoraram 14 anos. Outro dado preocupante é que apenas 20% dos usuários de dispositivos Android utilizam algum tipo de antivírus, segundo a Trend Micro (IDGNOW, 2013b). Só no primeiro semestre de 2013, foram registradas 520 mil novas variantes de *malwares*, segundo estudo realizado pela empresa G DATA (IPNEWS, 2013). A expectativa é que até o final do ano o número de *malwares* desenvolvidos para Android chegue a 1 milhão (IDGNOW, 2013b).

No último ano, 804 novas famílias ou variantes de *malware* foram desenvolvidas para Android, representando 97% de todo *malware mobile*

(PPLWARE, 2014). Outra informação importante e que apenas 0,1% deste *malware* está na Google Play, loja oficial do Android.

Outro ponto que impulsionou o aumento no número de malwares para Android foi o desenvolvimento de software conhecido como *binders*, que automatizam o processo de reempacotamento (repackaging) e injeção de códigos maliciosos em aplicações legítimas (SYMANTEC, 2013). Um exemplo desse tipo de software é o Android APK Binder, que custa em torno de $35 e permite monitorar e fazer ligações, enviar SMS, obter coordenadas de GPS, ativar e utilizar a câmera e microfone e acessar arquivos armazenados no *smartphone*.

Baseado nos números apresentados e sabendo das tarefas importantes que um *smartphone* com sistema operacional Android possibilita realizar e da existência de *software* que facilita o processo de inserção de códigos maliciosos em aplicativos Android legítimos, este trabalho tem como objetivo geral averiguar a pertinência e recursos necessários para o desenvolvimento de um *malware*, bem como o nível de *expertise* dos usuários no quesito segurança. Como objetivos específicos, averiguar a compatibilidade conceitual da API de desenvolvimento AFE (Android Framework for Exploitation) com o sistema operacional Android em um smartphone com a versão Jelly Bean e testar se os aplicativos modificados são detectados pelos sistemas de segurança das lojas.

Para avaliar o nível de conhecimento dos usuários, foi utilizado um formulário com perguntas relacionadas à segurança da informação e o SO Android.

Nas próximas seções, serão apresentados os embasamentos teóricos, os tipos de *software* e a metodologia utilizada, o laboratório desenvolvido para testes, as modificações realizadas no código de uma aplicação legítima e os resultados obtidos com a utilização do *framework* AFE e a pesquisa com usuários.

2 REFERENCIAL TEÓRICO

Para realizar a pesquisa, foram utilizados alguns tipos de *software* com o objetivo de analisar os dados obtidos ao longo deste trabalho.

2.1 ANDROID

Desenvolvido inicialmente pela empresa Android Inc., comprado e aprimorado posteriormente pela Google, em agosto de 2005, é um SO móvel de código fonte aberto e livre, baseado no Kernel Linux 2.6. Segundo Lecheta (2013, p.22), o 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".

Por trás do Android, além do Google, está o OHA (*Open Handset Alliance*), que

> [...] é um grupo formado por gigantes do mercado de telefonia de celulares liderados pelo Google. Entre alguns integrantes do grupo estão nomes consagrados como a HTC, LG, Motorola, Samsung, Sony Ericsson, Toshiba, HTC, Huawei, Sprint Nextel, China Mobile, T-Mobile, Asus, Intel, Acer, Dell, Garmin e muito mais. (LECHETA, 2013, p. 23).

Este grupo liderado pela Google foi criado para padronizar uma plataforma de código aberto e livre para celulares, para atender a todas as exigências do mercado. O objetivo principal dessa aliança é criar uma plataforma robusta para o desenvolvimento de aplicações corporativas e ao mesmo tempo fornecer aos usuários finais um produto que atenda a todas as suas exigências.

A arquitetura Android é construída em camadas como a maioria das outras plataformas. As camadas mais baixas fornecem serviços para as camadas

superiores. As camadas que compõem o SO Android são: Linux *Kernel*, *Libraries* (Bibliotecas), *Application Framework* (*Framework* de Aplicação) e *Applications* (Aplicações), conforme apresentado na Figura 1.

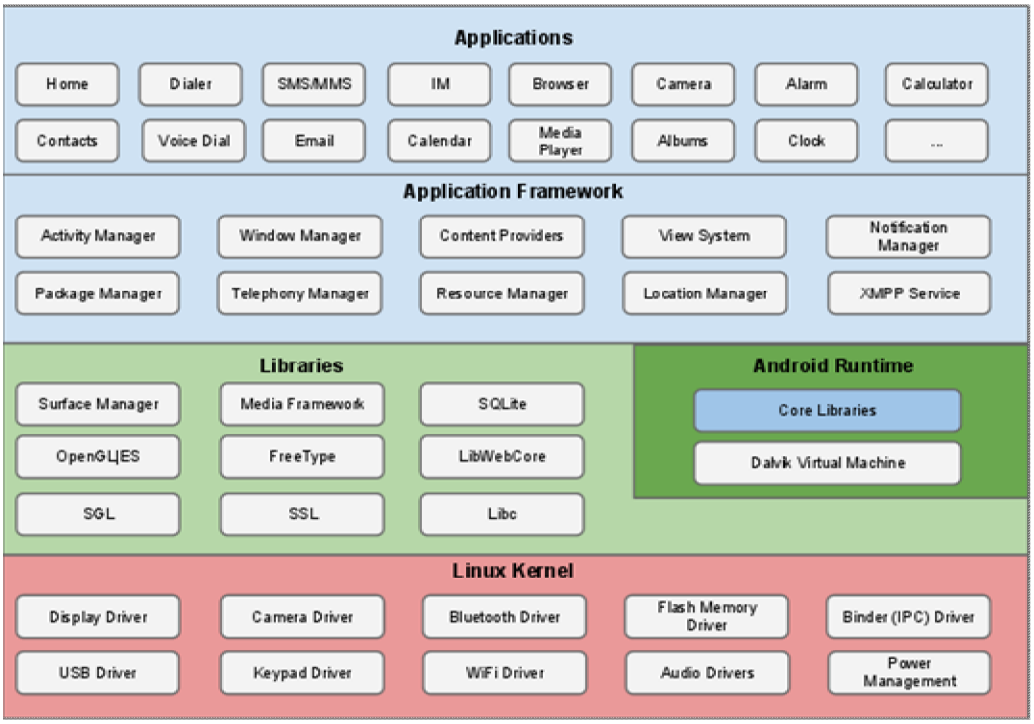


Figura 1 – Arquitetura Android
Fonte: SOURCE ANDROID, 2013

Basicamente é dividida em 4 camadas, em que cada camada é responsável por gerenciar seus próprios recursos. Essas camadas funcionam da seguinte forma, começando pela camada mais baixa até a mais alta:

- Linux *Kernel*: o Android é baseado no *Kernel* Linux 2.6, que é responsável por gerenciar os processos, *threads*, memória, segurança dos arquivos e pastas, além da rede e *drivers*. "Essa é a camada onde todos os drivers de hardware específicos do dispositivo serão executados, permitindo aos fornecedores de hardware desenvolver drivers em um ambiente familiar" (SIX, 2012, p. 32).
- *Libraries* (Bibliotecas): nessa camada estão as bibliotecas básicas do sistema (bibliotecas nativas), "[...] que são módulos de código compilados para código de máquina nativo do dispositivo e fornecem alguns dos serviços comuns que são disponíveis para apps e outros programas" (Six, 2012, p. 32). A maior parte dessas bibliotecas e serviços foi desenvolvida em C/C++.
- Android *Runtime*: Segundo Six (2012, p. 32), "Cada app é executado na sua própria instância do runtime Android, e a base de cada instância é uma Máquina Virtual Dalvik (Dalvik Virtual Machine – VM)". A Dalvik é uma máquina virtual escrita para dispositivos móveis (uma implementação de uma JVM (*Java Virtual Machine*)), que requer pouca memória e é projetada para permitir que

múltiplas instâncias de máquina virtual rodem ao mesmo tempo, deixando para o sistema operacional o gerenciamento dos outros recursos do sistema. A VM Dalvik executa arquivos no formato Dalvik *Executable* (*.dex), que são *bytecodes* Java (*.class) convertidos com a ferramenta "dx", desenvolvida pela Google.

- *Application Framework* (*Framework* de Aplicação): estão todas as API's e recursos utilizados pelos aplicativos. "Sendo executados nesse nível, encontram-se entidades como o *Package Manager*, responsável pelo gerenciamento de apps no telefone, e o *Activity Manager*, responsável por carregar *Activities* e gerenciar a pilha *Activity*" (SIX, 2012, p.33).
- *Application* (Aplicação): Essa é a camada mais alta da arquitetura Android, na qual estão localizados os aplicativos do usuário. Nesta camada estão os aplicativos padrões do sistema e os desenvolvidos pelos programadores e pela Google. Aplicativos, como navegador de internet, email, calendário, etc, são escritos na linguagem Java.

O modelo de segurança do Android é provido pela implementação do *Kernel* Linux, baseado no conceito de gerenciamento de usuários e grupos. Para cada usuário e grupo criado no sistema Linux é atribuído um ID. O usuário criado recebe um ID de usuário (*user* ID ou UID), e o grupo recebe um ID de grupo (*group* ID ou GID). Cada usuário precisa pertencer a um ou mais grupos. Como cada arquivo ou processo pertence a um usuário, este arquivo/processo pertence ao grupo de seu proprietário, conforme consta em (SIX, 2012, p. 33.). Dessa forma, cada arquivo/processo está associado a um UID e a um GID.

Outra propriedade importante herdada do Linux é o sistema de permissão e privilégios. Este sistema permite que sejam definidas políticas para acesso dos usuários e grupos aos arquivos, diretórios e programas do sistema.

Os privilégios são divididos em três classes: proprietário (*owner*), grupo (*group*) e outros (*other*). E cada classe é composta por três níveis básicos de permissões: permissão de leitura (*read* - R), permissão de escrita (*write* - W) e permissão de execução (*execute* – X).

Conforme Six (2012), quando um aplicativo é instalado no Android,

> [...] um novo ID de usuário (um que não esteja atualmente em uso no dispositivo) é criado e o novo app é executado sob esse IUD. Além disso, todos os dados armazenados por esse aplicativo recebem o mesmo UID, seja um arquivo, base de dados ou outros recursos. As permissões do Linux de recursos para esse app são configuradas para permitir permissão total pelo UID associado e nenhuma permissão de outro modo. [...] O Linux impede que apps com diferentes UIDs acessem dados, ou seja, acessem o processo de memória de outros apps, fornecendo assim a base para a separação de entre apps na plataforma Android [...]. (SIX, 2012, p.34).

Apesar de o modelo de segurança do Android prover o isolamento de dados e recursos de cada aplicativo, é possível compartilhar, de forma segura, dados e recursos com outros aplicativos. Para isso é necessário que os aplicativos que irão compartilhar os dados sejam do mesmo desenvolvedor e seja declarada no *AndroidManifest.xml* a mesma UID no atributo *sharedUserId* na *tag* <*manifest*>, e também que estes aplicativos sejam assinados com o mesmo certificado digital, como reitera Six (2012, p.38). Dessa forma o *Kernel* Linux vai reconhecer estes aplicativos como o mesmo aplicativo.

O sistema de arquivos do Android cria um diretório (*/data/data/app_package_name*) específico para cada aplicativo instalado. Este diretório é configurado para que apenas o aplicativo proprietário tenha permissões sobre ele, conforme consta em Six (2012, p.41).

Dessa forma nenhum outro aplicativo tem acesso a este diretório, com a exceção das UIDs compartilhadas.

O Android implementa um outro sistema de segurança baseado em permissões, conhecido como Permissões de Aplicativo. Este sistema permite definir e limitar os serviços (SMS, GPS, Bluetooth, câmera etc) que cada aplicativo poderá executar. Essas permissões são definidas no arquivo *AndroidManifest.xml*, presente em qualquer aplicativo desenvolvido para a plataforma Android. Antes de qualquer instalação, é apresentada para o usuário uma lista das permissões de serviços do aplicativo, podendo o usuário recusar e cancelar a sua instalação.

As permissões de serviços são divididas em quatro categorias:

- Normal – não pode causar danos ao usuário
- *Dangerous* (perigosa) – pode causar danos ao usuário, como roubo de informações pessoais e prejuízos financeiros.
- *Signature* (assinatura) – destinada a aplicativos desenvolvidos pelo mesmo desenvolvedor e irá compartilhar dados e recursos.
- *Signature Or System* (Assinatura Ou Sistema) – semelhante a Signature, essa permissão é usada por fabricantes de dispositivos para permitir aplicativos criados pelas operadoras.

2.2 Loja e Instalação de Aplicativos

A Google Play é a loja de aplicativos oficial do Android, anteriormente chamada de Android Market. Além de aplicativos, é possível comprar livros e alugar filmes. Nela os usuários podem realizar pesquisas, ver a descrição e funcionalidades de um aplicativo, bem como instalar, desinstalar ou atualizar um aplicativo.

A publicação de aplicativos na Google Play é feita pelo próprio desenvolvedor do aplicativo. Para publicar um aplicativo, é necessário assiná-lo com um certificado digital.

Um aplicativo pode ser assinado de duas formas. Em modo *debug*, quando o aplicativo ainda está na fase de desenvolvimento e testes. Ou em modo *release*, quando o aplicativo foi finalizado, testado e está pronto para ser utilizado. Aplicativos que são assinados em modo *debug* ou não foram assinados ou não podem ser publicados.

Essa assinatura tem como objetivo identificar o autor do código, verificar se o aplicativo foi modificado e estabelecer uma confiança entre aplicativos. O Google não exige que o certificado digital tenha uma CA (Autoridade Certificadora) - este certificado pode ser autoassinado. O certificado digital pode ser gerado através da ferramenta Keytool, disponível no Java JDK. O próximo passo é assinar o aplicativo com a chave privada do certificado digital que foi gerado, com a ferramenta Jarsinger (IBM, 2010). Após estes procedimentos o aplicativo está pronto para ser publicado na Google Play, bastando apenas criar-se uma conta no Google e pagar uma taxa de registro.

Além da Google Play, existem lojas alternativas (não oficiais) de aplicativos para Android, como a Aptoide, F-Droid e BlackMart.

Apesar de a loja oficial do Google não estar livre de *malwares*, lojas como a BlackMart e outras lojas alternativas estão mais susceptíveis a aplicações maliciosas. Um estudo feito pela AV-Comparatives entre novembro de 2012 e maio de 2013, entre as 20 maiores lojas deste tipo, encontrou 7.175 *malwares* e *greywares* (TECMUNDO, 2013b).

A instalação de aplicativos no Android é feita apenas com a autorização do usuário. Ao clicar em “instalar”, são mostradas em uma janela as permissões necessárias para o funcionamento do aplicativo. Ao

concordar com as permissões solicitadas, o processo de instalação é iniciado.

Durante o processo de instalação, dois aplicativos do sistema são utilizados: o *Package Manager* e o *Package Installer*. O *Package Installer* é responsável por gerenciar os aplicativos instalados e interagir com o usuário no processo de instalação. E o *Package Manager* é responsável por instalar, desinstalar ou atualizar um aplicativo.

2.3 MALWARE

Assim como a plataforma Windows, o Android se tornou o alvo principal de ataques envolvendo código malicioso de acordo com Kaspersky (2013).

Malware,

> "[...] é a contração de "malicious software" (programa malicioso) e identifica qualquer programa desenvolvido com o propósito de causar dano a um computador, sistema ou redes de computadores [...]. Os mais comuns são os vírus, worms e cavalos de troia [...]. O malware também tenta explorar as vulnerabilidades existentes nos sistemas, tornando sua entrada discreta e fácil". (CASSANTI, 2014, p.8).

O *software* malicioso pode ser dividido em duas categorias: aqueles que precisam de um programa hospedeiro e aqueles que são independentes. E podem ser diferenciados entre ameaças de *software* que não se replicam e aquelas que se replicam, como consta em Stallings (2010, p.428).

2.4 VETORES DE ATAQUES

De acordo com o glossário da Symantec, umas das maiores empresas desenvolvedoras de soluções de segurança e também de assinaturas e certificados digitais, após aquisição da VeriSign, vetor de ataque, é o método que o agente ameaçador utiliza para atacar um sistema.

Dentre estes métodos destacam-se a exploração de vulnerabilidades em protocolos, criptografia, aplicações, sistema operacional etc, e a manipulação de pessoas com a engenharia social.

2.4.1 EXPLORAÇÃO DE SOFTWARE

Conforme Hoglund e McGraw (2006, apud Viega e McGraw, 2001, p. xix), "não seria necessário empregar tanto tempo, dinheiro e trabalho em segurança de rede se não tivéssemos uma segurança de *software* tão ruim".

Ainda de acordo com Hoglund e McGraw, a falha ou mau funcionamento de um *software* pode:

- Expor dados confidenciais a usuários não autorizados (inclusive invasores);
- Travar ou parar de funcionar ao ser exposto a entradas defeituosas;
- Permitir que um invasor "injete" um código e o execute;
- Executar comandos sigilosos em nome de um invasor esperto. (HOGLUND; McGRAW, 2006, p.3).

2.4.2 ENGENHARIA SOCIAL

Outro vetor de ataque que merece atenção é a que envolve a manipulação de pessoas. A técnica conhecida como Engenharia Social (*Social Engineering*) explora o elo mais fraco da segurança da informação, o fator humano. A definição mais apropriada para essa técnica segundo Mann (2011) é: "Manipular pessoas, enganando-as, para que forneçam informações ou executem uma ação".

Ataques de engenharia social são mais destrutivos quando utilizam elementos de tecnologia, como consta em Mitnick e Simon (2003, p.152). A combinação destes ataques também pode ser utilizada contra usuários da plataforma Android.

O *cracker* utiliza dessas duas técnicas para fazer com que o usuário baixe uma aplicação (alterada com a injeção de códigos maliciosos, que pode ser realizada com o AFE) de um site qualquer. Dessa forma o

cracker pode obter dados pessoais ou obter vantagem financeira do usuário que instalar a aplicação.

2.5 ANDROID SDK

O Android SDK (*Software Development Kit*) é um *software* para o desenvolvimento e teste de aplicativos para o Android, que permite emular um dispositivo Android, além de possuir ferramentas utilitárias e uma API completa para a linguagem Java, conforme consta em Lecheta (2013, p.32).

2.6 ANDROID FRAMEWORK FOR EXPLOITATION

A utilização de novas tecnologias para desenvolvimento de *software* mais eficiente vem sendo utilizado amplamente. O *framework* é uma dessas novas tecnologias e permite ao desenvolvedor programar de forma mais rápida, fácil e eficiente. Assim como o Android SDK, um *framework* dispõe de um ambiente completo tanto para desenvolvimento e testes de *software*, além de metodologias que auxiliam durante todas as fases de desenvolvimento de um *software*.

Criado por Aditya Gupta e Subho Halder, AFE é um *framework* para a realização de pesquisas voltadas para a área de segurança da informação da plataforma Android (XYSEC, 2012, p.2, tradução dos autores). Desenvolvido principalmente na linguagem de programação Python, o AFE utiliza *scripts*, bibliotecas externas no formato (.jar) para automatizar algumas tarefas e algumas ferramentas como Apktool, utilizada para fazer a engenharia reversa de arquivos apk, e Signapk, para assinar os aplicativos.

Outra ferramenta de fundamental importância para funcionamento do AFE é a Ant (Another Neat Tool), desenvolvida por James Duncan Davidson e mantida pela Apache *Software* Foundation. Ant é uma ferramenta escrita em Java utilizada para automatizar a construção de *software* em linha de comando e facilitar seu gerenciamento. O Ant fornece uma série de funcionalidades que permite compilar, montar, testar e executar aplicativos Java. È também extremamente flexível, permitindo ao usuário desenvolver suas próprias tarefas.

O *framework* AFE é dividido em 5 módulos:

Malware Creator – Criação de *malware* e módulos *Botnet*. Usado para injetar códigos maliciosos em aplicações legitimas. (XYSEC, 2012, p.2, tradução nossa).

Listener – usado para escutar e capturar os dados de um telefone/emulador. (XYSEC, 2012, p.2, tradução nossa).

Exploiter – usado para explorar várias vulnerabilidades em aplicativos e plataforma. (XYSEC, 2012, p.2, tradução nossa).

Stealer – usado para roubar informações de um telefone, incluindo contatos, histórico de chamadas, mensagens de texto, arquivos do cartão SD, entre outros (XYSEC, 2012, p.2, tradução nossa).

Crypter – tornar um *malware* indetectável a um antivírus (XYSEC, 2012, p.2, tradução nossa).

Com o AFE também é possível definir a versão mínima suportada e a versão do Android para qual o aplicativo será otimizado.

A extensibilidade é outra característica importante deste *framework*. Através do desenvolvimento de *plugins* é possível integrar novas funcionalidades.

2.7 AFE E O MODELO ANDROID

O Android é desenvolvido sobre uma estrutura robusta provida pelo *Kernel* Linux com a implementação de algumas melhorias, como o isolamento de dados e recursos de cada aplicativo e com permissões de

aplicativo, responsável por definir o nível de acesso a cada serviço.

Para cada aplicativo, é definido pelo desenvolvedor quais e qual o nível de acesso a cada serviço. Aplicativos desenvolvidos para uma mesma finalidade podem ter permissões de aplicativo diferentes. O modelo Android não define um padrão de permissões de aplicativo para cada tipo de aplicativo.

Apesar de não ser uma vulnerabilidade ou falha do sistema Android, desenvolvedores de aplicações maliciosas têm explorado com frequência este recurso.

As funcionalidades do AFE estudadas neste artigo atuam exatamente sobre permissões de aplicativo. O *Malware Creator* é responsável por criar aplicações maliciosas explorando as permissões de aplicativo do Android e também da falta de boas práticas de programação e desenvolvimento seguro por parte dos desenvolvedores. Técnica como a ofuscação de código, além de proteger o código, dificulta o processo de engenharia reversa.

O módulo *malware* utiliza permissões de aplicativo e provedores de conteúdos (*content providers*) para roubar informações. E o módulo *apk_injector,* além das permissões de serviços, utiliza um receptor de transmissão (*broadcast receiver*) para receber instruções de um *cracker*.

3 METODOLOGIA

Para a realização dos estudos foram utilizados aplicativos em destaque da loja oficial do Android, a Google Play.

O AFE foi utilizado para se fazer a alteração dos aplicativos baixados. Em seguida estes aplicativos foram enviados a uma loja não oficial de aplicativos para Android. A loja escolhida foi a BlackMart, por ser bastante popular e por não exigir cadastro para utilização e nem para envio de aplicativos.

Para demonstrar a alteração realizada nos aplicativos, foi utilizada a distribuição Linux Santoku, destinada a forense, engenharia reversa, análise de *malware* e testes de segurança em dispositivos mobile.

3.1 TESTE COM APLICATIVOS

Como contexto de análise foi utilizado o módulo *Malware Creator*, utilizado para injetar códigos maliciosos em aplicativos da respectiva loja citada anteriormente.

Para dificultar a detecção pelos sistemas de segurança das lojas, poderá ser utilizado o módulo Crypter para ofuscação de código. Outro teste que poderá ser feito futuramente será testar a eficiência dos antivírus desenvolvidos para o Android.

3.2 EXPERTISE DO USUÁRIO

Para avaliar o nível de conhecimento do usuário, foi desenvolvido um formulário online (Google Formulário) com perguntas de múltipla escolha sobre a plataforma Android e segurança da informação, que foi enviado via *email*.

3.3 DISPOSITIVO PARA TESTE

Com o objetivo de tornar os testes mais próximos da realidade, foi utilizado um *smartphone* LG L1 II E410 f ao invés de um simulador. A versão do Android escolhida para teste foi a 4.1.2, conforme Figura 2, conhecida como *Jelly Bean* (versão mais utilizada do Android) (ANDROID AUTHORITY, 2014).

Figura 2 – Versão do Android

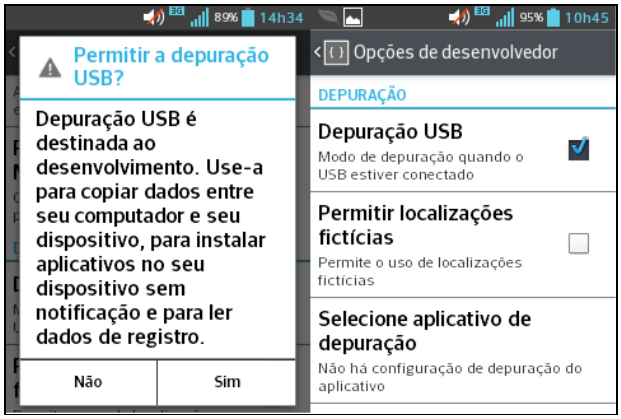


Figura 3 – Depuração USB

4 DESENVOLVIMENTO

A seguir são descritas as etapas realizadas durante os testes, desde a parte de configuração do *smartphone* até a tentativa em distribuir o aplicativo na loja para Android definida anteriormente.

4.1 CONFIGURANDO O DISPOSITIVO

Antes de se iniciar os testes propostos neste artigo, foram realizadas algumas configurações no *smartphone*, para permitir a captura de alguns dados e a instalação de aplicativos.

A primeira configuração realizada foi ativar o modo de depuração USB. Por motivos de segurança e por ser pouco utilizado por usuários leigos, este modo vem por padrão desativado. Este modo é usado principalmente por desenvolvedores para realizar testes em seus aplicativos. Através deste modo, é possível instalar aplicativos sem a necessidade de interagir com o *smartphone*, obter dados do aplicativo em execução e ter um maior controle sobre algumas configurações. Na Figura 3 é possível ver a mensagem de aviso sobre a ativação da Depuração USB.

Outra configuração feita foi permitir a instalação de fontes desconhecidas. Essa configuração permite instalar aplicativos de outras lojas ou que estejam disponíveis em algum site. A ativação deste modo pode trazer riscos à segurança do *smartphone* e aos dados do usuário, uma vez que aplicativos instalados dessa forma não são testados pelo sistema de segurança da loja oficial do Google. Na Figura 4 é mostrada a mensagem avisando o usuário sobre os riscos de ativar essa opção.

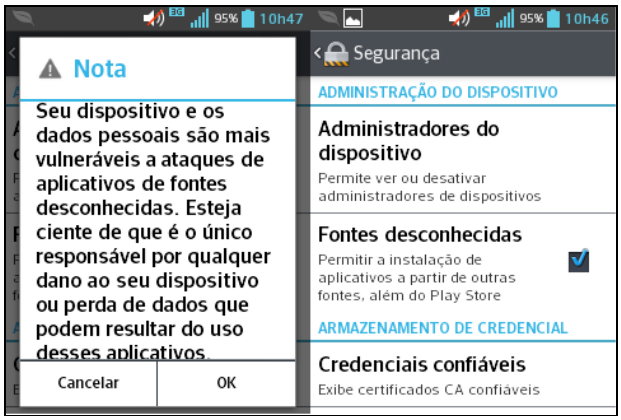


Figura 4 – Fontes Desconhecidas

Após essas configurações, o aplicativo escolhido para testes foi extraído do *smartphone* com a ferramenta ADB (*Android Debug Bridge*), que é parte do kit de desenvolvimento Android SDK. O ADB permite a

interação com o Android através de um terminal e fornece uma série de funcionalidades para análise de aplicativos e do próprio Sistema Operacional.

4.1.1 ESCOLHA DO APLICATIVO PARA TESTES

Para acessar e baixar aplicativos da Loja Google Play é necessário criar uma conta no Google. Após a criação da conta é possível ter acesso a todo o conteúdo da loja.

Devido às limitações do *smartphone* utilizado para os testes, alguns aplicativos não estavam disponíveis por exigirem um *hardware* mais avançado.

O aplicativo escolhido para testes foi o jogo Pou. O principal motivo para escolha deste aplicativo foi a sua popularidade, bem como o público-alvo, destinado principalmente para os mais jovens, e também por ser o primeiro aplicativo da lista da categoria "Principais Gratuitos", conforme a Figura 5. Este aplicativo possui mais de 100 milhões de downloads e aproximadamente 4 milhões de avaliações pelos usuários.

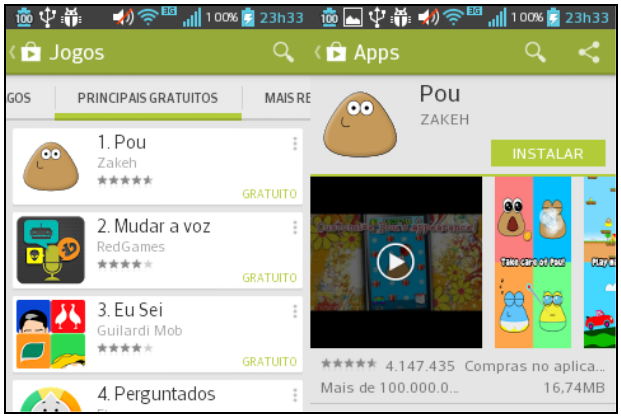


Figura 5 – Aplicativo Pou

Após o download e instalação do aplicativo no *smartphone*, foram obtidas algumas informações com a ferramenta Drozer. Este é um *framework* para testar a segurança de aplicativos e explorar vulnerabilidades da plataforma Android.

Na Figura 6 são mostradas as permissões de aplicativo necessárias para seu funcionamento.

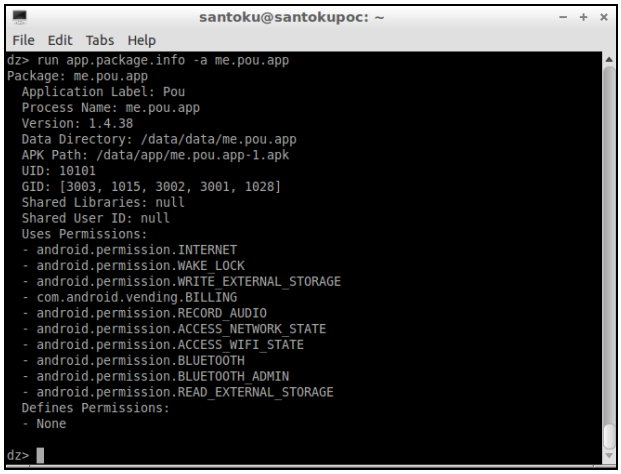


Figura 6 – Permissões do aplicativo Pou

E na Figura 7 são mostradas as principais *Activity*, *Broadcast Receiver*, *Content Providers* e *Services* do aplicativo.

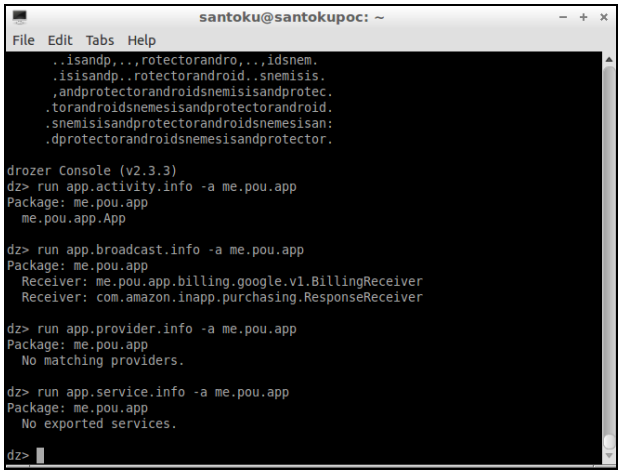


Figura 7 – *Activity, Broadcast Receiver, Content Providers* e *Services* do aplicativo Pou

Após obter essas informações, o aplicativo Pou foi submetido ao *framework* AFE.

Conforme citado anteriormente, o *framework* AFE permite a injeção de códigos maliciosos em aplicativos, sendo este o principal teste realizado no aplicativo Pou. A Figura 8 mostra a tela inicial do AFE.

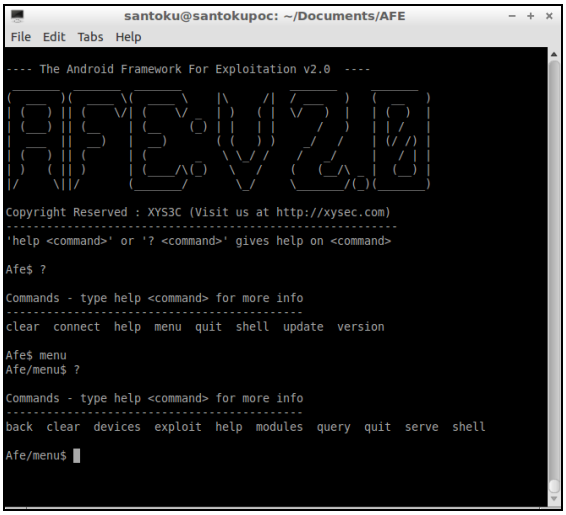


Figura 8 – Tela inicial AFE

Nessa tela também é possível ver as principais funcionalidades do AFE.

Após a execução do AFE, foram acessadas as opções "*menu>modules>list>run apk_inject*". Esta é a utilizada para a injeção de códigos maliciosos, conforme mostrado na Figura 9.

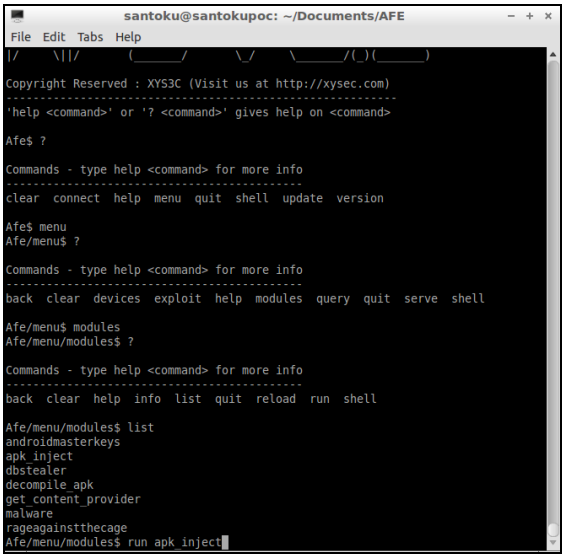


Figura 9 – Opção de injetar códigos maliciosos

Ao executar a opção "*run apk_inject*", é solicitado o nome do aplicativo em que será injetado o código malicioso, conforme Figura 10.

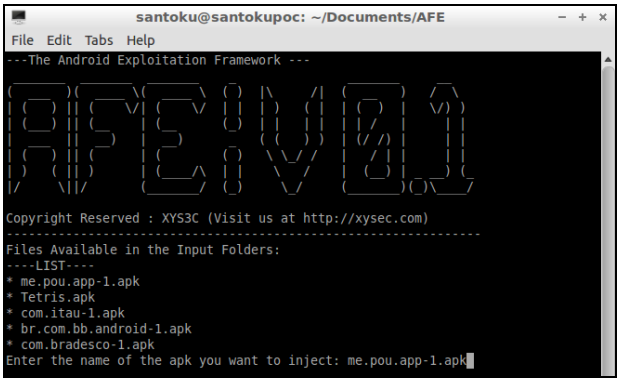


Figura 10 – Escolha do aplicativo para injetar códigos maliciosos

A Figura 11 mostra, de forma detalhada, cada etapa do processo de injeção de códigos maliciosos. No primeiro momento, o AFE realiza a engenharia reversa do aplicativo, obtendo o código fonte. Em seguida é injetado o código malicioso. A próxima etapa é adicionar as novas permissões de aplicativo. Após adicionar as novas permissões, são injetadas as *Services* e as *Activities* do código malicioso. Na etapa

final, o aplicativo é reconstruído e assinado, pronto para a instalação em qualquer *smartphone* com Android.

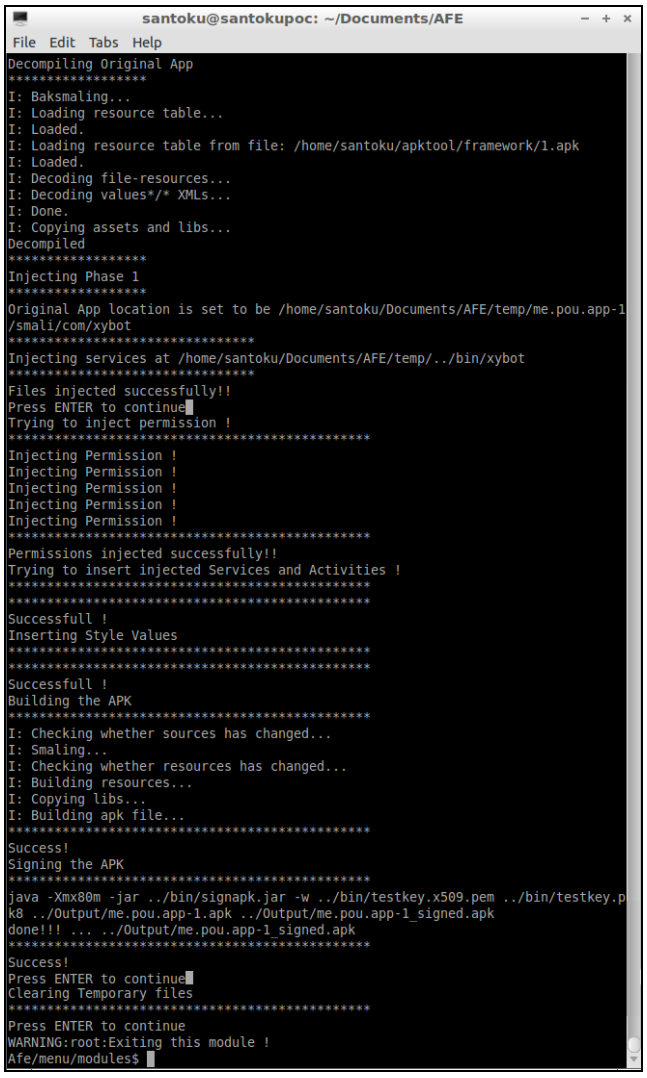


Figura 11 – Processo de injeção de códigos maliciosos

Após a execução do AFE, foram executados os mesmos comandos no aplicativo Pou, para se verificar o que foi alterado no aplicativo. Na Figura 12 é possível ver as novas permissões de aplicativo que foram adicionadas. Estas últimas permitem:

- *android.permission.RECEIVE_SMS* – receber mensagem SMS;
- *android.permission.READ_SMS* – ler mensagem SMS;
- *android.permission.WRITE_SMS* – escrever mensagem SMS;
- *android.permission.SEND_SMS* – enviar mensagem SMS;
- *android.permission.READ_CONTACTS* – ler contatos;
- *android.permission.READ_CALL_LOG* – ler registro de chamadas.

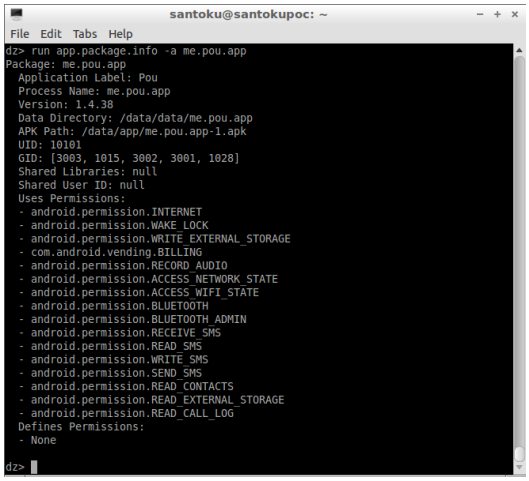


Figura 12 – Novas Permissões de Aplicativo

Outras alterações podem ser vistas na Figura 13. Foram adicionados pelo *framework* AFE ao aplicativo uma *Activity* e um *Broadcast Receiver*.

A *Activity* me.pou.app.XybotActivity é responsável por tratar os eventos gerados para o novo código injetado.

A *Broadcast* com.xybot.SMSReceiver é responsável por receber uma mensagem SMS e processá-la.

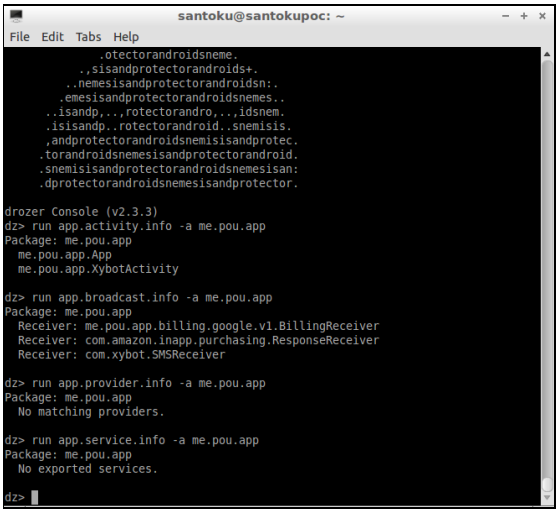


Figura 13 – Nova *Activity* e *Broadcast Receiver* do aplicativo Pou.

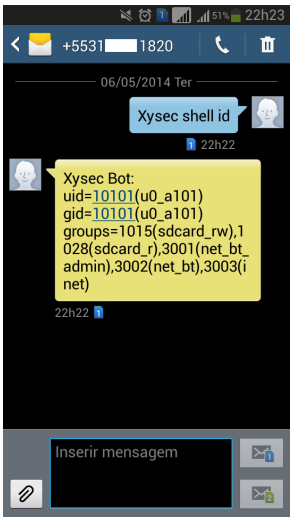


Figura 14 – Envio e resposta do comando solicitando o ID

4.1.2 CONTROLANDO O SMARTPHONE INFECTADO

O código malicioso injetado no aplicativo Pou é classificado como *Botnet*. Após ser injetado em um aplicativo e instalado em um *smartphone*, este tipo de *malware* fica aguardando instruções do servidor C & C (Comando e Controle).

Nos testes realizados, foram executados três comandos. O primeiro comando “Xysec Shell id” retorna o UID, GID e os grupos dos quais o aplicativo - em que foi injetado o código malicioso - faz parte, conforme mostrado na Figura 14.

O segundo comando “Xysec Shell cat /proc/version” retorna a versão do *Kernel* Linux e a do gcc (compilador padrão do Linux), conforme a Figura 15.

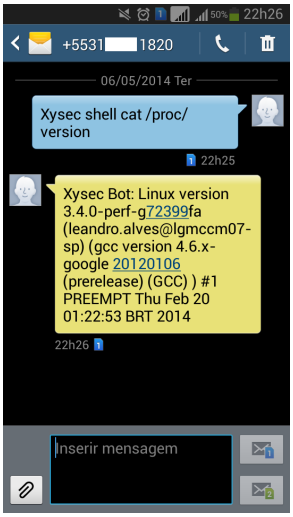


Figura 15 – Envio e resposta do comando solicitando a versão do *Kernel*

O terceiro comando “Xysec Shell ls /storage/external_SD/” lista todos os arquivos e pastas que estão armazenados no cartão SD, conforme Figura 16.

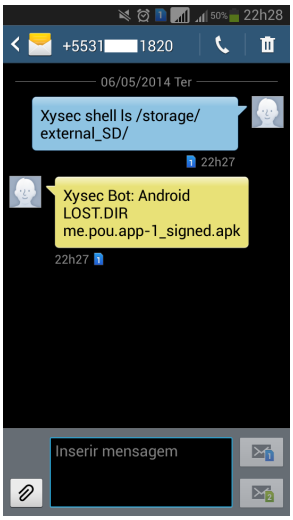


Figura16 – Envio e resposta do comando solicitando a listagem dos arquivos e pastas do cartão SD

4.2 TESTE NA BLACKMART

A BlackMart é uma loja de aplicativos não oficial para Android, que disponibiliza aplicativos compartilhados por seus usuários, conforme a Figura 17.

Figura 17 – Loja BlackMart

Essa loja possui um sistema semelhante à Google Play, dividindo os aplicativos por categorias e também classificando os mais utilizados, como mostra a Figura 18.

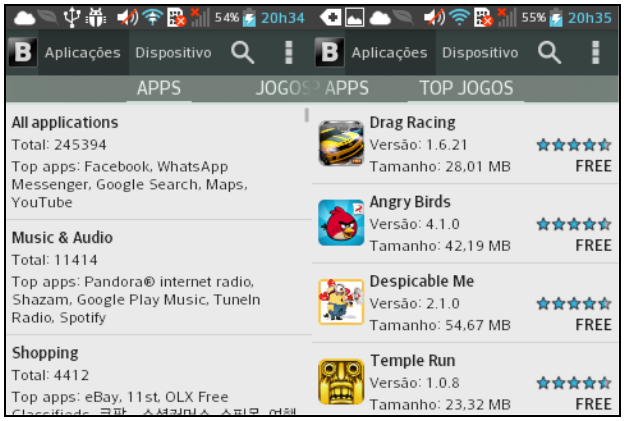


Figura 18 – Categorias de aplicativos da BlackMart

Em uma consulta rápida no Google pelo termo BlackMart é possível encontrar vários sites que disponibilizam este aplicativo. Nas pesquisas realizadas, a versão atual é a 0.99.2.52. O aplicativo BlackMart testado neste artigo foi obtido através de um link em uma página do Facebook que possui mais de 34 mil *likes*, na qual é feita também a sua divulgação. De acordo com a *Timeline* do Facebook, a página foi criada no dia 01 de Abril de 2011. É possível ver também nos *posts* a divulgação de novas versões do aplicativo; a última atualização ocorreu no dia 21 de abril de 2014.

Como descrito anteriormente, o aplicativo Pou foi infectado com um código malicioso que permite controlar o *smartphone* infectado através de mensagens SMS. Após este procedimento, o aplicativo Pou foi instalado no *smartphone* e houve a verificação se estava funcionando corretamente.

Ao instalar a BlackMart são verificados os aplicativos que estão instalados no *smartphone* e, caso o aplicativo não faça parte da base de dados da loja, é possível compartilhar o arquivo através da opção *upload*. O aplicativo Pou infectado não pode ser compartilhado por já estar disponível e por ser uma versão mais antiga. Neste caso é possível fazer a atualização do aplicativo, conforme mostra a Figura 19.

Figura 19 – Tela de Upload da BlackMart

Para continuar com os testes, foi escolhido o aplicativo Polaris Viewer 4, que não estava disponível na loja. Foram realizados os mesmos procedimentos feitos anteriormente com o aplicativo Pou.

Após infectar e instalar o aplicativo, foi realizado o Upload, conforme mostra a Figura 20.

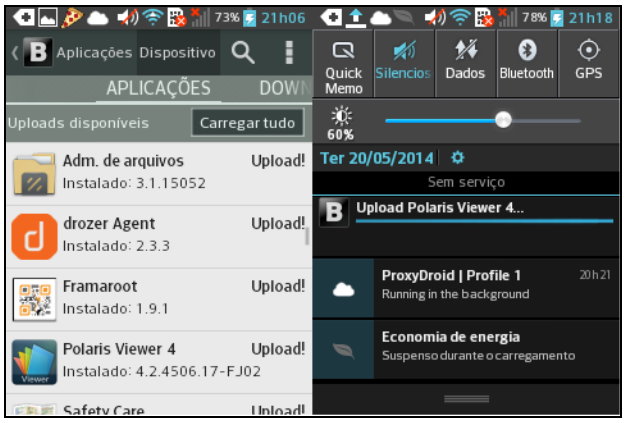


Figura 20 – Upload do aplicativo Polaris

Após o *upload* foi confirmado que o aplicativo foi enviado para a Blackmart, como mostra a Figura 21.

Figura 21 – Confirmação do *upload* do aplicativo

Para verificar se o aplicativo infectado estava disponível, foram realizadas algumas buscas pela loja, mas não foi possível encontrar o aplicativo. Após alguns dias, a opção de fazer o *Upload* do aplicativo estava disponível novamente.

A BlackMart não possui um sistema de *feedback* sobre as aplicações enviadas, assim não foi possível verificar se o aplicativo foi rejeitado por algum sistema de segurança da loja por ser um *malware* ou por algum outro filtro específico. Para verificar se o aplicativo foi rejeitado por ser um *malware*, aquele poderia ter sido submetido a uma ferramenta de otimização e ofuscação de código, como o Proguard, que dificultaria a detecção. Este teste se dará em pesquisas futuras.

4.3 Pesquisa com Usuários e Resultados

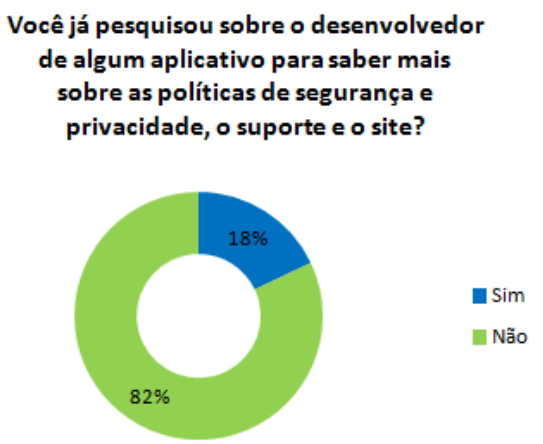
A pesquisa desenvolvida para verificar o nível de entendimento do usuário sobre a segurança do sistema operacional Android foi enviada principalmente para os alunos do UniBH. Foram contabilizadas 53 respostas e, após o fechamento do

formulário, foi feita uma filtragem das respostas, com o objetivo de se obter dados mais precisos, validando 39 respostas.

Conforme citado anteriormente, o Android se tornou o principal Sistema Operacional móvel, e também o principal alvo de ataques de *malwares*. Os resultados desta pesquisa trouxeram dados importantes sobre o comportamento dos usuários em relação a alguns quesitos de segurança.

De acordo com a pesquisa realizada, 82% dos usuários não pesquisam sobre o desenvolvedor do aplicativo que irão utilizar, conforme mostra o Gráfico 1.

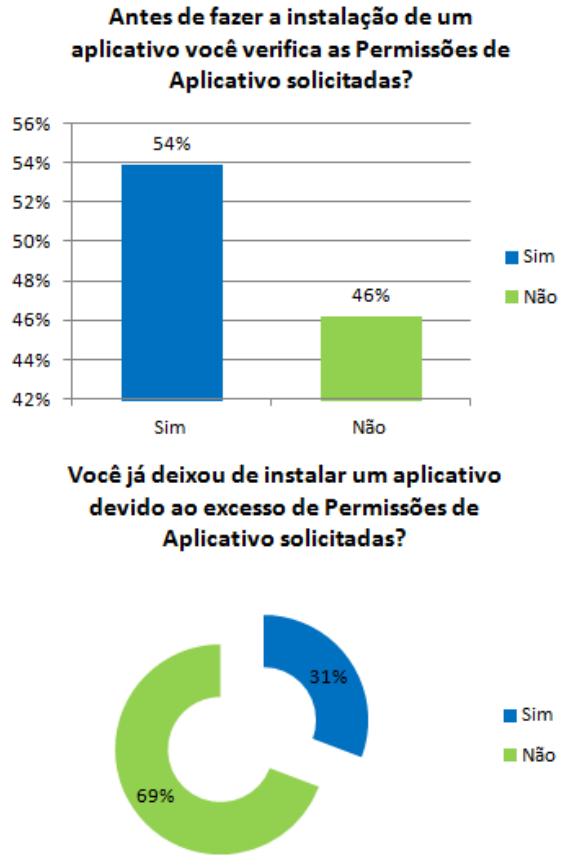
Gráfico 1 – Gráfico sobre Desenvolvedor de um aplicativo



Jogos populares na Google Play são plagiados com frequência. Um exemplo claro disso é o jogo Flappy Bird, que ficou conhecido mundialmente. Com o sucesso do jogo, surgiram outros milhares de jogos com nomes e jogabilidade similares, “criados” por outros desenvolvedores. Estes aplicativos, apesar de disponibilizados na loja oficial do Google, podem ter uma carga de código maliciosa para roubar dados pessoais ou até mesmo causar prejuízos financeiros aos usuários.

Outro dado importante mostrado no Gráfico 2 e que 46% dos usuários não verificam as Permissões de Aplicativos, portanto 69% instalam aplicativos com permissões que não condizem com as suas funcionalidades.

Gráfico 2 – Gráfico sobre Permissões de Aplicativos

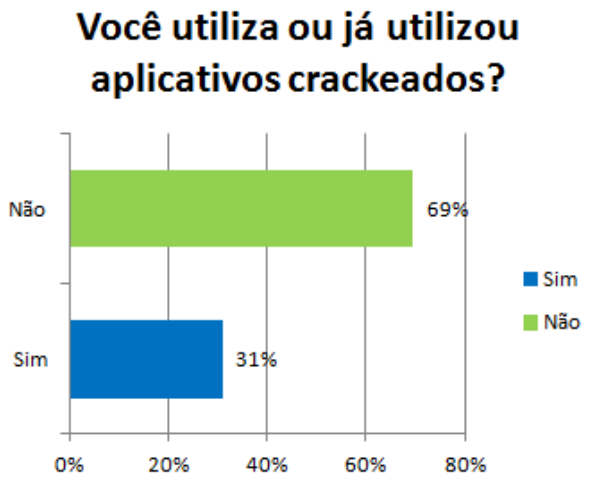


Um exemplo que ilustra muito bem o uso de indevido de Permissões de Aplicativos é o aplicativo Brightest Flashlight Free. Tal ferramenta é usada para ativar o flash da câmera e utilizá-lo como lanterna. Ao ser utilizada, coletava e compartilhava informações pessoais, como localização, sem o consentimento dos usuários. As informações capturadas eram compartilhadas com anunciantes e serviam para

definir quais propagandas apareceriam para o usuário. Após serem denunciados pela Comissão Federal de Comércio (FTC), os desenvolvedores do aplicativo alteraram os termos de uso, deixando claro que o aplicativo compartilha dados com terceiros (EXAME, 2013).

O uso de aplicativos crackeados (pirateados) é outro fator que compromete a segurança do Android. A pesquisa mostrou que 31% dos usuários utilizam ou já utlizaram algum aplicativo crackeado, conforme o Gráfico 3.

Gráfico 3 – Utilização de aplicativos crackeados



Um ponto importante a ser destacado é o preço dos aplicativos da Google Play, que varia em torno de 5 a 20 reais, salvo algumas exceções. Mesmo com o baixo custo dos aplicativos, alguns usuários acham que estão levando vantagem ao instalar aplicativos crackeados, mas, ao fazerem isso, estão abrindo as portas do seu *smartphone* para a entrada de *malwares*.

5 Resultados Obtidos

Ataques a aplicativos como os demonstrados neste trabalho são recorrentes e tendem a aumentar, uma vez que desenvolvedores e usuários não se preocupam com princípios básicos de segurança da informação. Técnicas como desenvolvimento seguro e proteção de código dificultam a engenharia reversa e mitigam ataques direcionados a aplicativos. Os usuários, por sua vez, estão preocupados com desempenho e design de seus *smartphones*, e esquecem que suas informações e dados pessoais estão sujeitas aos mesmos ataques realizados contra computadores.

Com o desenvolvimento dos objetivos propostos neste trabalho, foi possível se verificar que, para desenvolver uma campanha de *malware* para Android, não são necessários conhecimentos avançados, uma vez que desenvolvedores não protegem o código de seu aplicativo e os usuários não se preocupam com as Permissões de Aplicativos que são solicitadas ao instalarem um aplicativo. Os testes realizados neste artigo compravam a facilidade em se obter um aplicativo e injetar códigos maliciosos. Apesar de o aplicativo não ter sido publicado na BlackMart, técnicas como ofuscação de código poderiam ter sido utilizadas para dificultar a detecção por um sistema de segurança ou por algum outro filtro específico da loja.

Outro ponto importante observado foi a pirataria de aplicativos para Android. Mesmo com preços mais acessíveis - comparados a *software* desenvolvidos para computadores -, usuários ainda procuram formas de obterem o aplicativo de forma gratuita.

Com este trabalho foram identificadas também outras áreas de pesquisa em segurança da informação para a plataforma Android. Posteriormente a extensão deste trabalho será voltada à análise e à busca de vulnerabilidades em aplicativos bancários.

AGRADECIMENTOS

Os autores agradecem a todos que contribuíram para a realização deste trabalho.

REFERÊNCIAS

ANDROID AUTHORITY. **Almost 60 percent of active Android devices run Jelly Bean, KitKat registers small growth**. : Android Authority, 2014. Disponível em: http://www.androidauthority.com/android-version-breakdown-january-2014-334318/. Acesso em: 18 abr. 2014.

CASSANTI, M. O. **Crimes Virtuais, Vítimas reais**. 1.ª ed. Rio de Janeiro: Brasport, 2014. 136 p. ISBN: 978-85-7452-638-6.

CATHO. **Comunicação: uma necessidade humana**. São Paulo: CATHO, 2012. Disponível em: http://www.catho.com.br/carreira-sucesso/colunistas/comunicacao-uma-necessidade-humana. Acesso em: 20 ago. 2013.

EXAME. **App de lanterna para Android "roubava" dados de usuários**. São Paulo: EXAME, 2013. Disponível em: http://exame.abril.com.br/tecnologia/noticias/app-de-lanterna-para-android-roubava-dados-de-usuarios. Acesso em: 18 abr. 2014.

HOGLUND, G.; MCGRAW G. C**omo quebrar códigos:** a arte de explorar (e proteger) software. 1.ª ed. São Paulo: Pearson Makron Books, 2006. 424 p. ISBN: 85-346-1546-2.

IBM. **Entendendo segurança no Android**. São Paulo: IBM, 2010. Disponível em: http://www.ibm.com/developerworks/br/library/x-androidsecurity/. Acesso em: 20 ago. 2013.

IDC. **Estudo da IDC revela explosão de vendas de smartphones no Brasil no segundo trimestre de 2013**. São Paulo: IDC, 2013. Disponível em : http://br.idclatin.com/releases/news.aspx?id=1511. Acesso em: 20 set. 2013.

IDGNOW. **Geração Y lidera crescimento do mercado de smartphones**. São Paulo: IDGNOW!, 2013a. Disponível em: http://idgnow.uol.com.br/ti-pessoal/2013/06/06/geracao-y-lidera-crescimento-do-mercado-de-smartphones/. Acesso em: 30 ago. 2013

PORTAL TERRA. **Android foi alvo de 79% de todo o malware em 2012, diz estudo**. Tecnologia Terra, 2013. Disponível em: http://tecnologia.terra.com.br/internet/android-foi-alvo-de-79-de-todo-malware-em-2012-diz-estudo,f9c9d294bba4d310VgnVCM20000099cceb0aRCRD.html. Acesso em: 30 set. 2013.

IDGNOW. **Android pode chegar a 1 milhão de malwares este ano**. São Paulo: IDGNOW!, 2013b. Disponível em: http://idgnow.uol.com.br/internet/2013/01/24/android-pode-chegar-a-1-milhao-de-malwares-este-ano/. Acesso em: 01 set. 2013.

IDGNOW. **Cresce em 35% o número de malwares para Android, diz McAfee**. São Paulo: IDGNOW!, 2013c. Disponível em: http://idgnow.uol.com.br/mobilidade/2013/08/23/cresce-em-35-o-numero-de-malwares-para-android-diz-mcafee/. Acesso em: 02 set. 2013.

IPNEWS. **Metade das empresas não gerenciam dispositivos em redes corporativas**. São Paulo: IPNEWS, 2013. Disponível em: http://www.ipnews.com.br/telefoniaip/index.php?option=com_content&view=article&id=27843:metade-das-empresas-nao-gerenciam-dispositivos-em-redes-corporativas&catid=67:seguranca&Itemid=566. Acesso em: 24 set. 2013.

KASPERSKY. **Malware: Android tornou-se no equivalente ao Windows no mundo dos smartphones**. Portugal: Kaspersky, 2013. Disponível em: http://www.kaspersky.com/pt/about/news/virus/2013/Malware_Android_tornou-se_no_equivalente_ao_Windows_no_mundo_dos_smartphones. Acesso em: 01 set. 2013.

LECHETA, R. R. **Google Android:** Aprenda a criar aplicações para dispositivos móveis com o Android SDK. 3.ª ed. São Paulo: Novatec Editora Ltda., 2013. 821 p. ISBN: 978-85-7522-344-4.

MANN, I. **Engenharia Social**. 1.ª ed. São Paulo: Editora Edgard Blucher Ltda., 2011. 235 p. ISBN: 978-85-212-0604-0.

MITNICK, K. D; SIMON W. L. **A Arte de Enganar**. 1.ª ed. São Paulo: Pearson Makron Books, 2003. 304 p. ISBN: 85-346-1516-0.

PPLWARE. **Apenas 0,1% do Malware para Android está no Google Play**. Portugal: PPLWARE, 2014. Disponível em: http://pplware.sapo.pt/smartphones-tablets/android/apenas-01-do-malware-para-android-esta-no-google-play/. Acesso em: 18 abr. 2014.

SIX, J. **Segurança de aplicativos Android**. 1.ª ed.. São Paulo: Novatec Editora Ltda., 2012. 140 p. ISBN: 978-85-7522-313-0.

SOURCE ANDROID. **Android Security Overview**. Disponível em: http://source.android.com/devices/tech/security/. Acesso em: 20 out. 2013.

SYMANTEC. **Remote Access Tool Takes Aim with Android APK Binder**. Disponível em: http://www.symantec.com/connect/blogs/remote-access-tool-takes-aim-android-apk-binder. Acesso em: 20 out. 2013.

STALLINGS, W. **Criptografia e segurança de redes***:* Princípios e práticas. 4.ª ed. São Paulo: Pearson Prentice Hall, 2008. 492 p. ISBN: 978-85-7605-119-0.

TECMUNDO. **Número de aparelhos com Android ultrapassa a marca de 750 milhões**. Curitiba: TECMUNDO, 2013a. Disponível em: http://www.tecmundo.com.br/android/37559-numero-de-aparelhos-com-android-ultrapassa-a-marca-de-750-milhoes.htm. Acesso em: 25 set. 2013.

TECMUNDO. **7 mil malwares habitam as lojas virtuais alternativas do Android**. Curitiba: TECMUNDO, 2013b. Disponível em: http://www.tecmundo.com.br/malware/43832-7-mil-malwares-habitam-as-lojas-virtuais-alternativas-do-android.htm. Acesso em: 18 abr. 2014.

XYSEC. **Android Framework for Exploitation**. Xysec, 2012. Disponível em: http://xysec.com/afe_manual.pdf. Acesso em: 20 set. 2013.

ZERO HORA. **Vendas de smartphones crescem 86% no Brasil em um ano**. Porto Alegre: Zero Hora, 2013. Disponível em: http://zerohora.clicrbs.com.br/rs/economia/tecnologia/noticia/2013/06/vendas-de-smartphones-crescem-86-no-brasil-em-um-ano-4170047.html. Acesso em: 31 ago. 2013.