DOI: 10.5769/C2012003 or http://dx.doi.org/10.5769/C2012003

Análise de Segurança nos Processos de Sincronização e Autenticação de Aplicativos Android

Ricardo Kléber Martins Galvão
Grupo de Pesquisa em Segurança da Informação e Software Livre
Instituto Federal de Educação, Ciência e Tecnologia do RN
Natal/RN, Brazil
ricardokleber@ricardokleber.com
Marbel C. de O. Barbosa
Grupo de Pesquisa em Segurança da Informação e Software Livre
Instituto Federal de Educação, Ciência e Tecnologia do RN
Natal/RN, Brazil
marbel@segurancaderedes.org

***Abstract*—This research aims to analysis of processes of user authentication for access to Google and Twitter services through mobile applications for Android devices, verifying the occurrence of vulnerabilities that offer threatens the confidentiality and integrity of information that users rely their service providers. The motivation originates from the discovery of researchers at the Ulm University, about vulnerability in the authentication process to access the mobile device applications that use protocol ClientLogin, the user data stored in their services. However the research described here is not restricted to the protocol ClientLogin, but extends the analysis of the OAuth protocol, also used by some applications of mobile devices for authentication.**

Keywords-component; Android, Apps, ClientLogin, Oauth, vulnerability.

I. Introdução

Com o avanço das tecnologias relacionadas a dispositivos móveis, de fácil alcance e uso para qualquer tipo de usuário, atualmente tornou-se mais fácil compartilhar mensagens, fotos, músicas, ideias, estabelecer negócios, administrar a contas bancárias e fazer compras em qualquer parte do mundo. Eis uma era em que as pessoas não se intimidam mais com a barreira do tempo e do espaço. A informação é transmitida sem fronteiras e em tempo real, uma realidade que transpôs até, em muitos casos, as limitações clássicas relacionadas à diversidade de classes sociais.

De acordo com o levantamento da empresa IMS Research a venda de *smartphones* ultrapassou os 420 milhões de dispositivos em dezembro de 2011, respondendo por 28% do mercado mundial de celulares. A perspectiva é que até 2016 as vendas de smartphones excedam a marca de 1 bilhão de equipamentos, com grande variedade de aparelhos e preços acessíveis.

Não é de se admirar que com essa "epidemia" tecnológica da informação, haja também uma forte preocupação em relação à segurança dessas informações transmitidas. O extravio e/ou captura não autorizada de dados pessoais ou de informações sigilosas podem comprometer a privacidade e a integridade das informações, além da possibilidade de uso indevido e/ou mal-intencionado desses dados.

Esse artigo aborda o tema segurança da informação em aparelhos móveis (smartphones e tablets) que usam o sistema operacional Android, uma das plataformas que mais crescem no mercado, focando nos processos de autenticação dos equipamentos equipados com esse sistema operacional nos principais serviços disponíveis na Internet. Segundo o Instituto Gartner, o Android, dobrou sua presença no setor de vendas de dispositivos móveis: em 2010 dominava 25,3% do setor, e em 2011 já detinha 52,5% dele, graças aos 60,5 milhões de smartphones comercializados, ante os 20,5 milhões de 2010. A empresa Apple, apesar do aumento de suas vendas no período – de 13,5 milhões para 17,3% - diminuiu ligeiramente, de 16,6% para 15%. Com o sucesso do Android, a quantidade de ataques maliciosos ao sistema também aumentou, tornando ainda mais necessária a preocupação com a segurança das informações dos usuários, e aumentando a necessidade de tornar o sistema minimamente seguro, buscando evitar possíveis ataques.

Admitindo-se que não existe segurança absoluta, torna-se imprescindível perscrutar os mínimos detalhes do sistema, as possíveis brechas que permitam uma futura invasão. E assim, fazendo a avaliação dos riscos e impactos, e procurando rapidamente encontrar soluções para o problema.

A partir desse princípio, pesquisadores da Universidade de Ulm, da Alemanha, descobriram uma vulnerabilidade no serviço ClientLogin, protocolo de autenticação padrão para de contas de usuários do sistema operacional Android. Os aplicativos nativos do Google usam o protocolo ClientLogin, e durante o processo de autorização de acesso aos Serviço do Google, armazenam um *token* de autenticação, que é usado

durante as solicitações de acesso aos dados do serviços, mantendo esta informação armazenada durante determinado período para novos acessos. Esse processo é chamado de método programático e é utilizado para evitar o envio das informações de autenticação (nome de usuário e senha) a cada novo acesso.

O experimento consistiu em realizar análises nos acessos a aplicativos Google: Google Calendar, Google Contacts e Gallery (Picasa) utilizando as principais versões do Android.

A vulnerabilidade foi detectada na constatação de que as transmissões de *tokens* em alguns casos são realizadas através do uso do protocolo HTTP sem criptografia, possibilitando a captura desses *tokens* em redes abertas (como as redes *wireless*) durante o processo de autenticação podendo, a partir de então, ser utilizado por terceiros para o acesso aos dados do usuário.

Essa foi uma descoberta de fundamental importância posto que, segundo os pesquisadores, 99% dos dispositivos móveis Android (Todas as versões anteriores ao Android 2.3.4) estavam/estão sob ameaça.

A importância dessa descoberta sobre a vulnerabilidade no uso do protocolo ClientLogin, sem a utilização de criptografia, despertou o interesse por pesquisa complementar iniciada com experimentos visando responder a duas questões: As medidas cabíveis para a solução destes problemas/ vulnerabilidades já foram tomadas? Existem outras aplicações similares, além das apresentadas na citada pesquisa como vulneráveis, que também oferecem risco de segurança?

Esta foi a motivação para a pesquisa que resultou neste artigo: avaliar o status atual do protocolo ClientLogin monitorando e analisando durante o processo de sincronização, alguns dos aplicativos instalados (não se restringindo somente aos aplicativos nativos do Google) na versão Android 2.3.3, versão utilizada, ainda, pela maioria dos usuários de smartphones.

II. Fundamentação Teórica

A. A Plataforma Android

O Android é uma plataforma mantida por empresas, lideradas pelo Google, voltada a dispositivos móveis, tendo em sua composição o próprio sistema operacional, bibliotecas e *frameworks* de *middleware* complementados com aplicativos específicos. Esta plataforma, de código fonte aberto, foi anunciada oficialmente em novembro de 2007, juntamente com ferramentas e informações para o desenvolvimento de aplicações para dispositivos móveis, sendo baseada na plataforma Java e no sistema operacional Linux. Essa plataforma passou a ser mantida pela OHA (Open Handset Alliance), um grupo de várias empresas fabricantes de soluções de hardware e software para dispositivos móveis com o objetivo de oferecer padrões, inovar e acelerar o desenvolvimento de aplicações e serviços para dispositivos móveis.

B. Arquitetura

A camada da plataforma Android dependente de hardware tem como base o kernel do Linux 2.6, responsável pelas tarefas fundamentais do sistema operacional, como segurança, gerenciamento de memória, gerenciamento de processos, pilhas de protocolo de rede e modelos de drivers. Atua como uma camada de abstração entre o hardware e os outros componentes de plataforma, que executam em espaço de usuário.

Os componentes independentes de hardware constituem a maioria da plataforma Android, incluindo a biblioteca padrão C customizada, *codecs* para inúmeros formatos multimídia, um *engine* de browser (Webkit), ambiente gráfico e gerenciador de pacotes e a máquina virtual (Java) Dalvik.

C. Linux Kernel 2.6

O Android, embora baseado no Kernel do Linux, não pode ser considerado um sistema operacional Linux. O Kernel é usado diretamente pelo Android, além de várias modificações que foram realizadas de modo a otimizar memória e o tempo de processamento das aplicações. O Kernel do Linux foi escolhido em razão de possuir boa quantidade de drivers de dispositivos estáveis, como também bons sistemas de gerenciamento de memória e processamento. Acrescenta-se a isso, o fato do Kernel também oferecer serviços de segurança e rede, servindo também como camada de abstração entre software e hardware. Não é possível programar nesta camada do Android, que faz uso indireto do Kernel por meio de APIs *(Application Programming Interface* ou Interface de Programação de Aplicativos) de níveis superiores (Figura 1).

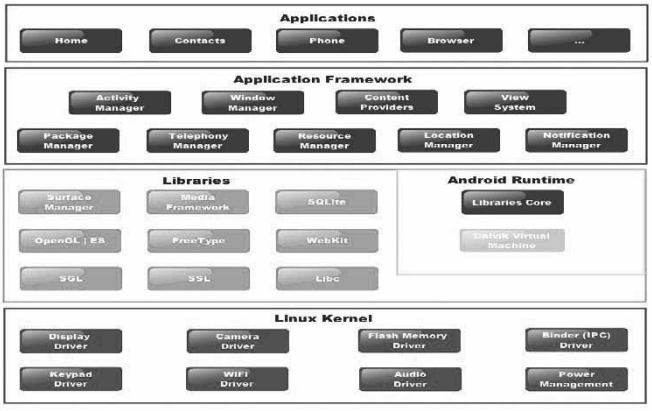

Figura 1 – Arquitetura da Plafatorma Android em Camadas (Fonte: developer.android.com)

D. Framework de Aplicação: Kit de Desenvolvimento de Software: SDK (Software Development Kit)

O Framework é uma base para as aplicações, onde desenvolvedores tem acesso completo. O framework do Android foi criado para facilitar o reuso dos componentes, bem como para abstrair parte dos procedimentos necessários para fazer uma aplicação funcionar. Assim, uma aplicação

pode capturar funcionalidades de outra criada anteriormente para o seu desenvolvimento. Nele estão inclusos:

- **Telephony manager:** gerência de hardware do dispositivo.
- **View system:** conjunto de telas (componentes gráficos) para desenvolver uma aplicação.
- **Content providers:** dados que são compartilhados entre várias aplicações, como por exemplo, a agenda.
- **Resource Manager**: administrador de recursos que permite acessar a recursos como Strings, gráficos, arquivos de layout.
- **Notification Manager:** administrador de notificações para mostrar alertas. As aplicações podem adicionar eventos numa barra de de notificações;
- **Activity Manager:** administrador de atividades. Este gerencia o ciclo de vida das aplicações e da navegação entre elas;
- **Location Manager**: serviço de localização. Permite o celular receber avisos, notificações, eventos, entre outros, de um lugar específico ou de sua localização atual;
- **Serviço XMPP.** Envio de mensagens para aplicações entre terminais Android. Pode ser utilizado entre jogos multiusuário, por exemplo.

E. Máquina Virtual Dalvik

A máquina virtual Dalvik é componente responsável pela execução dos softwares escritos em linguagem Java, compilando num formato especial de *bytecodes*, o .dex (Dalvik executable). Dalvik é uma máquina de registradores, em contraposição às tradicionais máquinas Java, que se utilizam do modelo de máquina de pilha, Dalvik é otimizada para sistemas com pouca memória, não usa *Just-in-time compilation* (método para melhorar a performance de execução programas de computadores) e não suporta arquivos .class como uma máquina virtual Java. Entretanto, a solução para esse problema é uma ferramenta (dx) que faz a ligação e converte vários arquivos .class para um único .dex.

F. Aplicações

As aplicações constituem a camada mais alta da arquitetura da plataforma Android, e contém uma série de aplicações chaves, que incluem: cliente de e-mail, SMS, mapas, calendário, browser, agenda de contatos, entre outros. As aplicações permitem embutir páginas de Web, rodam simultaneamente, além do usuário poder alternar entre as aplicações ativas. As aplicações não executam diretamente no Android, e sim, na máquina virtual Dalvik, em função dos aplicativos que são todos no formato Java. É devido a ferramenta do Dalvik que é possível converter o código Java em um arquivo .dex, possibilitando que o Android possa executar qualquer aplicativo Java.

III. Motivação da Pesquisa

A. Análise de Vulnerabilidade do Protocolo ClientLogin

Pesquisadores da Universidade de Ulm da Alemanha desenvolveram uma pesquisa que avaliou e questionou a segurança no processo de sincronização e autenticação de aplicativos Android, utilizando o serviço *Clientlogin*, através de redes wireless abertas. A análise foi feita a partir do princípio de funcionamento do *ClientLogin*, que é usado tanto por aplicativos de terceiros como aplicativos Android. Para toda aplicação que usa o API *ClientLogin*, há uma solicitação de um token de autenticação (*authToken*) a partir do serviço Google, transmitindo um nome de conta e senha através de uma conexão HTTPS. O *authToken* enviado como resposta pode ser usado para qualquer solicitação posterior pela API do serviço. Entretanto se caso o *authToken* for enviado através de HTTP não-encriptado, a captura dos *authTokens* por mal-intencionados pode ser feita com facilidade. E desse modo com o *Token* capturado é possivel acessar dados pessoais disponíveis pelo API do serviço.

Com base na suspeita de possível falha de segurança em aplicativos que usam *ClientLogin* os pesquisadores desenvolveram uma análise das seguintes versões do Android: 2.1 (Nexus One); 2,2 (HTC Desire, Nexus O*ne*); 2.2.1 (HTC Incredible S); 2.3.3 (Nexus One); 2.3.4 (HTC Desire, Nexus One); e 3.0 (Motorola XOOM).

Foram avaliados os seguintes aplicativos do Google: Google Calendar, Google Contacts, e Gallery. O resultado foi a constatação de que até a versão Android 2.3.3 os aplicativos Google Calendar e Google Contacts enviavam solicitações de autenticação e/ou sincronização via HTTP (sem criptografia) e, por isso, vulneráveis a ataques aos *authTokens*.

Analisando os aparelhos e sistemas operacionais em uso, identificou-se que o problema afetaria 99,7% de todos os smartphones Android (status de 02 de maio de 2011).

Apesar da correção realizada, já na versão 2.3.4, nos serviços descobertos como vulneráveis até a versão 2.3.3, quando passaram a utilizar conexões HTTPS no envio dos *tokens*, mesmo na nova versão do sistema operacional o aplicativo Gallery, que mantém a sincronização com os álbuns Web do Picasa, continuou enviando authTokens sem encriptação.

Um detalhe, que potencializa o risco de utilização de *tokens* capturados trafegando sem criptografia, tratado com detalhes adiante, é que os authTokens capturados expiram somente após 14 dias, o que habilitaria aos intrusos capturar e usá-los quando e quantas vezes quisessem por um considerável período de tempo.

Tabela 1 -*Uso de HTTPS no Android Google Apps.*

Android versão	Calendar Sync	Contacts Sync	Picasa Sync (Galeria)
3.0	sim	sim	?
2.3.4	sim	sim	**não**
2.3.3	**não**	**não**	**não**
2.2.1	**não**	**não**	n / a
2.2	**não**	**não**	n / a
2.1	**não**	**não**	n / a

(Fonte: ULM)

Como esclarecem Konings, Nickels & Schaub (2011) para coletar os authTokens em larga escala um intruso poderia configurar um ponto de acesso wifi com um SSID (Service Set IDentifier) comum (evil twin) de uma rede sem fio e sem criptografia, comum por exemplo em shoppings e áreas conhecidas como "WiFi Zone". Com as configurações-padrões, os dispositivos móveis Android se conectariam automaticamente a redes previamente conhecidas e muitos aplicativos tentariam sincronizar imediatamente. E quando a sincronização falhasse (a menos que o adversário encaminhasse os pedidos), o intruso capturaria authTokens de cada serviço que tentou sincronização. Mediante à longa duração das authTokens, seria fácil a possibilidade de se capturar um grande número de *tokens* e fazer uso deles, posteriormente, a partir de um local diferente.

B. Protocolo ClientLogin

O ClientLogin é um protocolo que funciona através de um método programático para obter acesso autorizado quando as aplicações instaladas no dispositivo móvel precisam trocar informações com os serviços do Google, e as mesmas são protegidas por um nome de usuário Google ou uma conta de aplicativo Google (hospedado).

Desse modo, os usuários podem fazer login em sua conta do Google e permitir acesso aos seus dados de serviços Google dentro de sua aplicação. Logo, o aplicativo contata o Google com os dados de login e solicitações de acesso a um serviço Google especificado. Uma vez que o acesso é autorizado, o aplicativo pode acessar os dados do serviço, permitindo ao usuário criar, ler, atualizar ou excluir dados do serviço conforme sua vontade, utilizando a interface do aplicativo.

O processo de autorização com o protocolo ClientLogin é um suporte a abordagem "low-tech", quando se é exigido que as aplicações incluam o nome de login do usuário e senha em cada pedido para um serviço do Google. Com login programático, o Google emite um *token* que pode ser referenciado em todos os pedidos subsequentes.

Admite-se assim que o desempenho é melhorado, tendo em vista, que os dados do login são validados apenas uma vez em cada sessão, ao invés de cada vez por solicitação. Diminui-se a quantidade de vezes em os dados do login são transmitidos por sessão, tornando, em tese, o processo mais seguro. É possível acrescentar medidas adicionais como o uso de CAPTCHAs, As medidas de autorização, em geral, podem ser facilmente melhoradas e ampliadas conforme a necessidade e importância das informações trafegadas.

Autorização com ClientLogin envolve uma sequência de interações entre três entidades: o aplicativo instalado, os serviços do Google, e o usuário. Como é ilustrado na Figura 2:

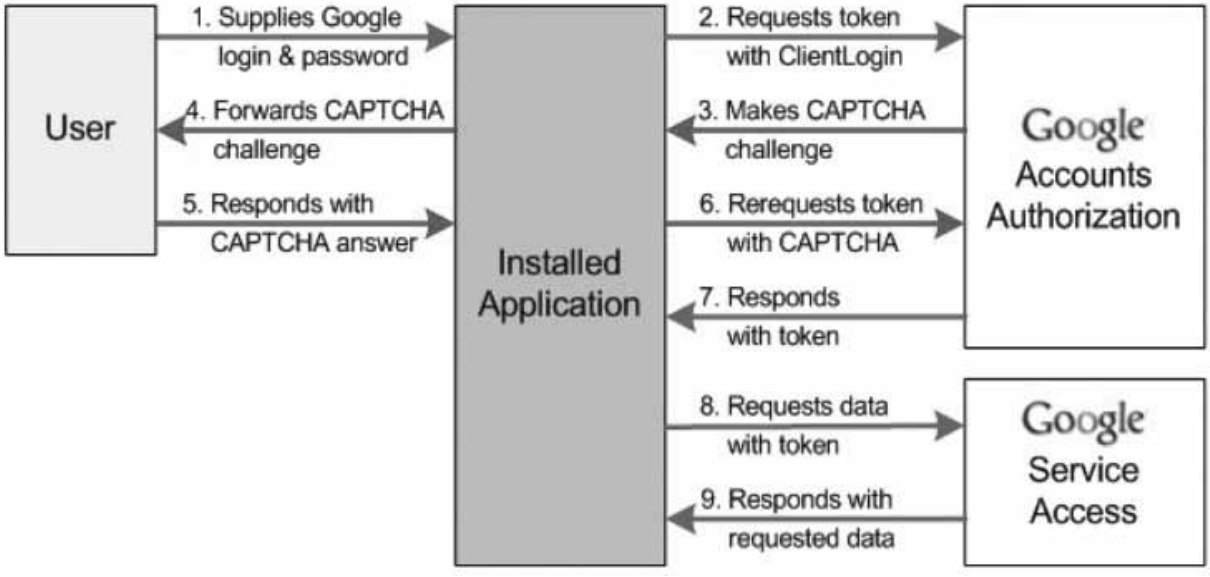

Figura 2 – Processo de autenticação entre o aplicativo e o Serviço Google (Fonte: Code Google)

- O aplicativo de terceiros faz uma chamada ClientLogin para o serviço Google de Autenticação.
- Se o serviço de autenticação Google decide que um veto adicional é necessário, ele retorna uma resposta de falha com um token CAPTCHA de desafio, na forma de uma URL para uma imagem CAPTCHA.
- Se um desafio CAPTCHA é recebido, o aplicativo de terceiros exibe a imagem CAPTCHA para o usuário e solicita uma resposta do usuário.
- Se solicitado, o usuário envia uma resposta ao desafio CAPTCHA.
- O aplicativo de terceiros faz uma nova chamada ClientLogin, desta vez incluindo a resposta CAPTCHA e o token (recebido com a resposta de falha).
- Em uma tentativa de login bem-sucedido (com ou sem desafio CAPTCHA), o serviço de autenticação Google retorna um token para o aplicativo.
- O aplicativo contata o serviço do Google com um pedido de acesso a dados, referenciando o token recebido do serviço de autorização Google.
- Se o serviço Google reconhece o token, ele fornece o acesso aos dados solicitados.

IV. Análise Adicional de Vulnerabilidades

A. Duração de Tokens

Segundo os desenvolvedores do Google o *token* de autenticação do ClientLogin pode durar **duas semanas** a partir da data de emissão, sendo o limite específico, de acordo com o serviço utilizado, podendo ser mais curto. Um dado bastante preocupante assumindo que os *tokens* expostos podem se capturados por qualquer intruso a qualquer momento em que usuário acesse os serviços Google, além de considerar que esses *tokens* podem ser usados confortavelmente para acessar os dados do usuário dentro desse período de tempo de duração.

O que se observa é que há uma relativa preocupação no que se trata de acesso não autorizado dos aplicativos aos dados do usuário, e que se sobrepõe a algo que não poderia deixar a desapercebido que é a questão da segurança no acesso à rede de dados (Internet, por exemplo), quando se trata de aplicativos para dispositivos móveis, tendo em vista a facilidade, que se sugere, da captura desses *tokens* de autenticação em tráfego.

O fato de não haver um tempo de expiração curto o suficiente que limite as possibilidades de um ataque, é um risco à integridade e privacidade das informações do usuário, além de considerar o descrédito em relação aos provedores de serviços, que devem estar sempre atentos às vulnerabilidades que expõem aos seus usuários.

Se já pode ser considerado preocupante o período de duração de um token na autenticação de serviços Google, constatou-se que no Twitter, serviço largamente utilizado para divulgação de informações e comunicação entre usuários de dispositivos móveis, os *tokens* **não possuem tempo de expiração**. Nesse serviço os *tokens* somente são considerados inválidos se o usuário explicitamente solicitar nas configurações do serviço ou se a administração do Twitter suspendê-lo.

B. Aplicações que Utilizam AuthTokens

Os principais aplicativos analisados, atualmente disponíveis para uso em dispositivos móveis Android, foram:

- **Google Calendar**: Calendário de Eventos/ compromissos (pessoais e/ou compartilhados);
- **Google Docs**: Repositório de documentos (pessoais e/ ou compartilhados);
- **Gmail**: Serviço de Webmail;
- **Google Contatos**: Gerenciador de contatos do usuário;
- **Picasa**: Aplicativo móvel de gerenciamento, edição e compartilhamento de fotos. (permite sincronização);
- **Twitter**: Aplicativo móvel sincronizável de acesso ao serviço do microblog Twitter.

C. Experimentos Complementares

Com vistas a repetir o experimento que fundamentou a pesquisa e realizar testes complementares em busca de novas vulnerabilidades, foi utilizado um aparelho Samsung Galaxy SII com sistema operacional Android 2.3.3, uma das versões que apresentava a vulnerabilidade no processo de autenticação dos aplicativos analisados, reportada pelos pesquisadores da Universidade de Ulm.

O ambiente de testes foi organizado de forma a permitir que os aplicativos sincronizassem com um ponto de acesso de uma rede *wireless* aberta e insegura (sem criptografia WEP ou WPA), simulando uma possível captura de um intruso (equipamento instalado entre o ponto de acesso e a rede externa realizando a captura de pacotes). Em seguida, foram monitoradas e capturadas as mensagens de conexão, solicitações e respostas durante o processo de autenticação entre o dispositivo e os serviços localizados na rede externa, por meio do software de *sniffer* **Tcpdump**. Cada arquivo de captura foi então, posteriormente, analisado no software de análise de pacotes de rede Wireshark de modo a identificar a forma como os tokens eram transmitidos no acesso a cada serviço.

O resultado foi a constatação de que, para a versão testada, os aplicativos do Google Contatos, Google Calendar, Google Docs, e Gmail já não mais apresentavam o *authToken* exposto, mas encaminhado em HTTP e encapsulado em túnel TLS (*Transport Layer Security*) garantindo a encriptação dos dados. O acesso ao serviço Picasa, porém, foi o único que ainda permanecia com a transmissão de *tokens* exposta, ou seja, ainda fazendo uso do protocolo HTTP sem criptografia para a transmissão dos dados de conexão e sincronização.

Foi observado, ainda, que o aplicativo utilizado para acesso ao serviço Twitter (UberSocial) utiliza/utilizou um outro processo de autenticação chamado **OAuth**. No uso

desse mecanismo, detalhado adiante, porém o *token* foi enviado sobre HTTP sem encriptação. constatação bastante preocupante tendo em vista a quantidade de usuários que acessam esse serviço. Analisando, em seguida, a configuração do aplicativo de acesso ao serviço, observou-se que existe/existia uma opção específica para ativar o uso de criptografia no processo de autenticação. Esta opção, porém, desabilitada por padrão, só permite o tráfego seguro de dados entre o dispositivo e o servidor do Twitter caso seja explicitamente marcada pelo usuário. Analisar outros softwares de acesso ao serviço Twitter fogem ao escopo dessa pesquisa e deverão ser objeto de pesquisa complementar do grupo.

D. Autenticação com OAuth

Embora semelhante aos mecanismos de autenticação dos serviços Google, o método de autenticação OAuth, utilizado pelo serviço Twitter, tem algumas características e etapas específicas detalhadas a seguir e ilustradas na Figura 3.

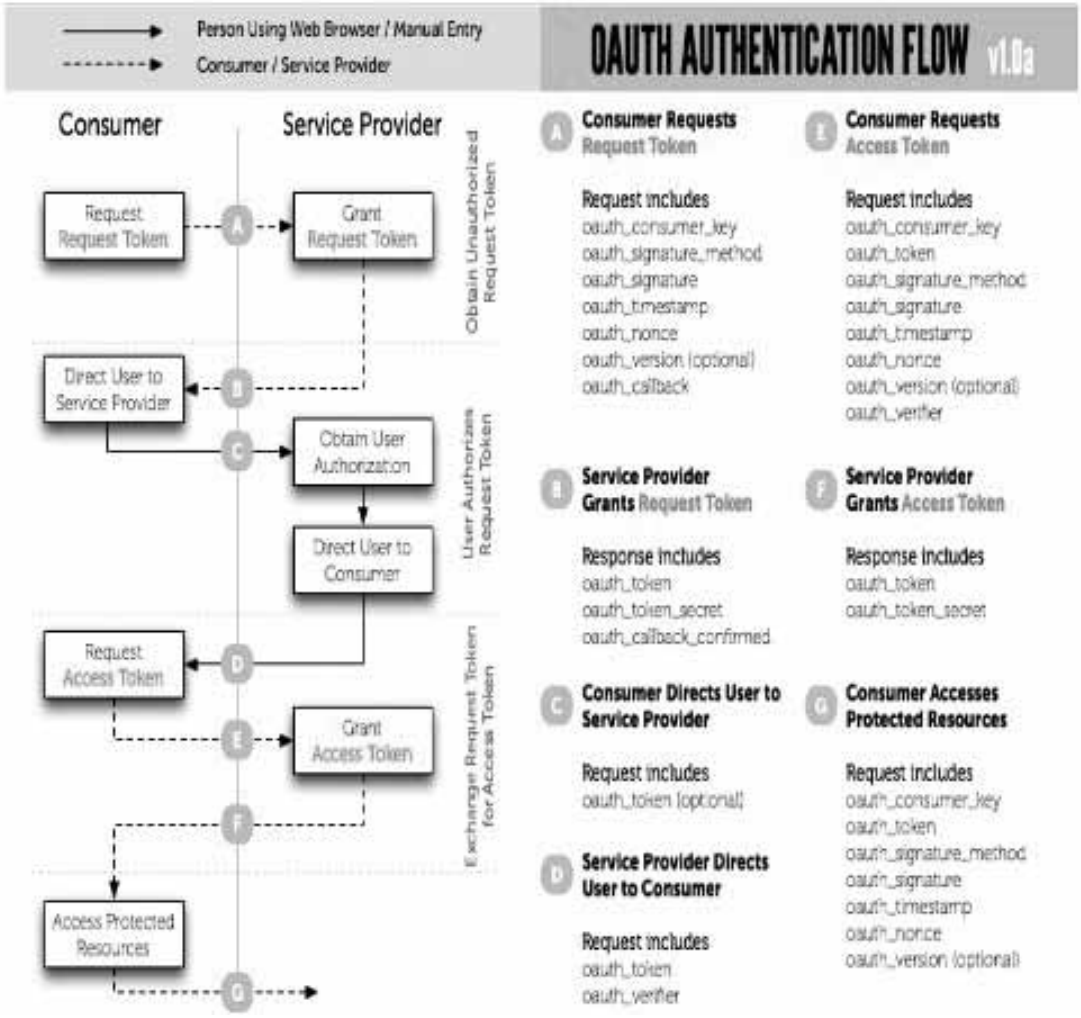

Figura 3 – Processo de Autenticação do OAuth (Fonte: www.oauth.core.net)

Tokens de solicitação: Utilizado pelo consumidor para solicitar ao usuário autorização de acesso aos recursos protegidos. O Token de solicitação de autorização do usuário é trocado por um *token* de acesso, devendo somente ser usado uma vez e sem outra finalidade.

Tokens de acesso: Utilizado pelo consumidor para acessar os recursos protegidos em nome do usuário. Os *tokens* de acesso podem limitar o acesso a certos recursos protegidos, e pode ter um "tempo de vida" limitado.

Cabe a responsabilidade aos Prestadores de serviços de assegurar que transmissões onde estão envolvidos os *tokens* sejam protegidos usando mecanismos da camada de transporte, como TLS ou SSL.

O protocolo OAuth não especifica nenhum mecanismo para proteger *tokens* e outros dados de análise pós-captura quando estes são transmitidos a partir do provedor de serviços para o consumidor. Isso significa que, como não há no mínimo o uso de algum protocolo de segurança da

camada de transporte, usuários mal intencionados podem ter acesso total às solicitações do OAuth e às assinaturas, tornando possível a realização de ataques de força bruta para a recuperação/descoberta das credenciais utilizadas pelo consumidor. Prestadores de serviços devem ter o cuidado de atribuir *tokens* e dados (segredos) do consumidor de modo que sejam longos e aleatórios o suficiente para resistir a esses ataques, pelo menos, durante o período de tempo que os segredos são válidos.

V. Conclusão

Analisando os aplicativos durante os processos de autenticação e sincronização em redes *wireless* abertas foi possível, a partir da reprodução do cenário de vulnerabilidades apresentado na pesquisa que fundamenta esse artigo, avaliar que alguns problemas de vulnerabilidades ainda persistem, fruto da falta de utilização de serviços baseados no uso de criptografia na transmissão de informações importantes (como é o caso das credenciais dos usuários e/ou *tokens* equivalentes).

Mesmo com a correção do problema diagnosticado até a versão 2.3.3 do sistema Operacional Android, passando a adotar o uso do criptografia na transmissão de *tokens* para praticamente todos os serviços do Google a partir da versão 2.3.4, o serviço Picasa permaneceu vulnerável, mesmo com o alerta apresentado pelos pesquisadores da Universidade de Ulm ainda no primeiro semestre de 2011, deixando, ainda, os usuários que mantém contas Google associadas ao Picasa passíveis de interceptação e sequestro de contas.

Também é fato que, embora a correção para os demais serviços Google tenha sido feita para versões posteriores à 2.3.3. do sistema operacional Android, ainda existem em uso (sem atualização) vários aparelhos com a versão vulnerável seja por falta de conhecimento de seus usuários sobre a necessidade de atualização para a proteção no uso dos serviços ou, ainda, pelas limitações tecnológicas dos aparelhos que, em alguns casos, pode impedir a atualização do sistema operacional, mesmo com a intenção por parte do usuário.

Em relação ao Twitter, mediante ao tamanho sucesso do microblog, é de extrema preocupação que o aplicativo também trafegue com *tokens* sobre HTTP sem encriptação, permitindo que intrusos sejam capazes de capturá-los em qualquer momento da transação da autenticação e possam reutilizá-los posteriormente já que contam com uma chave sem período de expiração, e assim deixando vulneráveis os dados do usuário a ataques. O uso do OAuth, por si só, não resolveu/resolve o problema. É necessário a obrigatoriedade (e não a opção) do uso de criptografia no transporte dos *tokens*, além da previsão de expiração desses tokens em um período razoavelmente curto.

Outras medidas que podem reduzir os riscos de vazamento de dados de usuários a partir de interceptação de dados críticos é a atualização, se possível, dos sistemas operacionais de *smartphones* e *tablets* para o Android 2.3.4 ou posterior; a desabilitação da sincronização automática dos aplicativos a serviços baseados nesses métodos de autenticação, sobretudo em redes abertas Wi-Fi; e o hábito de alteração de senha frequente por parte do usuário, de modo a evitar a reutilização de *tokens* em períodos muito longos.

Este artigo pode/deve ser complementado em um futuro próximo com um detalhamento nas pesquisas utilizando outros aplicativos de acesso ao serviço Twitter, além do que foi testado, bem como estendendo a análise a outros serviços com grande utilização na grande rede como o Facebook, Youtube, autenticação em portais baseados em CMS e outros eventuais serviços que se popularizem durante o trabalho do grupo de pesquisa.

VI. Referências Bibliográficas

[1] ClientLogin for Installed Applications. Disponível em: http://code.google.com/intl/pt-BR/apis/accounts/docs/AuthForInstalle dApps.html. Acesso em 01 de Julho de 2012.

[2] Bachiega, P. C.; Campos, D.; Flor, J. P. P.; Haensh, M. O. *Seminário de S.O - Google Android.* 2008.

[3] Catching AuthTokens in the Wild. The Insecurity of Google's ClientLogin Protocol. Disponível em: http://www.uni-ulm.de/en/in/mi/staff/koenings/catching-authtokens.html. Acesso em 01 de Julho de 2012.

[4] Global Smartphones Sales Will Top 420 Million Devices in 2011, Taking 28 Percent of all Handsets, According to IMS Research. Disponível em: http://imsresearch.com/press-release/Global_Smartphones_Sales_Will_Top_420_Million_Devices_in_2011_Taking_28_Percent_of_all_Handsets_According_to_IMS_Research. Acesso em 01 de Julho de 2012.

[5] Gartner Says Sales of Mobile Devices Grew 5.6 Percent in Third Quarter of 2011; Smartphone Sales Increased 42 Percent. Disponível em:http://www.gartner.com/it/page.jsp?id=1848514. Acesso em 01 de Julho de 2012.

[6] How long does an access token last? Disponível em: https://dev.twitter.com/docs/auth/oauth/faq. Acessado em 01 de Julho de 2012.

[7] Platforms Versions. Disponível em: http://developer.andro id.com/resources/dashboard/platform-versions.html. Acesso em 01 de Julho de 2012.

[8] Oauth Core 1.0. Disponível em :http://oauth.net/core/1.0. Acessado em 01 de Julho de 2012.

[9] What is Android?. Disponível em: http://developer.android.com/guide/basics/what-is-android.html. Acesso em 01 de Julho de 2012.