Análise de segurança em aplicativos bancários na plataforma Android

Rafael J. Cruz[1], **Diego F. Aranha**[1]

[1]Laboratório de Segurança e Criptografia (LASCA)
Instituto de Computação (IC) – Universidade Estadual de Campinas (Unicamp)
Av. Albert Einstein, 1251 – Campinas/SP – Brasil

{raju,dfaranha}@lasca.ic.unicamp.br

Abstract. *This paper presents the results of a security analysis on banking applications in the Android platform. The analysis included some aspects of the mobile application, the server configuration and the connection between app and server; and spanned 7 different Brazilian banks: Banco do Brasil, Bradesco, Caixa Econômica Federal, Citibank, HSBC, Itaú, and Santander. It was possible to mount a server impersonation attack in most banks, and obtain sensitive data, such as authentication credentials and financial information. Collected observations were not restricted to impersonation attacks, including server configuration flaws and questionable design choices, such as integration with social networks.*

Resumo. *Este trabalho apresenta os resultados de uma análise de segurança em aplicativos bancários na plataforma Android. A análise abrangeu alguns aspectos do aplicativo móvel, como a configuração do servidor e a conexão entre aplicativo e servidor. Os bancos analisados foram Banco do Brasil, Bradesco, Caixa Econômica Federal, Citibank, HSBC, Itaú e Santander. Foi possível montar um ataque de personificação do servidor com sucesso na maioria dos aplicativos e obter informações sigilosas, credenciais de autenticação e dados financeiros. As observações coletadas não se resumiram apenas aos ataques de personificação, mas também a falhas na configuração dos servidores e decisões de projeto questionáveis, como integração com redes sociais.*

1. Introdução

O protocolo SSL/TLS [Hirsch and Engelschall 2013] é o mais utilizado para se estabelecer conexões seguras via Internet, mas vários cuidados precisam ser tomados em sua implementação. Em especial, instituições financeiras – por sua natureza crítica – devem manter boas práticas de segurança, monitorar novos ataques, substituir algoritmos criptográficos obsoletos e implementar novas medidas de segurança. Isso é essencial para resistir a agentes maliciosos cada vez mais sofisticados. Os ataques bancários, por representarem uma atividade muito lucrativa, possuem intrinsecamente uma natureza adaptativa.

Segundo a Federação Brasileira de Bancos [FEBRABAN 2014], cerca de 24% (25 milhões) de contas utilizam *Mobile Banking*, ou seja, uma em cada quatro contas utiliza a tecnologia, que já representa 12% do número total de transações. Um pronunciamento da FEBRABAN sobre dicas de segurança eletrônica informou que os bancos brasileiros

tiveram uma perda de R$ 1,4 bilhão devido a fraudes em 2012 [FEBRABAN 2012]. Houve na ocasião uma certa comemoração, apesar do valor expressivo informado, já que o valor representa menos de 0,007% das transações bancárias.

2. Fundamentação Teórica

Nesta seção discute-se o protocolo SSL/TLS, a resolução da camada física, o ataque Homem No Meio e práticas recomendadas de segurança.

2.1. Protocolo SSL/TLS

Em 1994, a empresa Netscape reconheceu a necessidade de se estabelecer uma conexão segura para realizar comércio eletrônico e projetou a primeira versão do protocolo criptográfico SSL (*Secure Sockets Layer 1.0*). Este protocolo foi concebido com o objetivo de fornecer as propriedades clássicas da Segurança da Informação:

- **Confidencialidade:** é a garantia do sigilo das informações fornecidas e proteção contra sua revelação não autorizada.
- **Integridade:** significa ter a disponibilidade de informações confiáveis, corretas e em formato compatível com o de utilização. Em outras palavras, que a informação não foi alterada de forma indevida.
- **Disponibilidade:** refere-se a resistência a falhas e manutenção do serviço pelo máximo de tempo possível. No contexto de segurança, há ênfase em resistir não apenas a falhas e acidentes, mas também a ataques de negação de serviço [Handley and Rescorla 2006].
- **Autenticidade:** consiste na identificação e a confirmação de origem da informação.
- **Irretratabilidade:** propriedade do que não pode ser rejeitado e tratado novamente. É frequentemente associada à natureza irrevogável de assinaturas digitais.

Após várias mudanças no protocolo SSL, o navegador Netscape lançou a versão 3.0 em 1996. Entretanto, uma contribuição mais fundamental seria essencial para o desenvolvimento do protocolo. Em 1999 surgiu o sucessor do protocolo SSL: o protocolo TLS 1.0 (*Transport Layer Security*) – que agora é controlado por uma comunidade aberta, a *IETF – Internet Engineering Task Force*.

2.1.1. Descrição do protocolo

O SSL/TLS funciona na camada de aplicação do modelo TCP – ponto onde são processados dados para envio – e acima da camada de transporte – local onde mensagens são enviadas. Qualquer aplicação que faça uso do protocolo SSL/TLS fica isolada da camada de transporte, ou seja, suas mensagens passam primeiro pelo SSL/TLS, antes de prosseguir para a camada de transporte.

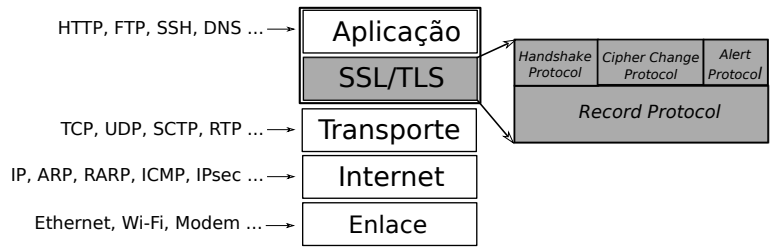

Figura 1. Modelo TCP com SSL/TLS sobre a camada de aplicação.

Dentro do protocolo SSL/TLS há quatro subprotocolos:

- ***Handshake Protocol*:** negocia as informações de sessão entre o cliente e o servidor. As informações de sessão consistem na identificação da sessão, troca de certificados, conjunto de algoritmos a serem utilizados, algoritmos de compressão e um segredo compartilhado usado para gerar chaves;
- ***Cipher Change Protocol*:** troca dados para gerar chaves criptográficas entre cliente e servidor. O subprotocolo consiste de um única mensagem enviada para a outra entidade da sessão SSL/TLS, o emissor deseja utilizar um certo conjunto de chaves. A chave é computada das informações trocadas pelo *Handshake Protocol*;
- ***Alert Protocol*:** indica mudanças no estado ou condição de erro entre as entidades envovildas. Os alertas são comuns quando a conexão é encerrada, uma mensagem inválida é recebida, a mensagem não pode ser decifrada, ou o usúario cancela a operação;
- ***Record Protocol*:** faz a segurança e verificação dos dados da aplicação, usando as chaves criptográficas derivadas durante a etapa de *handshake*.

2.1.2. *Handshake*

No protocolo SSL/TLS, a etapa de *handshake* [Moser 2009] especifica como se dá a troca de mensagens no início da conexão, onde o cliente faz um pedido de conexão ao servidor e obtém uma resposta informando que o servidor está pronto para a conexão. Em uma conexão segura, a troca de mensagens é bem mais intensa do que em uma conexão comum. Isso porque, além do cliente e do servidor aceitarem a conexão um do outro, devem também combinar uma maneira segura para se comunicarem. Durante esta troca de mensagens do *handshake* seguro, há a opção de utilizar a função de certificação digital, que é um modelo onde é possível comprovar por terceiros – Autoridade Certificadora [Instituto Nacional de Tecnologia da Informação (ITI) 2014] – que tanto o servidor quanto opcionalmente o cliente, são realmente quem dizem ser. Além da autenticação, é também possível negociar quais algoritmos criptográficos serão utilizados posteriormente, durante a troca efetiva de dados.

2.1.3. Algumas vulnerabilidades do SSL/TLS

Nem mesmo a versão mais recente do protocolo (1.2, publicada em 2008) está imune aos ataques, como sugere o *Triple Handshake Attack* [Bhargavan et al. 2014]. Grande parte dos ataques no protocolo SSL/TLS buscam brechas na implementação ou na configuração do protocolo SSL/TLS. Dentre eles, destacam-se:

1. O ataque **BEAST** [Sarkar and Fitzgerald] explora escolha não aleatória do vetor de inicialização no modo de operação CBC de uma cifra de bloco;
2. Os ataques **CRIME** e **BREACH** [Sarkar and Fitzgerald] analisam a compressão do protocolo SSL/TLS para capturar informações. A compressão torna alguns *bytes* previsíveis;
3. O **TIME** [Sarkar and Fitzgerald] analisa diferenças no tempo de envio e recebimento de mensagens comprimidas;

4. O **LUCKY 13** [Sarkar and Fitzgerald] analisa variações no tempo de verificação de pacotes autenticados contendo 13 *bytes* do cabeçalho;
5. No **Ataque a Cifra RC4** [Sarkar and Fitzgerald], um atacante ativo pode forçar um cliente estabelecer ou renegociar diversas vezes as chaves de sessão até que um mesmo texto seja cifrado com chaves RC4 diferentes;
6. O **FREAK** [CVE 2014] opera sobre a negociação de chaves entre o cliente e o servidor, forçando a utilização de uma chave RSA de apenas 512 *bits*. Um ataque similar no acordo de chaves Diffie-Hellman é chamado **Logjam** [Adrian et al. 2015].
7. O **POODLE** [CVE 2014] opera sobre o preenchimento inadequado do cabeçalho dos protocolos SSL 3.0 e TLS 1.0. O ataque necessita de 256 requisições para recuperar um *byte* de mensagem cifrada.

2.2. Resolução na camada física

O Protocolo de Resolução de Endereço (ARP – *Address Resolution Protocol*) [Plummer 1982] é utilizado para encontrar o endereço físico da camada de enlace (endereço MAC no protocolo *Ethernet*) a partir do endereço da camada de rede local (endereço IP). O ARP contém dois métodos de resolução de nomes:

(i) o emissor difunde em *broadcast* – envio de mensagem para todos na rede local – um pacote ARP, contendo o endereço IP do outro *host* e espera uma resposta com o respectivo endereço MAC;
(ii) o emissor difunde em *broadcast* seu MAC com seu respectivo IP.

O mapeamento obtido durante a resolução é armazenado em uma tabela de mapeamento em *cache*, para reduzir a latência e carga na rede caso as mesmas resoluções sejam realizadas no futuro próximo. Esta tabela contém um tempo limite de armazenamento e, se algum *host* solicitado tiver sua entrada na tabela expirada, a resolução é repetida.

2.3. Ataque De Homem No Meio - MITM

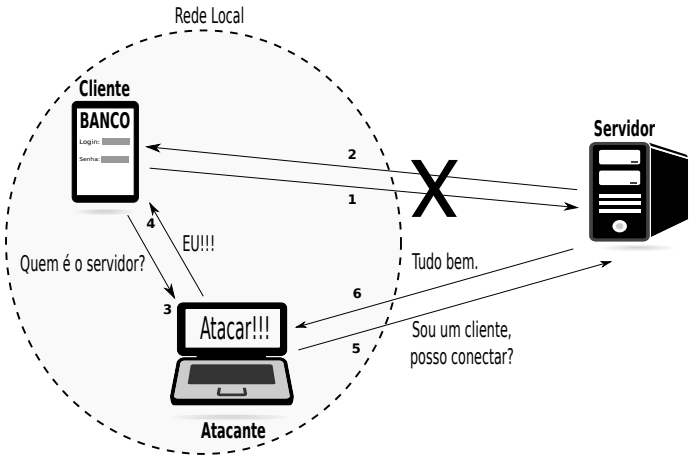

Figura 2. Ataque *Man In The Middle* dentro de uma rede local.

O ataque Homem No Meio (do inglês *Man In The Middle – MITM*) é uma forma de ataque em que os dados trocados entre duas partes são interceptados, registrados e possivelmente alterados sem que as vítimas percebam. O ataque pode ser tentado por qualquer agente intermediário em uma conexão (roteador, *proxy* ou servidor DNS, por

exemplo), mas no contexto desse trabalho, assume-se que o ataque é realizado na mesma rede local em que encontra-se o cliente. Desta forma, o agente malicioso é capaz de personificar o servidor apresentando um certificado falso para interceptar o conteúdo de uma conexão SSL/TLS. Para isso, basta que o atacante seja o administrador da rede ou forje a resolução do ARP. No caso de forjar o ARP, o atacante deve direcionar tanto o tráfego do cliente quanto do servidor para a máquina maliciosa, forjando a resolução ARP de maneira bidirecional.

A Figura 2 apresenta o funcionamento geral do ataque e pode ser detalhada da seguinte forma:

- Mensagens **1** e **2**: referem-se à conexão normal entre o servidor e o cliente, sem influência maliciosa;
- Mensagem **3**: o agente malicioso percebe que o cliente deseja entrar em contato com o servidor alvo;
- Mensagem **4**: o agente malicioso responde prontamente para o cliente, personificando o servidor. Com isso, a falsa conexão cliente-servidor está estabelecida entre o cliente e o atacante;
- Mensagem **5**: o agente malicioso finge ser o cliente para o servidor e abre uma nova conexão;
- Mensagem **6**: o servidor aceita a conexão, e ao finalizar o passo 6, o atacante detém toda a informação para realizar o ataque com sucesso.

A prevenção desse tipo de ataque é realizada com uma boa implementação do protocolo SSL/TLS, pois suas propriedades de segurança estabelecem segurança fim-a-fim, evitando influência indevida por máquinas intermediárias.

Ataque MITM com Certificado Autoassinado

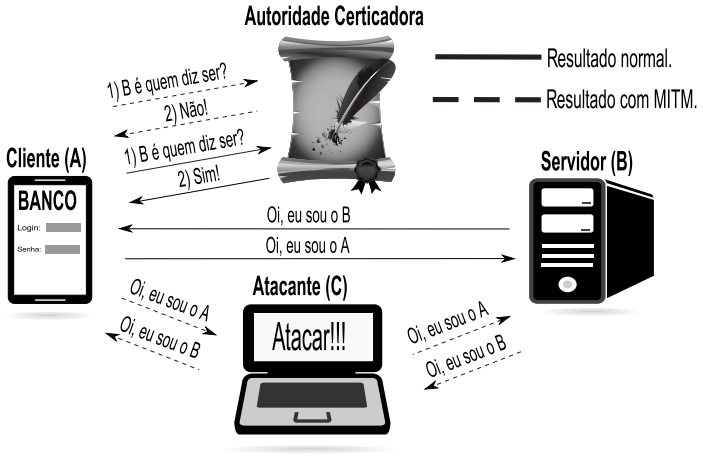

Figura 3. Propriedade de autenticidade do SSL/TLS, supondo que o cliente não possui certificado digital.

É necessário analisar a Figura 3 para compreender o funcionamento da variante deste ataque. Observe que a conexão faz uso da autoridade certificadora descrita durante o *handshake*. Com isso, é possível concluir que a verificação cuidadosa do certificado invalida o ataque MITM.

Um agente malicioso pode ser capaz de convencer o usuário a instalar um certificado raiz. Este fator é o que torna o ataque possível, já que o certificado instalado permite a validação no resultado com MITM da Figura 3.

Para evitar este problema, as boas práticas de segurança determinam então, que o aplicativo seja capaz de detectar a validação indevida do certificado falso, a partir da implementação de uma técnica de pinagem de chave pública, ou seja, qualquer informação relevante sobre os certificados do servidor armazenada no cliente, por exemplo: a autoridade certificadora ou *hash* da cadeia de certificados.

2.4. Práticas recomendadas

Para adquirir um bom nível de segurança ao implementar o protocolo SSL/TLS, servidores devem realizar os seguintes procedimentos:

- **Suporte a Segredo Futuro [Kaliski 1999]:** é a propriedade em que uma comunicação presente é confidencial, e espera-se que ela se mantenha confidencial no futuro, mesmo que as chaves de longa duração de um ou de outro (chave privada correspondente ao certificado do servidor, usuário/senha do cliente) sejam comprometidas. Alternativamente, se uma única comunicação for comprometida, não há impacto na confidencialidade de todas as comunicações anteriores a ela. Uma maneira de realizar uma implementação do Segredo Futuro, é gerar chaves de sessão – chave utilizada após o *handshake* – de maneira aleatória.
- **Atualização de algoritmos:** remover o suporte a algoritmos criptográficos inseguros. Algumas funções de *hash* ultrapassadas, como MD5 [Rivest 1992] e SHA-1 [Eastlake and Hansen 2011], devem ser atualizadas para algoritmos mais seguros (SHA-2 [Eastlake and Hansen 2011], por exemplo), e deixar de utilizar algoritmos de cifração como RC4, que permite recuperação trivial da chave de sessão utilizando o ataque de criptograma escolhido.
- **Validação de Certificados:** obter um certificado adequado e utilizá-lo coerentemente no protocolo é de suma importância. Uma boa cadeia de certificação gerada pelas autoridades certificadoras (VeriSign, Equifax, Certisign, entre outros), garante o início de uma boa segurança, o que significa utilizar certificados confiáveis. Fornecer uma lista de certificados revogados (CRL [Housley et al. 2002]) e um protocolo de verificação do estado do certificado (OCSP [Santesson et al. 2013]) auxiliam na verificação da validade do certificado.
- **Atualização de versão de protocolo:** abandonar protocolos obsoletos, como SSL 3.0. Protocolos antigos e mal implementados deixam o servidor vulnerável a ataques como POODLE [CVE 2014], FREAK [CVE 2014] e DoS [Handley and Rescorla 2006], vulnerável à ataques de deterioração (que reduzem a capacidade do *hardware* ou *software* – neste caso reduzem a segurança dos algoritmos criptográficos, o que permite o FREAK, por exemplo), má implementação do Diffie-Hellman [Rescorla 1999], entre outros. Ainda no protocolo, é possível fazer uso do HTTP *Public Key Pinning* (HPKP) [Evans et al. 2015] e OCSP [Santesson et al. 2013].

A responsabilidade do cliente envolve prevenir configurações inseguras do servidor quando possível e, especialmente:

- **Validação cuidadosa de chave pública:** os clientes devem avaliar cuidadosamente as chaves públicas do servidor trocadas durante a conexão, e portanto, uma mínima verificação do certificado é necessária, para comprovar sua autenticidade – consultando os certificados das autoridades certificadoras instaladas nos dispositivos, por exemplo. Há ainda, a possibilidade de verificar a expiração do certificado, através da sua lista CRL e do seu OCSP. O cliente também deve realizar a pinagem no certificado. Caso o servidor não tenha suporte a pinagem, através da extensão HPKP, a verificação é feita por alguma informação instalada no próprio cliente. O artigo [Georgiev et al. 2012] contém vários exemplos, obtidos de algumas bibliotecas da Amazon e Paypal, de verificação insuficiente da chave pública por parte do cliente. Isto reforça ainda mais o cuidado que aplicações críticas devem ter ao realizar a verificação do certificado.

3. Metodologia

A metodologia consistiu em três grandes partes:

1. Análise dos códigos e identificação dos servidores de cada aplicação bancária:
   - Capturar os *hostnames* de servidores bancários, com auxílio da ferramenta `Wireshark` [Wireshark Project]. Alguns bancos contém mais de um servidor atendendo o aplicativo.
   - Baixar os aplicativos bancários no formato `.apk`, utilizando `APK Downloader` [Redphx].
   - Descompilar os aplicativos com as ferramentas `dex2jar` [Pan], que transforma o `.apk` para o `.jar`, e `JD-GUI` [Java Decompiler], que transforma o `.jar` para arquivos em `.java`.
   - Utilizar os aplicativos bancários com o `Monitor-SDK` [Google, Inc.], que mostra alguns *logs* do aplicativo conforme há iteração no aparelho.
2. Ataque Homem no Meio com e sem certificado raiz forjado instalado:
   - Redirecionar o tráfego de rede sem fio para o computador malicioso. No caso do sistema operacional Linux basta alterar `iptables`.
   - Na rede local, controlar a rota do cliente através de um *proxy* malicioso ou ataque na resolução ARP com a ferramenta `arpspoof` [Song].
   - Gerar um certificado com auxílio do `OpenSSL` [The OpenSSL Project].
   - Controlar o tráfego SSL/TLS de forma transparente, através da ferramenta `sslsplit` [Roethlisberger]. O `sslsplit` utiliza o certificado forjado para trocar informações na conexão SSL/TLS. Opcionalmente instalar o certificado forjado no *smartphone* (ataque MITM com certificado raiz).
3. Exame dos servidores identificados:
   - Exames semanais, por três meses, das configurações dos servidores através do `SSLlabs` [Qualys, Inc.].

4. Resultados e discussão

Seguindo a metodologia proposta, podemos utilizar os resultados semanais do SSLlabs para avaliar os servidores na Tabela 1. As notas são atribuídas pelo SSLlabs.

Em particular, o Bradesco contém dois servidores para atender a aplicação móvel. O HSBC é um aplicativo global (também funciona fora do Brasil), por este motivo contém

três servidores para atender a aplicação móvel. Em uma análise mais profunda foi possível verificar que há um servidor para os membros (Nota C), um servidor para serviços (Nota A-) e um servidor nacional para transações (Nota A-).

Tabela 1. Resultado do SSLlabs sobre os servidores examinados. As notas variam de F a A+.

	Banco do Brasil	Bradesco	Caixa Econômica Federal	Citibank	HSBC	Itaú	Santander
☺ Emprega TLS 1.2	✘	✘	✔	✘	✔	✔	✘
☺ Emprega TLS 1.1	✘	✘	✘	✘	✔	✔	✘
😐 Emprega TLS 1.0	✔	✔	✔	✔	✔	✔	✔
☹ Suporta SSL 3.0	✔	✔	✔	✘	✔	✔	✔
☹ Suporta SSL 2.0	✘	✔	✘	✘	✘	✘	✘
☹ Suporta RC4	✔	✔	✔	✘	✘	✔	✔
☹ Suporta MD5	✔	✔	✘	✘	✘	✘	✘
😐 Suporta SHA-1	✔	✔	✔	✔	✔	✔	✔
☺ Suporta Segredo Futuro	✘	✘	✘	✘	✘	✘	✘
☺ Suporta OCSP	✘	✘	✘	✘	✔	✘	✘
☺ Suporta HPKP	✘	✘	✘	✘	✘	✘	✘
☹ Diffie-Hellman Inseguro	✔	✘	✘	✘	✘	✘	✘
☹ Vulnerável a Deterioração	✔	✔	✘	✘	✔	✔	✔
☹ Vulnerável ao POODLE	✔	✘	✔	✔	✔	✔	✘
☹ Vulnerável ao DoS	✘	✘	✔	✔	✘	✘	✘
☹ Vulnerável ao FREAK	✔	✔	✘	✘	✘	✘	✘
Nota	F	F, F	C	C	C, A-, A-	F	C

Propriedades: ☺ (Boa) | 😐 (Neutra) | ☹ (Ruim) || ✔ Sim/Aplica-se | ✘ Não/Não se Aplica.

Após a montagem da rede local e realização de ataques seguindo a metodologia descrita, os resultados dos clientes analisados foram coletados e podem ser encontrados na Tabela 2. As notas foram obtidas como a seguir: ***não sofre MITM comum*** pontua *1,5 estrela*, por ser um ataque simples que não precisa de nada instalado no cliente; ***não sofre MITM com certificado raiz*** pontua *1 estrela*, pois necessita do certificado autoassinado instalado no dispositivo da vítima, sendo mais intrusivo; ***não vaza credenciais*** pontua *1 estrela*, por ser uma das consequências do MITM; ***não vaza informações financeiras*** pontua *1 estrela*, por também ser uma das consequências do MITM; ***não contém redes sociais externas*** pontua *0,5 estrela*, já que integração com redes sociais pode gerar pelo menos duas consequências: vazamento de dados bancários (privacidade) e comprometimento da defesa em profundidade, já que a segurança do aplicativo pode ser fragilizada por vulnerabilidades na rede social por transitividade. Pinagem de chave pública afeta diretamente os ataques realizados, portanto o uso de pinagem não garante pontuação. Ainda assim, destacamos essa propriedade para fins de ilustração. A vulnerabilidade contra ataques MITM (comum ou com certificado raiz) conta com um total de 2,5 estrelas, enquanto o vazamento de informações sensíveis (credenciais ou informação financeira) representam 2 estrelas. Os critérios de pontuação foram projetados para capturar a gravidade das vulnerabilidades e o nível de dificuldade imposta ao atacante.

Algumas observações gerais coletadas para cada aplicativo e servidor correspondente podem ser encontradas a seguir:

Tabela 2. Resultado dos aplicativos Android analisados quanto à vulnerabilidade a ataques de personificação e outras decisões de projeto. As notas variam de 0 a 5 estrelas.

	Banco do Brasil	Bradesco	Caixa Econômica Federal	Citibank	HSBC	Itaú	Santander
Versão analisada	6.5.0.7	2.9.6	1.3.3	9.0	1.5.10.0	4.1.5	4.4.0
Não sofre MITM comum	✔	✔	✔	✔	✔	✔	✘
Não sofre MITM com certificado raiz	✔	✘	✘	✘	✘	✘	✘
Não Vaza credenciais	✔	✘	✘	✘	✔	✘	✘
Não Vaza informações financeiras	✔	✘	✘	✘	✘	✘	✘
Não contém redes sociais externas	✔	✘	✔	✘	✔	✘	✔
Utiliza pinagem de chave pública	✔	✘	✘	✘	✘	✘	✘
Nota	★★★★★	★½	★★	★½	★★★	★½	½

✔ Sim/Aplica-se | ✘ Não/Não se Aplica.

- **Banco do Brasil (BB):** a solução do banco para pinagem de chave pública foi instalar um arquivo no aplicativo que contém a cadeia de certificação do servidor. Além disso, o BB tem sua própria rede social interna. O servidor utiliza o protocolo SSL 3.0 e algoritmos inseguros, como RC4, MD5 e SHA-1, além de não utilizar novos protocolos, como TLS 1.2 e 1.1. Estas configurações permitem que o BB seja vulnerável à maioria dos ataques citados na Tabela 1.
- **Bradesco:** o conjunto aplicativo-servidores mostrou-se muito vulnerável – apesar da resistência contra o ataque MITM comum. Para deixar de ser vulnerável aos ataques deste trabalho, o Bradesco deve realizar algum tipo de pinagem no cliente ou servidor (ou ambos). O servidor tem compatibilidade com protocolos muito antigos, como SSL 2.0 e SSL 3.0, além de utilizar algoritmos inseguros, como RC4, MD5 e SHA-1. Um outro problema de segurança é a integração do aplicativo com redes sociais externas (como Facebook).
- **Caixa Econômica Federal (Caixa):** o aplicativo ainda precisa realizar a pinagem de chave pública, assim como o Bradesco. O servidor suporta SSL 3.0 e a falta de uma renegociação segura de chaves permite o ataque DoS. Um outro problema de segurança, é o excesso de utilização do *JavaScript*, que fica armazenado no cliente de maneira transparente, permitindo uma possível injeção de código.
- **Citibank:** a situação do aplicativo é absolutamente análoga ao Bradesco. A falta de renegociação segura de chaves permite o ataque DoS e a não verificação do preenchimento restante da mensagem permite a utilização do POODLE em TLS.
- **HSBC:** apesar de sofrer o ataque MITM com certificado autoassinado, não transmitiu de maneira clara – sem cifrar – as credenciais do cliente, pois a habilitação de um dispositivo móvel permite negociação prévia de chaves criptográficas. Entretanto, algumas informações financeiras dos clientes foram vazadas no tráfego capturado, como saldo e limite do cartão de crédito, por exemplo. Ainda assim, é necessário realizar pinagem de chave pública, como o Bradesco. Do ponto de vista de segurança, um dos servidores precisa de uma melhor configuração.
- **Itaú:** a situação do aplicativo é absolutamente análoga ao Bradesco. O servidor utiliza SSL 3.0 e algoritmos inseguros, como RC4 e SHA-1, além de permitir ataques de deterioração e POODLE.
- **Santander:** dada a vulnerabilidade a ataques MITM comuns, o aplicativo opta por utilizar um protocolo adicional customizado na camada de aplicação, ao invés de utilizar somente o protocolo SSL/TLS. Nesse protocolo, o servidor envia uma chave pública RSA de 2048 *bits* que é utilizada pelo cliente para cifrar uma chave pública

efêmera de 1024 *bits*, gerada a cada conexão. Ao recuperar a chave pública efêmera utilizando sua chave privada, o servidor cifra uma chave de sessão para proteger a comunicação subsequente com o cliente. Verificou-se por engenharia reversa do aplicativo ofuscado que o protocolo customizado também não faz nenhuma autenticação de chaves públicas, tornando-se trivialmente vulnerável ao ataque MITM. A correção pode ser feita ao não depender do protocolo customizado e utilizar o protocolo TLS de maneira correta, pois desta maneira, haverá pelo menos dois benefícios: ganho de desempenho por ter apenas um protocolo utilizado e correção dos ataques detectados por este trabalho. O servidor utiliza o protocolo SSL 3.0, mas apesar da correção do ataque POODLE, é necessário abandonar este protocolo. O servidor ainda utiliza algoritmos inseguros, como RC4 e SHA-1.

5. Conclusão

Como continuidade do trabalho, todos os bancos foram notificados dos problemas encontrados. Entretanto, o departamento de segurança quase sempre foi de difícil acesso, o que mostrou-se preocupante, uma vez que este departamento deve ser um dos mais acessíveis para pesquisadores notificando vulnerabilidades. É importante ressaltar que o banco é responsável por proteger as credenciais e informações financeiras de seus clientes, desta maneira deve haver uma maior preocupação na correção dos problemas encontrados. Os resultados foram coletados em Maio de 2015 e os bancos notificados alguns dias após as coletas. Até o prazo de submissão da versão final, os autores puderam verificar que nenhum dos bancos analisados mitigou completamente o impacto das falhas descobertas.

Espera-se que os resultados deste trabalho sejam úteis para aprimoramento de segurança de aplicativos bancários, pela adoção de duas medidas principais:

1. Aplicações do lado do cliente precisam avaliar cuidadosamente chaves públicas do servidor enviadas durante a conexão. Durante a validação feita pelo cliente, deve-se também checar CRL, OCSP e pinagem de certificado.
2. Servidores também devem aumentar o nível de segurança. Este objetivo pode ser alcançado abandonando algoritmos e protocolos criptográficos obsoletos, além de implementar novas medidas de segurança (Segredo Futuro, por exemplo).

Referências

Adrian, D., , Bhargavan, K., Durumeric, Z., Gaudry, P., Green, M., and others. (2015). Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice.

Bhargavan, K., Delignat-Lavaud, A., Fournet, C., and Alfredo Pironti, P.-Y. S. (2014). Triple Handshakes and Cookie Cutters: Breaking and Fixing Authentication over TLS.

CVE (2014). *Common Vulnerabilities and Exposures: FREAK - CVE-2015-0204, POODLE - CVE-2014-3566*.

Eastlake, D. and Hansen, T. (2011). RFC 6234 - US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF).

Evans, C., Palmer, C., and Sleevi, R. (2015). RFC 7469 - Public Key Pinning Extension for HTTP.

FEBRABAN (2012). Federação Brasileira de Bancos dá dicas de segurança eletrônica. `https://www.febraban.org.br/Noticias1.asp?id_texto=1886`.

FEBRABAN (2014). Pesquisa de Tecnologia Bancária da Federação Brasileira de Bancos.

Georgiev, M., Iyengar, S., Jana, S., Anubhai, R., Boneh, D., and Shmatikov, V. (2012). The most dangerous code in the world: validating ssl certificates in non-browser software. In *ACM Conference on Computer and Communications Security*, pages 38–49.

Google, Inc. Device Monitor. `https://developer.android.com/tools/help/monitor.html`.

Handley, E. M. and Rescorla, E. E. (2006). RFC 4732 - Internet Denial-of-Service Considerations.

Hirsch, F. J. and Engelschall, R. S. (2013). *SSL/TLS Strong Encryption: An Introduction*.

Housley, R., Polk, W., Ford, W., and Solo, D. (2002). RFC 3280 - Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.

Instituto Nacional de Tecnologia da Informação (ITI) (2014). Autoridades Certificadoras.

Java Decompiler. JD-GUI - Java Decompiler. `http://jd.benow.ca`.

Kaliski, B. (1999). IEEE 1363-2000: IEEE Standard Specifications For Public Key Cryptography.

Moser, J. (2009). The First Few Milliseconds of an HTTPS Connection.

Pan, B. dex2jar - tools to work with android .dex and java .class files. `https://github.com/pxb1988/dex2jar`.

Plummer, D. C. (1982). RFC 826 - An Ethernet Address Resolution Protocol – or – Converting Network Protocol Addresses.

Qualys, Inc. SSLlabs - Qualys SSL Labs. `https://www.ssllabs.com`.

Redphx. APK Downloader - Download APK files from Android Market to PC. `http://codekiem.com/2012/02/24/apk-downloader`.

Rescorla, E. (1999). RFC 2631 - Diffie-Hellman Key Agreement Method.

Rivest, R. (1992). RFC 1321 - The MD5 Message-Digest Algorithm.

Roethlisberger, D. SSLsplit - transparent and scalable SSL/TLS interception. `https://github.com/droe/sslsplit`.

Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and Adams, C. (2013). RFC 6960 - X.509 Internet PKI Online Certificate Status Protocol - OCSP.

Sarkar, P. G. and Fitzgerald, S. Attacks on SSL a Comprehensive Study of BEAST, CRIME, TIME, BREACH, LUCKY 13 & RC4 biases. White paper.

Song, D. Dsniff - a collection of tools for network auditing and penetration testing. `http://www.monkey.org/~dugsong/dsniff`.

The OpenSSL Project. OpenSSL - The Open Source toolkit for SSL/TLS. `https://www.openssl.org`.

Wireshark Project. Wireshark - Go Deep. `https://www.wireshark.org`.