Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

Segurança de aplicações baseada em informações de contexto

Bruno Augusto Clemente de Assis

JUIZ DE FORA
DEZEMBRO, 2014

Segurança de aplicações baseada em informações de contexto

Bruno Augusto Clemente de Assis

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Eduardo Barrére

JUIZ DE FORA

DEZEMBRO, 2014

Segurança de aplicações baseada em informações de contexto

Bruno Augusto Clemente de Assis

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Eduardo Barrére
Dr. em Engenharia de Sistemas e Computação, UFRJ

Marcelo Ferreira Moreno
Dr. em Informática, PUC-Rio

Francisco Henrique Cerdeira Ferreira
M.Sc em Ciência da Computação, UFJF

JUIZ DE FORA

1 DE DEZEMBRO, 2014

A minha mãe e ao meu pai pela compreensão incondicional.

A minha irmã e aos meus amigos.

Resumo

A comunicação entre pessoas, dispositivos, processos e recursos está se tornando cada vez mais frequente. A utilização de dispositivos móveis para efetuar transações bancárias, trocar informações pessoais e transferir arquivos multimídia já é transparente para os usuários. Sendo assim, se faz necessário garantir a confiabilidade e a integridade das informações trocadas através das redes de dispositivos, sejam elas públicas ou privadas. Os mecanismos de segurança existentes muitas vezes se tornam obsoletos devido à constante evolução da área tecnológica, percebendo essa necessidade e utilizando informações sobre o contexto dos dispositivos, este trabalho tem como objetivo desenvolver um módulo de segurança para arquiteturas de aplicações sensíveis ao contexto.

Palavras-chave: Segurança da informação, sensível ao contexto, dispositivos móveis.

Abstract

Communication between people, devices, processes and resources is becoming increasingly common, the use of mobile devices to conduct banking transactions, exchange personal information and transfer multimedia files is already transparent to users, in this way, it is necessary to ensure reliability and integrity of information exchanged through networks of devices, whether public or private. Existing security mechanisms often become obsolete due to the constant evolution of technology, realizing this need and using context information of the devices, this work aims to develop a security module for context-awareness applications' architectures.

Keywords: Information Security, Context-aware, Mobile devices.

Agradecimentos

A minha mãe que em todas as ocasiões nunca deixou de me apoiar.

Ao meu pai pelo incentivo.

A minha irmã pela força.

A todos os meus parentes, amigos e amigas que sempre torceram por mim.

Ao professor Eduardo Barrére pela escolha de orientação, amizade e principalmente, pela paciência, sem a qual este trabalho não se realizaria.

Aos coordenadores do curso de Ciência da Computação, Jairo Francisco de Souza e Raul Fonseca Neto, pela atenção e pelo trabalho realizado durante todos esses anos.

Aos professores do Departamento de Ciência da Computação pelos seus ensinamentos e aos funcionários do curso, que durante esses anos, contribuíram de algum modo para o nosso enriquecimento pessoal e profissional.

"Uma mente que se abre a uma nova ideia jamais voltará ao seu tamanho original".

Albert Einstein

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

1.1 Contextualização

O rápido crescimento e evolução das diferentes tecnologias que permitem a comunicação entre pessoas, processos, máquinas ou qualquer outro dispositivo interconectado, tem facilitado o acesso à informação. Atualmente é possível acessar e-mails, ler notícias, realizar transações bancárias, enviar mensagens de texto, conversar por vídeo conferência, etc, de forma simples e de fácil acesso aos usuários, através de dispositivos móveis como smartphones e tablets.

Esta mudança vem aumentando ao longo dos últimos anos, quando gradualmente vimos a substituição de computadores desktops e notebooks por dispositivos portáteis. Grandes empresas foram forçadas a desenvolver sistemas específicos e adaptados para estas novas arquiteturas, citando como exemplo a Microsoft, uma empresa de grande porte na qual viu a necessidade de adaptar seu sistema operacional à esta mudança[1]. Isto pode ser evidenciado no *Windows* 8 que diferente dos seus antecessores foi desenvolvido para ser utilizado tanto em tablets quanto em computadores desktops e notebooks, desta forma, tornou-se imprescindível, para manter a longevidade de um software, fazer sua migração para a plataforma de dispositivos móveis.

Neste cenário podemos observar o grande aumento no uso de aplicações multimídia que permitem aos usuários tirar fotos, gravar vídeos ou compartilhar recursos multimídia entre seus dispositivos. Esta quantidade de dados carece de informações acerca do contexto em que foram inseridos. Em uma foto, por exemplo é interessante saber onde foi capturada, quando foi capturada e quem a capturou (FELICIANO et al, 2014). Logo, podemos chamar de aplicações sensíveis ao contexto, aplicações que não só armazenam os dados multimídias, mas também utilizam informações do contexto para prover conteúdo personalizado para os usuários (FENG et al, 2008; PESSOA et al, 2006) .

Uma parte dessas aplicações podem utilizar recursos de servidores web, processa-

[1]http://en.wikipedia.org/wiki/Windows_8. Acesso em: 26 de novembro de 2014

mento remoto e armazenamento remoto, seja para armazenar ou trocar informações entre serviços, recursos ou outros usuários. Assim vê-se a necessidade de garantir a integridade e a autenticidade dos dados trafegados na rede, de maneira que haja confiabilidade entre os dados recebidos e os enviados e, em caso de roubo da identidade do usuário possam ser utilizados mecanismos de bloqueio de acesso e da informação a ser recebida, para que este não tenha suas informações roubadas ou sua identidade violada.

1.2 Justificativa

Toda aplicação independente de plataforma, e que permita a troca de dados com outro recurso (seja através de arquiteturas Cliente-Servidor, Peer-to-Peer ou Híbrida) tem seus dados expostos através das redes que as interligam, se esta rede é a internet então torna-se praticamente inevitável impedir que esses dados possam ser capturados e até mesmo modificados, entre o trajeto: fonte e destino. Sendo assim, um dos primeiros passos para se ter confiabilidade (FLOSE et al, 2011), seria criptografar os dados enviados entre os participantes de uma rede para que, caso estes sejam lidos eles não possam ser transformados em informação e utilizados por quem os capturou.

Isto resolve o problema parcialmente pois ainda deixa em aberto uma solução para o problema de roubo de informações como, credenciais de acesso, que em posse desta informação seria possível acessar áreas reservadas de um determinado recurso com acesso privilegiado. Consequentemente, os esforços para manter uma aplicação segura podem ser feitos em conjunto com processo de desenvolvimento de software para que, desta forma, com a ajuda de informações do contexto das aplicações, seja possível identificar que o individuo autenticado no sistema apesar de ter as credenciais de acesso não seja reconhecido como um usuário legítimo.

Existem no mercado uma diversidade de sistemas operacionais e arquiteturas para dispositivos móveis e se citarmos os três mais utilizados: Android, iOS e Windows Phone, verificamos que as linguagens de programação e as arquiteturas utilizadas nos três sistemas são diferentes em muitos aspectos. Desta forma, tratar requisitos de segurança para cada aplicação desenvolvida em cada uma destas plataformas seria algo demorado e dispendioso.

Para encontrar uma solução viável e tentar garantir a legitimidade do usuário que se encontra autenticado na aplicação ou que esteja enviando dados a esta, podemos utilizar recursos de aplicações sensíveis ao contexto, que nos enviam informações que podem ser utilizadas para certificar autenticidade como, por exemplo, a geolocalização do aparelho, o horário em que foram enviados os dados, informações do dispositivo que está utilizando o aplicativo, formato de mídia, entre outras. De posse desses dados, é possível criar heurísticas capazes de perceber se quem está trocando dados com o servidor é o dono real das credenciais.

Com o estudo e o desenvolvimento de heurísticas para identificar tais anomalias em acessos, visa-se por conseguinte desenvolver um módulo de segurança para um Framework já desenvolvido em trabalhos anteriores, chamado ContextF (BARBOSA et al, 2013).

1.3 Objetivos

O objetivo geral deste trabalho busca definir mecanismos de segurança na camada de aplicação, e implementar um módulo de segurança para aplicações sensíveis ao contexto que seja genérico e desacoplado, com intuito de ser facilmente implementado em outras linguagens de programação.

Os objetivos específicos consistem em estudar mecanismos de criptografia de dados eficientes e de baixo custo computacional, buscar heurísticas para analisar e identificar acessos não autorizados a sistemas de informação assim como buscar mecanismos integrados de autenticação que possam interoperar com outras plataformas e serviços como, por exemplo, Google e Facebook.

1.4 Metodologia

O estudo desenvolvido neste trabalho envolve uma abordagem qualitativa, fazendo o levantamento de referências bibliográficas que forneçam embasamento teórico suficiente para a implementação e a documentação do projeto em questão.

O módulo de segurança foi desenvolvido utilizando o padrão de projeto MVC

(*Model, View, Controller*), com o objetivo de levantar os requisitos de segurança para cada camada separadamente. A implementação foi feita na linguagem de programação PHP em conjunto com o banco de dados MySQL, a escolha da linguagem e do banco de dados foram determinados por serem ambos Open Source. Além disto a linguagem PHP permite o desenvolvimento de módulos desacoplados, que podem ser desenvolvidos em linguagem C, priorizando o desempenho e facilitando assim a extensão do módulo de segurança da camada de aplicação para as demais de rede.

1.5 Estrutura do Trabalho

No Capítulo 2, Seção 2.1, o conceito de segurança da informação é apresentado, esclarecendo sua importância na sociedade e nas organizações. A seção 2.2 aborda o conceito de criptografia com uma introdução sobre o seu uso durante a história da humanidade e o seu funcionamento, também descreve algoritmos de chave simétrica e assimétrica utilizados para criptografar mensagens. A seção 2.3 aborda o conceito de contexto assim como processos que podem ser utilizados para derivar novos contextos a partir de informações elementares, e as aplicações sensíveis ao contexto que através dele conseguem alterar seu comportamento para prover serviços de forma adequada para cada perfil de usuário. No Capítulo 3, Seção 3.1 é descrito brevemente o módulo de segurança desenvolvido. A Seção 3.2 detalha os conceitos relacionados à arquitetura do módulo e o funcionamento do seu sistema de autenticação através de três etapas. A Seção 3.2 esclarece como é efetuada a análise do contexto para a tomada de decisões que podem negar ou permitir acesso à um determinado recurso baseando-se nas informações do contexto do dispositivo. O Capítulo 4, sugere recomendações de segurança que podem ser implementadas na tentativa de mitigar acessos indevidos a recursos privados. A Seção 4.1 trata de segurança na camada de persistência, sugerindo correções para evitar casos de Sql Injection. A Seção 4.2 sugere o uso de criptografia para efetuar a comunicação entre cliente e servidor. A Seção 4.3 trata de uma maneira geral falhas de segurança que podem comprometer aplicações web. O Capítulo cinco, faz o estudo de caso do módulo desenvolvido como prova de conceito. O último e sexto capítulo descreve a conclusão obtida após o desenvolvimento deste trabalho.

2 Fundamentação Teórica

Este capítulo apresenta conceitos necessários para o entendimento do estudo proposto, aborda a conceituação do que é segurança da informação e sua importância, descreve conceitos fundamentais sobre criptografia e seus algoritmos e finaliza com a explicação do que é contexto e suas aplicações.

2.1 Segurança da Informação

A preocupação com segurança da informação não é um conceito recente e nem tampouco está restritamente relacionada à sistemas de informação. Há muito, nos tempos dos faraós ja se usava conceitos de cifra de mensagens para tentar esconder o significado de certas escrituras (SILVA et al, 2011), e não tão distante podemos citar também Júlio César, imperador romano que usava cifras em suas mensagens, que ficou conhecida posteriormente como cifra de César. (TANEMBAUM et al, 2011; SILVA et al, 2011).

Durante a segunda guerra mundial era de extrema necessidade que exércitos aliados pudessem se comunicar através de equipamentos de rádio, sem que a comunicação pudesse ser lida e interpretada pelos inimigos, deste modo era necessário garantir que a informação permanecesse confidencial. Foi a partir deste momento que começaram os primeiros avanços no ramo da criptografia.

Apesar dos avanços durante a guerra, a segurança da informação só começou a se tornar destaque na sociedade civil depois da invenção das redes de computadores. Essas tecnologias permitiram abrir caminho para uma nova forma de comunicação, entretanto, trouxeram também novos desafios para a segurança da informação.

2.1.1 Importância da Segurança da Informação

A informação vem gradualmente se tornando um quesito de primeira importância em qualquer organização. Com o advento e popularização das operações de comércio eletrônico a informação tornou-se um ativo valioso para qualquer pessoa ou corporação

sendo imprescindível armazená-la de forma confiável com o intuito de proteger segredos comerciais e industriais que poderiam comprometer toda uma organização (SILVA et al, 2011). Um exemplo palpável é, o que poderia acontecer ao mercado de bebidas se a fórmula da *Coca-Cola* tivesse sido liberada para o público antes mesmo de ser patenteada. Ainda no mundo digital, quais seriam as consequências se o codigo-fonte do *Windows* fosse disponibilizado gratuitamente na internet? A resposta para essas perguntas é incerta mas poderíamos presumir que o impacto seria suficiente para mudar o mercado acionário, e causar a falência ou enriquecimento de algum grupo de indivíduos.

Podemos dividir a segurança da informação em duas partes, a parte física, e a parte lógica (SILVA et al, 2011). Por ser muito abrangente o foco deste trabalho está diretamente relacionado à parte lógica da segurança ou seja, tratando os requisitos somente em meios digitais .

Proteger a informação significa que alguns pilares da segurança da informação possam ser garantidos. Alguns autores defendem que tais pilares podem ser divididos em três grupos: confidencialidade, integridade e disponibilidade (MONTESDIOCA et al, 2013) ou em quatro: confidencialidade, integridade, disponibilidade e autenticidade (MARCIANO et al, 2006; SILVA et al, 2011). Esses pilares podem ser descritos brevemente como:

- *Confidencialidade*: A informação deve ser acessada somente por aqueles que tem prévia autorização para acessar determinado recurso.
- *Integridade*: A informação deve estar correta, ser legítima e não ter sido alterada ou danificada.
- *Disponibilidade*: A informação deve estar sempre disponível para ser acessada caso contrário não há necessidade de se preocupar com segurança visto que, a informação já estaria inacessível e por conseguinte segura contra acessos não autorizados.
- *Autenticidade*: Devem existir mecanismos que garantam que a informação é proveniente da fonte anunciada.

Com esses pilares satisfeitos pode-se começar a desenvolver sistemas mais seguros e que sejam mais confiáveis. Apesar de sua importância, a segurança deve ser balanceada

com a usabilidade de um sistema de informação. Deve-se implementar níveis de segurança diferentes para cada tipo de aplicação. Uma aplicação de mensagem instantânea não necessariamente precisa ter o mesmo nível de segurança de uma aplicação bancária, desta forma, mecanismos de segurança devem ser implementados visando contemplar as diferentes necessidades de cada aplicação.

2.2 Criptografia

A criptografia é caracterizada por alguns autores como a arte ou ciência na qual o objetivo principal é produzir, criar ou utilizar mecanismos que possam ocultar determinada informação, para que esta se torne ininteligível para um grande grupo de entidades e inteligível para um pequeno e específico grupo. (ZUQUETE et al, 2011; VERMA et al, 2011)

O uso da criptografia não é recente, há relatos históricos que descrevem que o processo de cifrar mensagens era usado desde a época da Grécia antiga, a qual ameaçada pelo reino Pérsia utilizou mecanismos de cifras de mensagens para ganhar a guerra e se livrar da eminente dominação do imperador Xerxes. (FLOSE et al, 2011)

Não muito distante, durante a segunda guerra mundial, os Estados Unidos utilizaram uma técnica de criptografia simples e eficiente que resultou em grandes vitórias nas batalhas contra o Japão, a técnica consistia em fazer a comunicação entre as tropas utilizando uma linguagem muito pouco conhecida na época, a linguagem dos índios Navajo. (TANEMBAUM et al, 2011)

2.2.1 Funcionamento

A técnica de criptografia tem como princípio transformar um texto claro (inteligível) em um texto oculto (ininteligível) utilizando cifras. Uma cifra pode ser considerada uma aplicação da criptografia através ou não de um algoritmo (ZUQUETE et al, 2011), na qual utiliza-se uma chave com o intuito de efetuar permutações entre o conjunto de caracteres de um texto até que ele se torne ininteligível para pessoas que não estão autorizadas à acessarem seu conteúdo.

É comum pensar que os algoritmos criptográficos não devem ser divulgados com a justificativa de que terceiros, conhecendo o algoritmo, poderiam utilizá-lo para descriptografar textos cifrados. Este é um engano comum entre os iniciantes no estudo da criptografia, pois já é difundido entre os profissionais e entusiastas de segurança da informação que, algoritmos criptográficos devem ser sempre públicos permitindo assim que eles sejam validados, testados e verificados pelo máximo de pessoas possíveis, em busca de falhas de segurança que possam comprometê-lo. A figura 2.1 ilustra o processo de cifragem e decifragem de uma mensagem.

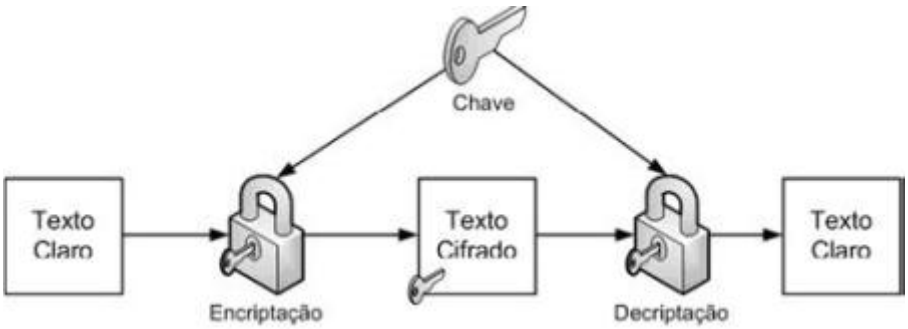

Figura 2.1: Funcionamento básico de criptografia [1]

2.2.2 Criptografia por chave simétrica

A criptografia por chave simétrica é assim chamada porque executa algoritmos criptográficos que utilizam a mesma chave tanto para criptografar quanto para descriptografar mensagens.(TANEMBAUM et al, 2011). Tais algoritmos utilizam o princípio de transposição ou permutação de bits para gerarem o texto criptografado. Isto significa que quanto maior o tamanho da chave mais difícil se torna obter o texto original. Este tipo de criptografia é frequentemente utilizada em comunicação de dados entre redes Wi-Fi e, ou, entre dispositivos que necessitam de segurança com baixo custo computacional (ZIBIDEH et al, 2014). Alguns algoritmos que se baseiam neste princípio são: *DES (Data Encryption Standard)* , que é atualmente considerado inseguro pois utiliza chaves de 56-bits, *AES (Advanced Encryption Standard)* , considerado o sucessor do DES, utiliza chaves de 128,192 e 256 bits, *3DES (TRIPLE DES)*, que se baseia em aplicar o DES três vezes de

[1] http://penseresponda.files.wordpress.com/2009/04/imagem11.jpg. Acesso em: 15 nov. 2014

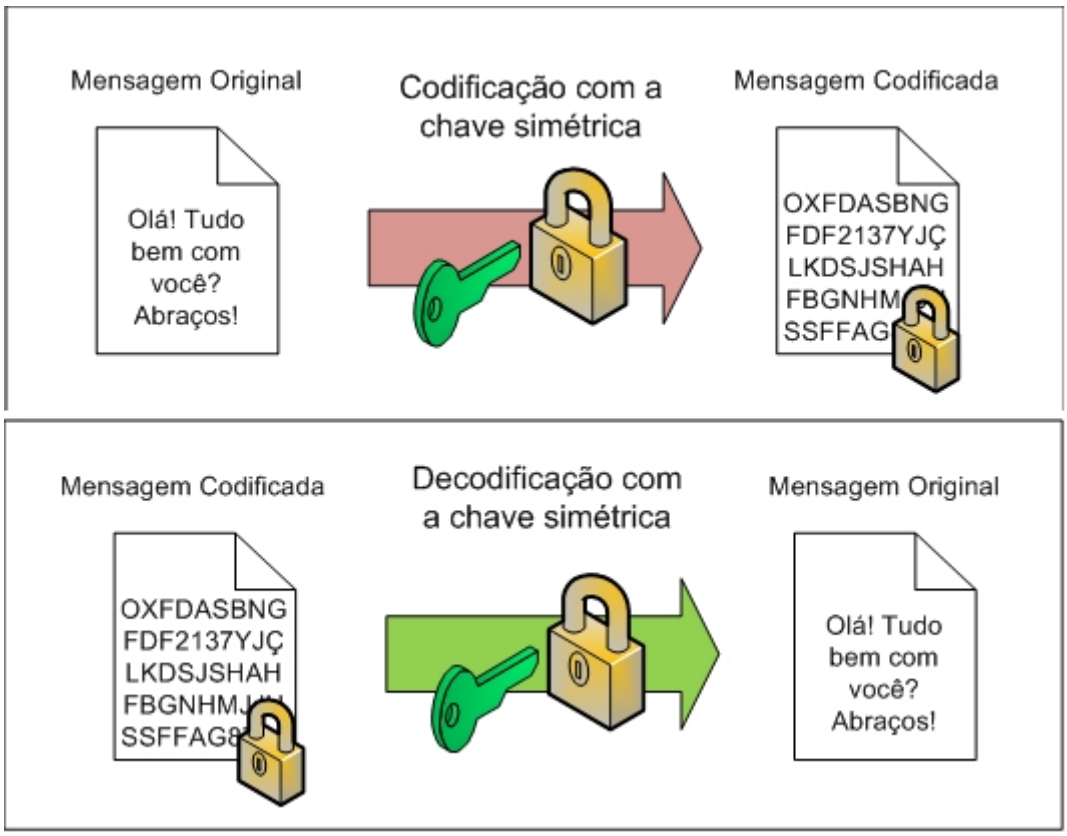

Figura 2.2: Codificação por chave simétrica [2]

forma consecutiva e o *Blowfish* que utiliza chaves entre 32 até 448 bits (MILAD et al, 2012; MANDAL et al, 2012). A figura 2.2 ilustra o processo de criptografia simétrica.

2.2.3 Algoritmo AES

O algoritmo AES sigla para *Advanced Encryption Standard*, é uma cifra de bloco e permite chaves de tamanhos de 128, 192 e 256 bits e blocos com tamanhos de 128bits e é atualmente, o modelo de cifra simétrica adotado pelo governo dos Estados Unidos. Cifra de bloco consiste em dividir o texto claro em pequenos segmentos, chamados de bloco e criptografá-los separadamente, melhorando consideravelmente o desempenho do algoritmo. (VERMA et al, 2011; UMAPARVATHI et al, 2010)

Foi desenvolvido através de uma competição promovida pelo governo norte americano onde os competidores foram desafiados a criar um algoritmo mais eficaz do que o DES, que já não se mostrava tão seguro e abriu precedentes para o desenvolvimento de

[2] $http://www.gta.ufrj.br/ensino/eel879/trabalhos\_vf\_2008\_2/hugo/NotesImages/Topic12NotesImage3.jpg$. Acesso em: 15 nov 2014

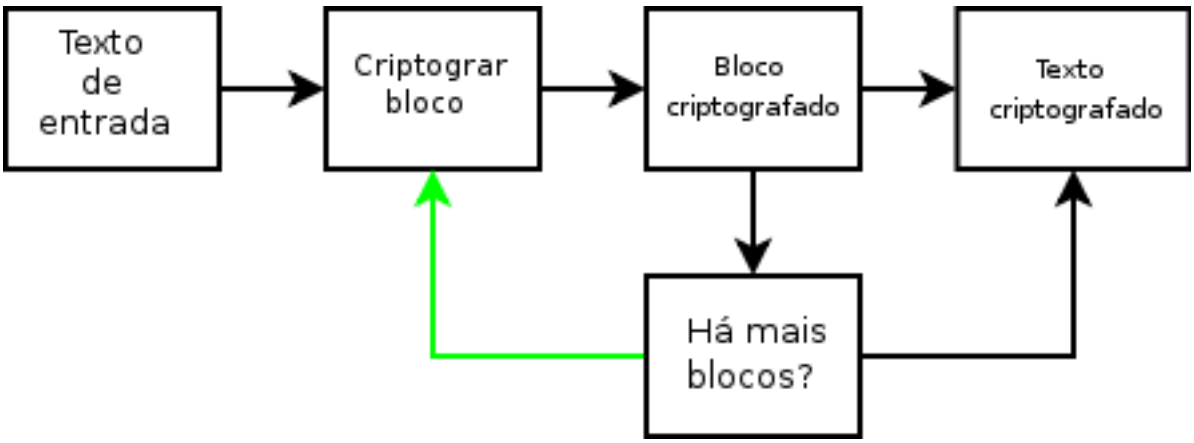

Figura 2.3: Funcionamento de uma cifra de blocos [3]

um novo padrão de criptografia de chaves simétricas.

O algoritmo AES é computacionalmente seguro pois utilizando uma chave de 128 bits o espaço de busca seria de 2^{128} chaves, isto significa que se existisse uma máquina com 1 bilhão de processadores trabalhando em paralelo capazes de verificar uma chave a cada 1 picossegundo, seriam necessários 10^{10} anos para conseguir decifrá-la (TANEMBAUM et al, 2011). Em outras palavras, ninguém estaria vivo para esperar e conhecer a resposta. Além de segurança o AES mostrou que possuía o melhor custo computacional, se comparado à seus concorrentes como DES, 3DES e Blowfish (UMAPARVATHI et al, 2010). As chaves simétricas podem resolver uma grande parte dos problemas relacionados a criptografia entretanto, caso seja vazada por algum motivo o sistema se torna completamente falho. Para resolver o problema de distribuição de chaves, foi proposto um modelo diferente chamado criptografia por chaves assimétricas.

2.2.4 Criptografia por chaves assimétricas

Também chamados de algoritmos de chaves públicas, a criptografia assimétrica trabalha maneira diferente: O processo consiste em gerar duas chaves distintas sendo uma chamada de chave pública, que deve ser distribuída publicamente, e a chamada de chave privada, na qual somente uma entidade a conhece. Com a chave pública é possível criptografar o texto claro e enviar para a entidade detentora da chave privada correspondente. A chave pública permite também verificar a autenticidade de uma mensagem recebida

[3] *http : //www.laurentluce.com/images/blog/pycrypto/2.png*. Acesso em: 15 nov 2014

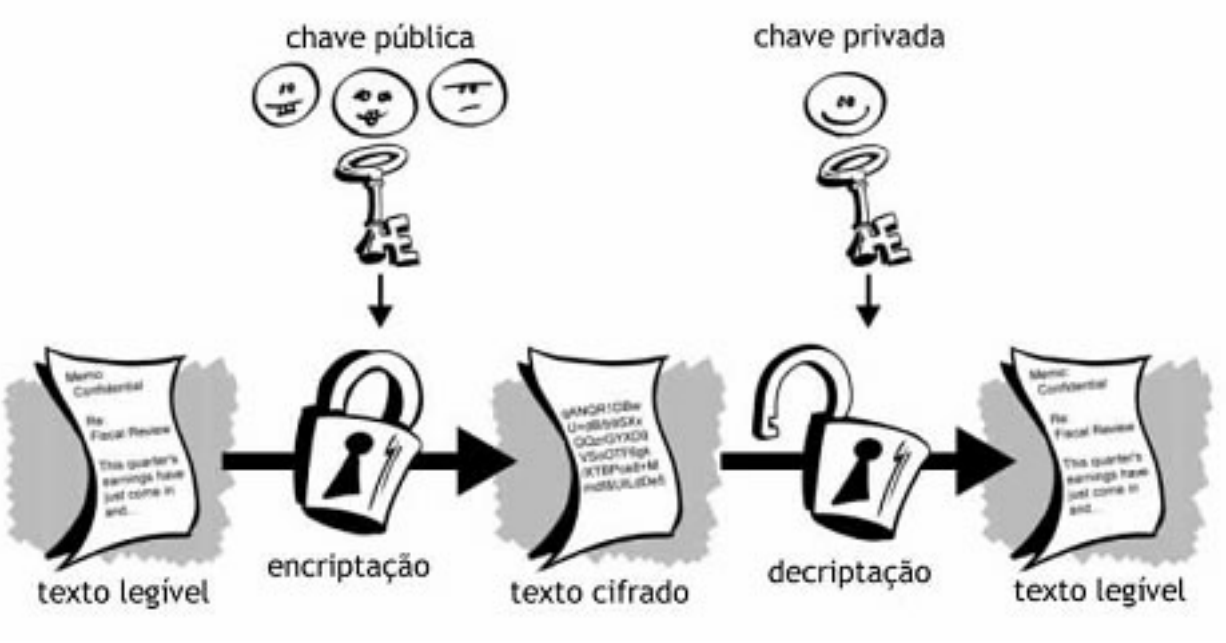

Figura 2.4: Criptografia por chaves assimétricas [4]

pela entidade detentora da chave privada, processo chamado de assinatura (TANEMBAUM et al, 2011; ZUQUETE et al, 2011). Já a chave privada é capaz de criptografar e descriptografar a mensagem recebida.

2.2.5 Algoritmo RSA

O RSA é um algoritmo de chaves assimétricas e foi apresentado pela primeira vez no ano de 1977. Seu nome se dá pelas iniciais dos nomes de seus criadores - Rivest, Shamir e Adleman, todos pesquisadores do MIT na época. (NAQI et al, 2013) Este algoritmo é considerado um algoritmo muito seguro já que sobreviveu até hoje a todas as tentativas de tentar quebrá-lo. (TANEMBAUM et al, 2011). Sua eficácia é assegurada pela dificuldade computacional de se efetuar fatoração e cálculo de logaritmos modulares de grandes números. (ZUQUETE et al, 2011) Suas aplicações são ilimitadas, mas sendo muito utilizado nos protocolos SSL/TLS para distribuição de chaves simétricas através dos navegadores, assim como o FTPs, protocolo seguro para troca de arquivos.

O processo de geração de chaves pode ser descrito em 5 passos:

- Escolha de forma aleatória dois números primos grandes P e Q.

[4] $http://www.dsc.ufcg.edu.br/pet/jornal/abril2014/images/materias/historia_da_computacao/img4.jpg$. Acesso em: 15 nov 2014

- Calcule n = $p \times q$.
- Calcule a função totiente em $\phi(n) = (p-1) \times (q-1)$.
- Escolha um inteiro e tal que $1 < e < \phi(n)$ de forma que e e $\phi(n)$ sejam primos entre si.
- Calcule d de forma que $d \times e \equiv 1 \times (\mod \phi(n))$.

Ao final da execução desse processo teremos a chave pública como sendo o par de números n e e e a chave privada o par de números n e d. Denotando m por um texto plano, c como sendo a mensagem cifrada e (n e e) como sendo a chave pública. Para criptografar uma mensagem basta calcular a função: $c = m^e \times \mod n$. Da mesma maneira sendo (n e d) a chave privada, para descriptografar uma mensagem basta calcular a função $m = c^d \times \mod n$. A única desvantagem deste algoritmo é que exige que chaves tenham um tamanho de pelo menos 1024 bits para se ter um nível de segurança confiável , tornando-o o lento em aplicações que necessitam de um rápido tempo de resposta.(TANEMBAUM et al, 2011)

2.3 Contexto

Contexto pode ser definido como sendo informações relevantes ou não, que caracterizam determinado momento em que uma aplicação ou entidade se encontram (HSU et al, 2011). Três importantes aspectos sobre contexto podem ser enumerados como: onde você está, com quem você está e os recursos que se encontram próximo a você. Uma definição de contexto pode ser traduzida como:

> Qualquer informação que caracteriza a situação de uma entidade, sendo que, uma entidade pode ser uma pessoa, um lugar ou um objeto considerados relevantes para a interação entre um usuário e uma aplicação. O contexto é tipicamente a localização, a identidade e o estado das pessoas, grupos ou objetos físicos computacionais (BARBOSA et al, 2013, p. 8)

. Segundo SCHILIT et al (1994), podemos categorizar o contexto em 4 items:

- **Contexto computacional**: rede, conectividade, custo da comunicação, banda passante, recursos (impressoras, estações, etc.).

- **Contexto do usuário**: perfil do usuário, posição, velocidade, pessoas próximas, situação social, etc.
- **Contexto físico**: luminosidade, nível de ruído, temperatura e humidade.
- **Contexto de tempo**: hora do dia, dia/mes/ano, semana, época do ano.

Tais informações e outras similares podem ser utilizadas para criar aplicações que se comportam de maneira diferente dependendo do contexto em que estão inseridas. Essas aplicações são comumente chamadas de aplicações sensíveis ao contexto (PESSOA et al, 2006).

2.3.1 Aplicações sensíveis ao contexto

Aplicações sensíveis ao contexto (ASC), são softwares que utilizam dados acerca do contexto em que estão inseridos, para tomar decisões ou fornecer informações. Em geral aplicações não sensíveis ao contexto trabalham com os dados explicitados pelos usuários, já nas ASC, toda e qualquer informação pré-definida como contexto, é também utilizada para permitir uma maior interação entre a aplicação e o usuário, ou para fornecer dados estatísticos para o serviço fornecido. Uma ilustração deste tipo de aplicação pode ser visualizada na figura 2.5.

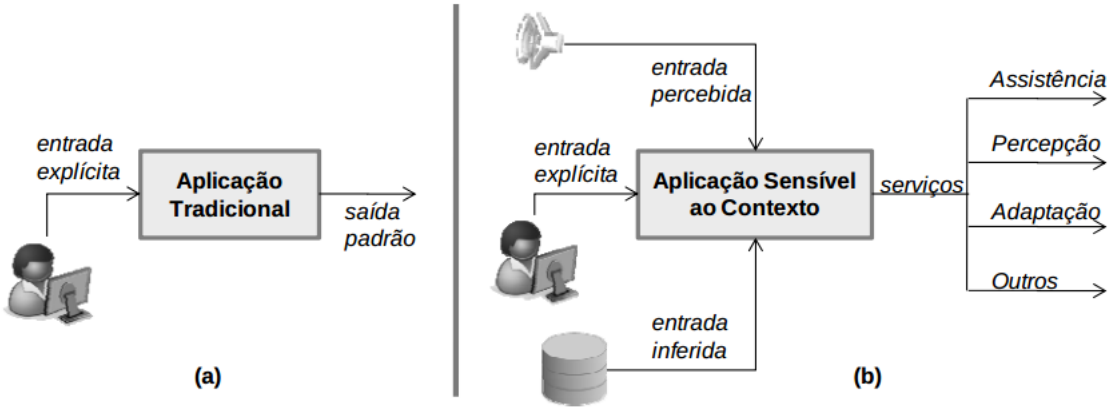

Figura 2.5: Em (a) uma aplicação não sensível ao contexto, em (b) uma aplicação sensível ao contexto. [5]

As ASC podem utilizar diversas informações relevantes dos mais variados dispositivos para formar um contexto, podemos citar como exemplo, os smartphones modernos

que possuem diversos tipos de sensores como: de temperatura, altitude e pressão. Além disto também carregam dados sobre a geolocalização do aparelho, versão do sistema operacional, nome do dispositivo, entre outras (PESSOA et al, 2006). As informações relativas ao contexto podem ajudar a desenvolver aplicações com mais interatividade e segurança para os usuários. Para exemplificar pode-se imaginar uma aplicação para um campus universitário, ela poderia se comportar de maneira diferente em cada local da universidade, no caso de o usuário estar localizado no restaurante universitário a aplicação poderia mostrar o cardápio do dia, em contrapartida se o usuário se aproxima da biblioteca, poderia mostrar os livros com devolução em atraso.

[5] $http://nuvemandroid.files.wordpress.com/2014/02/aplicacaotradicioinal\_x\_sensivelcontexto.png$. Acesso em: 15 nov 2014

3 Módulo de segurança proposto

A proposta do módulo de segurança desenvolvido neste trabalho, foi elaborada para servir também como parâmetro de implementação, para que possa ser desenvolvido em outras linguagens de programação e arquiteturas que interajam com aplicações sensíveis ao contexto. No capítulo quatro são feitas recomendações de segurança que podem ser implementadas de acordo com o nível de segurança que se pretende obter. Quanto mais recursos de segurança forem implementados, maior a segurança, porém inevitavelmente maior será o tempo de desenvolvimento necessário para colocar a aplicação em produção, sendo assim a implementação de tais recomendações fica a critério do desenvolvedor.

3.1 Arquitetura

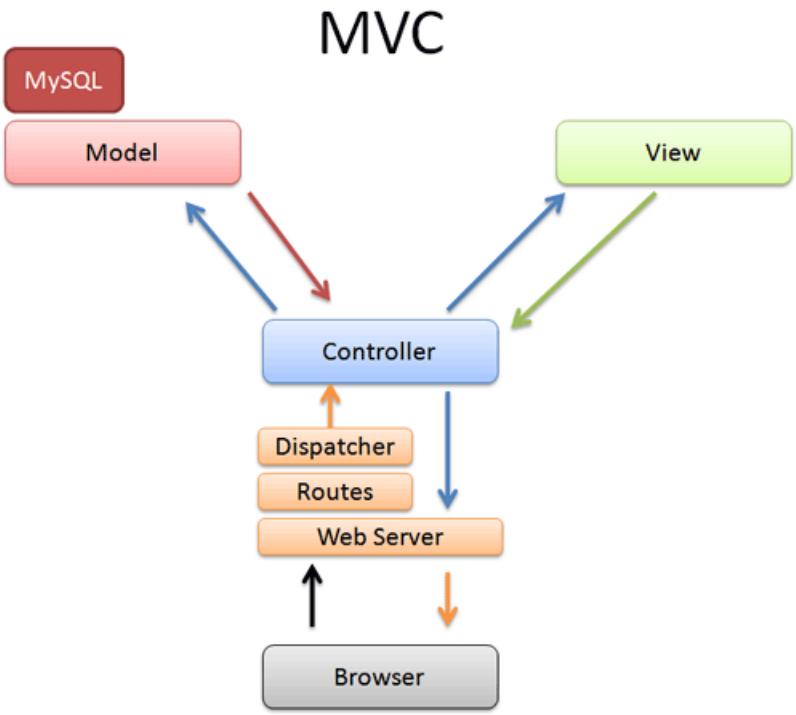

Figura 3.1: Arquitetura MVC [6]

[6] *http : //i.stack.imgur.com/Beh3a.png*. Acesso em: 15 nov 2014

O módulo foi desenvolvido utilizando o padrão de projetos MVC (*Model, View, Controller*), este padrão permite tratar os requisitos de segurança separadamente, dividindo-os em camadas, onde o *Model* é responsável pelos objetos e consultas ao banco de dados, a *View* é responsável pela visualização do conteúdo e o *Controller* responsável por receber requisições, enviar respostas ao cliente e também efetuar a comunicação entre o *View* e o *Model*. A figura 3.1 ilustra o funcionamento do modelo MVC padrão.

É de responsabilidade do módulo de segurança tratar todas as requisições de entrada e saída entre cliente e servidor, tratar consultas em banco de dados, efetuar autenticação dos usuários no sistema e detectar anomalias através de análise do contexto, este módulo se porta como uma nova camada no modelo MVC efetuando um filtro de segurança entre cada camada. A figura 3.2 ilustra o funcionamento do modelo MVC juntamente com o módulo de segurança incorporado.

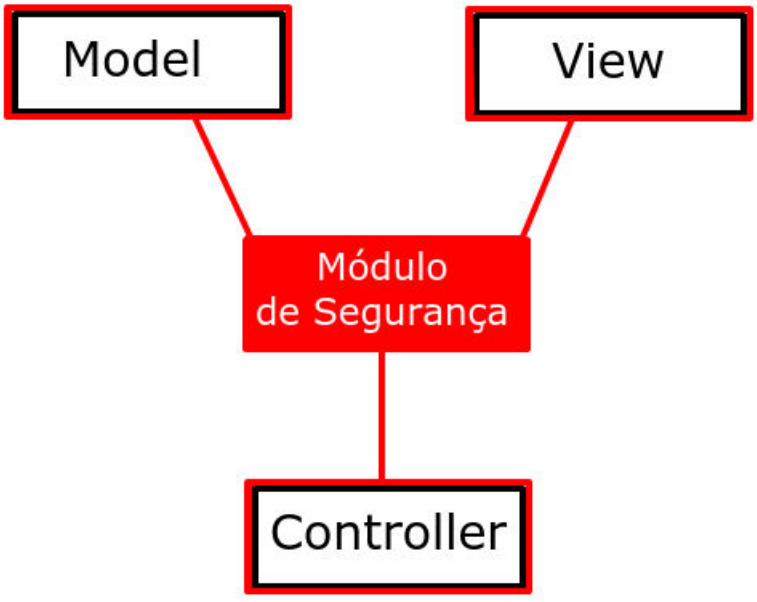

Figura 3.2: Arquitetura MVC com Módulo de segurança

3.2 Autenticação

A autenticação através deste módulo se dá em três etapas, a primeira etapa necessita da intervenção do usuário podendo ser atingida através de mecanismos de autenticação por login e senha ou através de mecanismos integrados como o OAuth2, protocolo este que é amplamente utilizado pelo Facebook e pelo Google+. A segunda etapa não há a intervenção do usuário, e é caracterizada pela autenticação do dispositivo no sistema através de uma chave cadastrada no dispositivo, a terceira etapa é feita através da autenticação do contexto em que o usuário está inserido, e também não há a intervenção do usuário.

3.2.1 Primeira etapa

Autenticação por Login e Senha

Esta forma de autenticação é simples e utilizada em alguns sistemas de informação, consiste em cadastrar um nome de usuário e uma senha para cada usuário em um banco de dados e solicitar esses dois campos a cada novo acesso ao sistema, então, compara-se esses dois valores fornecidos pelo usuário com os valores armazenados no banco de dados e, caso sejam idênticos, é gerado um token de acesso ao sistema que pode ser utilizado para identificar o usuário logado, liberando o acesso à recursos aos quais o usuário tenha permissão.

Autenticação por OAuth2

O OAuth2 é um protocolo de autenticação que está se tornando padrão para aplicações de redes sociais, grandes companhias como Google, Twitter e Facebook já o utilizam e fornecem APIs para que os desenvolvedores possam desenvolver mecanismos de autenticação baseados neste protocolo.

Seu funcionamento é simples, o primeiro passo do usuário é acessar uma aplicação da qual ele deseja obter recursos, após acessar a aplicação o usuário solicita acesso à um determinado recurso, como ainda não está autenticado no sistema, a aplicação redireciona o usuário para o servidor de autenticação, onde este deve fornecer seus dados de acesso.

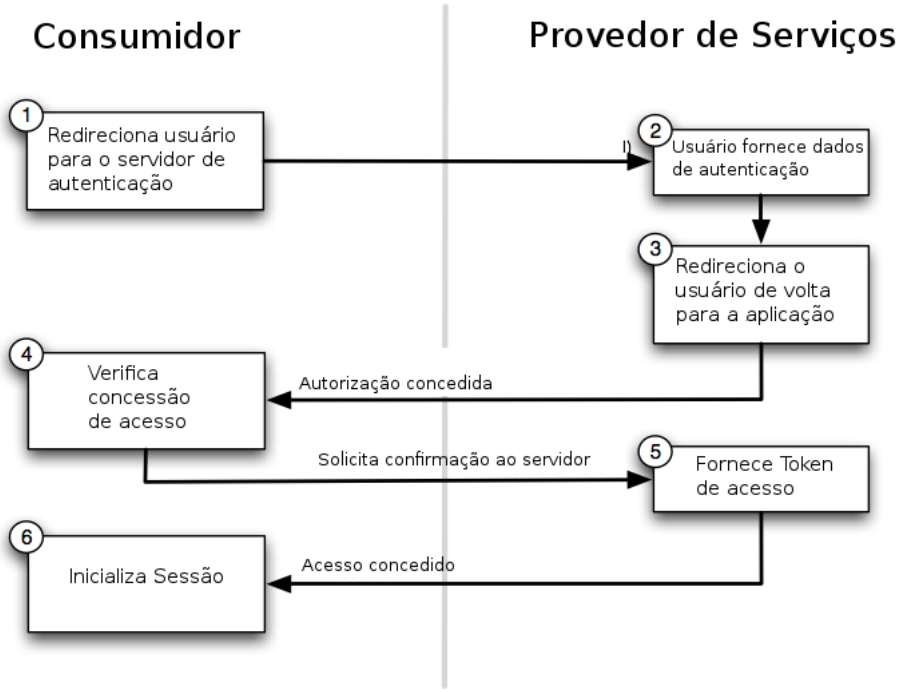

Figura 3.3: Autenticação através do OAuth2 [7]

Após fornecidos e o usuário autenticado, o servidor o redireciona de volta para a aplicação, esta por sua vez solicita uma confirmação ao servidor para verificar se foi realmente o usuário quem fez a requisição, o servidor de autenticação então responde com os dados do usuário e um token de acesso. O token recebido pode então ser utilizado para acessar recursos da aplicação.

3.2.2 Segunda etapa

Autenticação por chave de segurança

Após feita a autenticação da primeira etapa, neste momento é solicitado uma chave de segurança que somente o servidor de aplicação e o aplicativo instalado no dispositivo conhecem, esta etapa serve para autenticar o dispositivo, permitir o bloqueio em caso de perda ou roubo, além de negar acesso em caso da violação por terceiros, dos dados da etapa anterior. Seu funcionamento se dá da seguinte forma: ao instalar o aplicativo pela primeira vez é gerada uma chave única que fica armazenada no dispositivo. Após a execução da primeira etapa de autenticação a chave então é enviada ao servidor que

[7] *http* : *//blog.rivendel.com.br/wp − content/uploads/*2013/06/*oauth*2.*png*. Acesso em: 15 nov 2014

verifica sua autenticidade e, caso esta seja verdadeira, o dispositivo já foi cadastrado e liberado, sendo assim, a autenticação prossegue para a terceira etapa que é caracterizada pela análise do contexto. Caso a chave não corresponda com a cadastrada, o sistema deve gerar uma notificação através de e-mail ou SMS para o dono da conta perguntando se este deseja liberar este novo dispositivo, nesta notificação consta um código que será utilizado para liberar o dispositivo.

3.2.3 Terceira etapa

Autenticação por análise de contexto

A terceira e última etapa de autenticação, é feita através da análise do contexto em que o dispositivo está inserido e atribui uma pontuação para cada elemento pertencente ao contexto. As pontuações são concedidas baseadas nas heurísticas selecionadas, e a aprovação da autenticação é concretizada quando se obtém uma pontuação mínima, calculada com base no nível de segurança selecionado para esta etapa.

3.3 Análise de contexto

A análise do contexto é executada em dois momentos, no primeiro é feita a análise quando o usuário tenta se autenticar no sistema, no terceiro passo de autenticação. As outras análises de contexto são feitas toda vez que o usuário da aplicação faz uma nova requisição ao servidor, isso garante que em caso de roubo de sessão ou falha em outros mecanismos de segurança, o contexto possa bloquear o acesso indevido a algum recurso.

A cada nova requisição, heurísticas são utilizadas para comparar o último contexto armazenado no servidor com o contexto atual enviado pelo cliente e para cada heurística utilizada é atribuído um peso, que tem como finalidade ponderar a sua importância e ajustar a análise priorizando um determinados elementos do contexto. Com base nessas heurísticas, o módulo de segurança decide se libera ou não o acesso. Se o acesso for liberado, o sistema então atualiza o contexto no banco de dados. Exemplificando, imagine que um usuário encontra-se localizado na cidade do Rio de Janeiro, na data de 15 de novembro às 23:05 e envia um arquivo multimídia ao servidor e , logo após o envio do

arquivo um outro usuário que possa ter descoberto o login, a senha e a chave do dispositivo, envia um novo arquivo um minuto depois, mas agora este usuário está localizado no Acre, às 21:05. Pela diferença de localização e de fuso horário, o acesso é bloqueado, assumindo que houve uma violação de segurança. A análise não se limita somente à geolocalização, mas também utiliza outras informações como, qual o dispositivo utilizado, qual versão do sistema operacional, velocidade de deslocamento, distância máxima da localização anterior, atraso de horário e delimitação de área. Existem inúmeras outras informações que poderiam ser utilizadas para aprimorar a segurança baseada em contexto como: pressão atmosférica, altitude, sensor de batimento cardíaco, entre outras. Porém tais informações se restringem a dispositivos mais novos e inviabilizaria o desenvolvimento de um módulo mais genérico, como o proposto neste trabalho.

3.4 Heurísticas

Heurísticas são algoritmos que executam determinada função, com objetivo de obter uma solução viável mas não necessariamente ótima. Neste trabalho o conjunto de algoritmos que efetuam a análise do contexto, somados, constituem uma heurística. Esta heurística pode ser balanceada para tentar obter a melhor solução quando se trata de análise de contexto. Todos os algoritmos quando obtêm sucesso, retornam a pontuação parametrizada nas configurações do módulo ou 0 em caso de insucesso. As heurísticas utilizadas no módulo são descritas a seguir:

- **Distância máxima entre o contexto atual e o contexto anterior:** Verifica a diferença entre a distância do ponto geográfico atual enviado pelo usuário e a distância do último contexto armazenado no banco de dados, se esta diferença estiver dentro do intervalo selecionado é retornada a pontuação atribuída a esta heurística.
- **Velocidade de deslocamento:** Verifica a velocidade de deslocamento calculando a relação distância / tempo, entre a posição atual do usuário e a do último contexto, se esta velocidade for menor ou igual do que a estabelecida no parâmetro, retorna-se a pontuação.

- **Atraso de data:** Verifica se a data da requisição atual é anterior à da última requisição, desta forma pode-se bloquear o acesso caso haja diferença entre os relógios dos dispositivos. Caso a data seja posterior, retorna-se a pontuação parametrizada.

- **Análise do dispositivo:** Verifica se o dispositivo que fez a requisição é igual ao dispositivo que fez a última requisição.

- **Análise do O.S:** Verifica se o sistema operacional é idêntico ao que fez a última requisição.

- **Delimitação de área geográfica:** Pode-se também limitar o uso do aplicativo por áreas geográficas, esta heurística estabelece dois pontos para delimitar o raio de uso da aplicação, caso a geolocalização atual esteja dentro raio selecionado, retorna-se a pontuação parametrizada. A figura 3.4 ilustra o exemplo de um contexto X (inválido) e um Y (válido) para esta heurística.

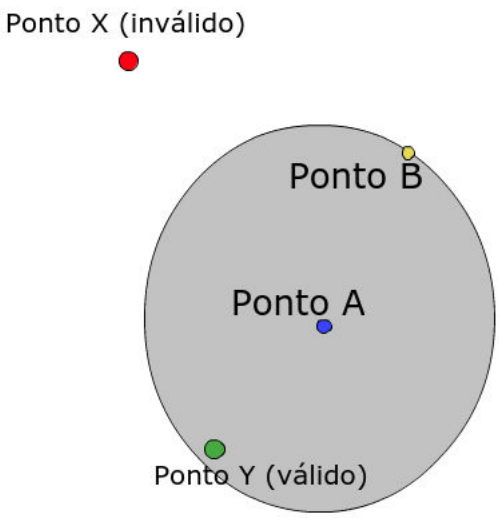

Figura 3.4: Heurística por delimitação de área geográfica

Além da seleção das heurísticas, ainda pode ser parametrizado o nível de segurança a ser utilizado, possuindo três opções: 100% (Segurança máxima), 60% (média) e 30% (mínima). Supondo que cada heurística utilize 10 pontos, para se obter autenticação

no sistema seriam necessários obter, para cada configuração respectivamente 60, 36 e 18 pontos. A figura 3.5 ilustra as possibilidades de configuração do módulo em questão.

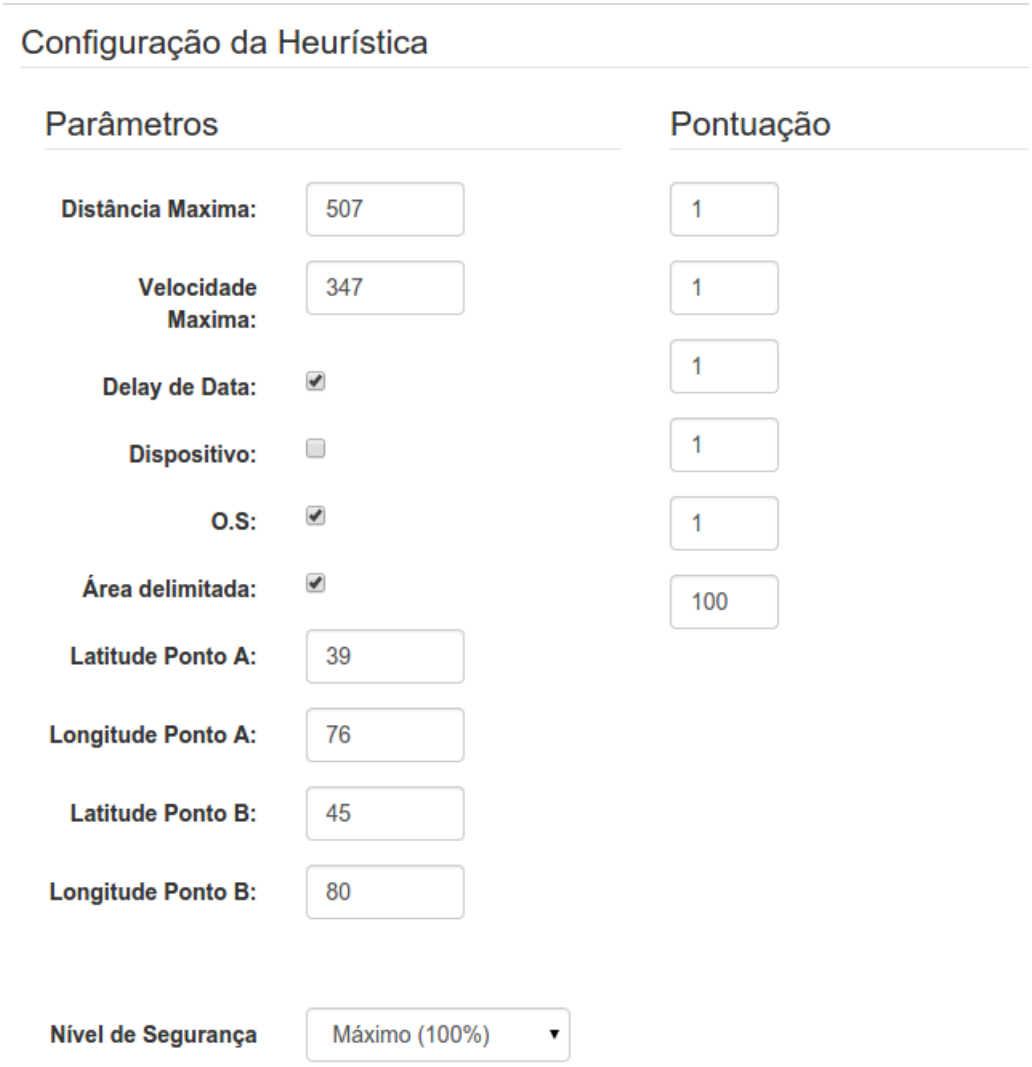

Figura 3.5: Configurações do módulo de segurança

4 Recomendações de segurança

Embora o trabalho desenvolvido tenha o objetivo de criar uma solução de segurança para aplicações sensíveis ao contexto, este por si só não soluciona por completo o problema de segurança, considerando que a segurança da informação é um conceito muito amplo, a seguir são feitas recomendações de segurança que devem ser implementadas sempre que possível. Tais recomendações são embasadas nas falhas de segurança que se encontram na lista das mais exploradas segundo a OWASP[8].

4.1 Segurança no banco de dados

Anti SQL Injection

SQL injection é um erro muito comum em aplicações que utilizam banco de dados, e pode ser facilmente evitado tratando a entrada de dados antes de fazer requisições ao banco, o exemplo a seguir em PHP e MySQL é uma sugestão de implementação que evita a falha:

```
1       $variavel = mysql_real_escape_string($variavel);
2       settype($variavel, 'integer');
3       if(is_number($variavel)) {
4            $query = "SELECT * FROM tabela WHERE id = $variavel;";
5      } else
6          die('A variável não é um número');
```

Além do tratamento de tipo recomenda-se também a parametrização da entrada como a seguir:

```
1      public function getById($id) {
2           try {
3                $dispositivo = null;
4                $stmt= "";
5                $query = "SELECT * FROM ".self::table_name." WHERE
                      id_dispositivo = ? ";
6                if($stmt = DataBase::getConn()->prepare($query)) {
7                     $stmt->bind_param("i",$id);
```

[8]https://www.owasp.org. Acesso em: 15 nov 2014

```
8                       $dispositivo = $this->montaObjetoCompleto(
                            $stmt);
9               }
10              return $dispositivo;
11
12          } catch (Exception $ex) {
13              throw $ex;
14          }
15
16      }
```

Permissões em banco de dados

Caso a segurança contra sql injection falhe, ainda podemos utilizar o mecanismo de permissões de usuário no banco de dados, para que o invasor só tenha acesso aos dados de uma tabela, e só consiga efetuar a leitura dos dados, não conseguindo fazer as operações de INSERT, UPDATE e DELETE nem nessa, e nem em nenhuma outra tabela do sistema, evitando assim, o comprometimento do resto do sistema. Sugestão em pseudo-código:

```
1       //Selecionamos o usuário que só tem permissão de SELECT na
            tabela XYZ.
2       set_user("UsuarioComPermissaoNaTabela-XYZ");
3       select("SELECT * FROM tabela-XYZ WHERE idXYZ =
            $sqlInjection");
```

Armazenamento em banco de dados

As senhas dos usuários devem ser armazenadas no banco de dados utilizando o algoritmo SHA-1 concatenado com um numero aleatório. Desta forma em caso de comprometimento do banco, evita-se que senhas muito simples possam ser descobertas através de *bruteforce*. O pseudo-código abaixo exemplifica o funcionamento:

```
1       $senhadoUsuario = filtraEntrada($senha);
2       $salt = rand(0,1000);
3       $senhadoUsuario = sha1($senhaDoUsuario.$salt);
4       $query = "INSERT INTO tabela_usuarios (senha,salt) VALUES(
            $senhadoUsuario,$salt);
```

4.2 Segurança na comunicação de dados

Ataques de "man in the middle", podem ser evitados utilizando criptografia para troca de informações entre cliente e servidor. Para aplicações que utilizam como cliente dispositivos móveis como, celulares e smartphones é de extrema necessidade que a criptografia, além de segura, seja eficiente, consumindo o mínimo de energia possível. Para o trabalho desenvolvido recomenda-se o uso do protocolo SSL/TLS no servidor, juntamente com RSA utilizando uma chave de tamanho mínimo de 1024 bits para a troca das chaves simétricas, e AES com uma chave mínima de 128 bits para troca de informações entre o cliente e o servidor, pois é considerado o algoritmo simétrico mais eficiente e de menor custo computacional de acordo com UMAPARVATHI et al (2010).

4.3 Segurança na aplicação

4.3.1 Cross site scripting (XSS)

Para aplicações que também rodam em um browser o módulo de segurança fornece a chamada da função *filtraSaida($var)*, que deve ser utilizada em todo conteúdo que será mostrado no navegador, com o intuito de tratar tentativas de injeção de códigos javascripts maliciosos que podem comprometer o sistema permitindo, por exemplo, o roubo de cookies de um usuário.

```
1  function filtraSaida($var) {
2      //Verifica aqui o conteúdo da string $var
3      //remover tags possivelmente maliciosas como <script>,
           alert(),document.cookie e similares.
4  }
```

4.3.2 Cross site request forgery (CSRF)

A falsificação de solicitação entre sites ou em inglês *Cross site request forgery (CRSF)* , pode ser evitado atribuindo um token único para cada página ou conteúdo gerado para o cliente. Assim, caso haja a tentativa de uma solicitação forjada, o servidor verifica se o token na sessão é o mesmo da página, se ambos forem idênticos, a requisição

é legítima, caso contrário, nega-se o recurso.

4.3.3 Geração da chave de segurança

A chave de segurança contida no dispositivo, pode ser gerada através da combinação dos elementos do contexto no momento anterior em que é gerada, concatenando-a à um número aleatório suficientemente grande. Em seguida aplica-se o algoritmo de SHA-1 sobre esses dados gerando uma nova chave, este processo torna a descoberta da chave por ataques por força bruta computacionalmente difícil.

4.3.4 Criptografar dados manipuláveis

Em aplicações onde há a exposição de dados que podem ser manipuláveis pelo usuário, é recomendado que sejam criptografados todos os dados enviados ao cliente. Com essa técnica evita-se qualquer tentativa de manipulação de dados, incluindo SQL Injection, XSS e CSRF. Abaixo seguem dois exemplos: o primeiro é um formulário com campos sem criptografia e o segundo com os campos criptografados.

```
1      <form>
2      <input type="hidden" name="idCliente" value="1234" />
3      <select name="ocupacao">
4           <option value="1"> Analista de Sistemas </option>
5           <option value="2"> Estudante </option>
6           ...
7      </select>
8      </form>
```

Dados criptografados:

```
1      <form>
2      <input type="hidden" name="idCliente" value="Awa#m4qan" />
3      <select name="ocupacao">
4           <option value="oinefwiuh"> Analista de Sistemas </
               option>
5           <option value="asçlijk9823"> Estudante </option>
6           ...
7      </select>
8      </form>
```

5 Estudo de caso

O estudo de caso, visa efetuar uma prova de conceito nos mecanismos de autenticação simples, por OAuth2, pela chave de segurança e pelo contexto. Atrelado a isso são simuladas falhas na primeira, segunda e terceira etapa do processo de autenticação, detectando mudanças bruscas no contexto para novas requisições, com o objetivo de verificar a eficácia das heurísticas de análise do contexto.

5.1 Simulação e configurações

Para os testes a seguir foram utilizadas as seguintes configurações iniciais:

Figura 5.1: Configurações iniciais do módulo de segurança

Tabela 5.1: Tabela (tb_contexto) com campos do contexto inicial

Campo	Descrição
Versão do S.O	7.1.2.
Dispositivo	iPhone 5
Latitude	40
Longitude	77
Data	14-11-2014 22:30:58

Tabela 5.2: Tabela (tb_dispositivo) com campos do dispositivo inicial

Campo	Descrição
Chave	ABCD-EFGH-IJKL-MNOP
Liberado	true

5.2 Simulação de perda de login e senha

Assumindo que o usuário perdeu, ou teve seus dados da primeira etapa de autenticação violados, ou seja, perdeu seu login e senha de acesso ao sistema ou teve comprometido seus dados de acesso ao sistema de autenticação por OAuth2. O invasor então obtém sucesso apenas na primeira etapa utilizando as credenciais abaixo:

Figura 5.2: Primeiro passo violado

Como o invasor não conhece a chave do dispositivo do usuário, o invasor efetua

Tabela 5.3: Tabela (tb_usuario) com campos do usuário inicial

Campo	Descrição
login	teste@moduloseguranca.com
senha	123456
Id OAuth2	800666789956826

uma tentativa de acesso utilizando a chave arbitrária "ABCD-ABCD-ABCD-ABCD", o módulo de segurança bloqueia o acesso e responde com seguinte mensagem:

Figura 5.3: Segundo passo violado

Como observado, o sistema identifica que a chave do dispositivo que está tentando acesso, não existe ou não está atrelada à esse usuário e envia um e-mail ou SMS para o dono da conta informando da tentativa de acesso e também um código para liberar o dispositivo.

5.3 Simulação de perda de chave de segurança

Assumindo que além dos dados da primeira etapa, a chave do usuário também tenha sido comprometida, e que o invasor se encontra em uma região diferente da que foi feita na última requisição legítima ao servidor, sendo assim, o contexto do atacante é descrito na tabela 5.4

Tabela 5.4: Tabela (tb_contexto) com campos do contexto do atacante

Campo	Descrição
Versão do S.O	7.1.2.
Dispositivo	iPhone 5
Latitude	15
Longitude	17
Data	14-11-2014 22:30:58

Após a tentativa de acesso, o sistema responde com a seguinte mensagem, representada na figura 5.4

Figura 5.4: Tentativa de violar o terceiro passo (contexto)

O sistema detecta que a geolocalização do usuário não contempla as configurações do módulo, bloqueia o acesso ao sistema e qualquer requisição futura.

6 Conclusão

O constante aperfeiçoamento de sensores, dispositivos móveis e equipamentos capazes de fornecer dados acerca do contexto em que estão inseridos, torna promissor o desenvolvimento de softwares e deixam a experiência do usuário mais objetiva e interativa, além de permitir a oferta de serviços e conteúdos mais precisos para cada perfil.

O desafio de desenvolver um módulo de segurança que fosse desacoplado e independente de arquitetura, foi atingido, embora limitações na quantidade de informações sobre o contexto terem estreitado o nível de segurança que poderia ser obtido caso fosse possível utilizar mais informações como: temperatura, altitude, pressão e etc, que são exclusivas de dispositivos mais modernos. Ficou evidenciado que, o contexto é uma ferramenta valiosa para auxiliar o desenvolvimento de medidas de seguranças. Foi possível perceber que a criptografia sozinha não soluciona todos os requisitos de segurança mas, deve ser sempre utilizada como ponto de partida quando se inicia o desenvolvimento de qualquer projeto que, exija confiabilidade.

Aliado ao uso do contexto também fica evidente que tratar a segurança separadamente em camadas, utilizando o padrão MVC para cada etapa de autenticação (usuário, dispositivo e contexto), minimizou significativamente os riscos de acessos não autorizados quando há perda de dados de autenticação.

Embora o estudo aqui proposto tenha abordado a segurança exclusivamente na camada de aplicação utilizando uma linguagem script como é o PHP, toda a análise de contexto pode ser desenvolvida em camadas mais baixas utilizando a linguagem C o que ocasionaria uma melhora do desempenho. Para tal é proposto por ANITHA et al (2006) um sistema de detecção de intrusão (IDS) em C, que analisa os cabeçalhos dos pacotes através de expressões regulares, em um trabalho futuro fica a proposta de acoplar um IDS ao módulo deste trabalho.

Referências Bibliográficas

Anitha, A.; Vaidehi, V. Context based application level intrusion detection system. **Networking and Services, ICNS. International conference**, p. 16, 2006.

Barbosa, M. H. **Contextf: framework de apoio a construção de aplicações sensíveis ao contexto**. Universidade Federal de Juiz de Fora, 2013.

Feliciano, R. A. **Uma arquitetura para um servidor de contexto**. Universidade Federal de Juiz de Fora, 2014.

Feng, Y.; Lapata, M. **Automatic image annotation using auxiliary text information**. In: ACL HLT, 2008.

Flose, V. B. S. **Criptografia e curvas elípticas**. Universidade Estadual Paulista "Julio de Mesquita Filho", 2011.

Hsu, I.-C. An architecture of mobile web 2.0 context-aware applications in ubiquitous web. **Journal of Software**, v.6, p. 705–715, 2011.

Mandal, P. C. Evaluation of performance of the symmetric key algorithms: Des, 3des ,aes and blowfish. **Journal of global research in computer science**, v.3, p. 67, 2012.

Marciano, J. L. P. **Segurança da informação - uma abordagem social**. Universidade de Brasilia, 2006.

Milad, A. A.; Muda, H. Z.; Noh, Z. A. B. M. ; Algaet, M. A. Comparative study of performance in cryptography algorithms (blowfish and skipjack). **Journal of computer sciences**, v.8, p. 1191–1197, 2012.

Montesdioca, G. P. Z. **Satisfação do usuário com as práticas de segurança da informação**. Universidade Federal do Rio Grande do Sul, 2013.

NaQi; Wei, W.; Zhang, J.; Wang, W.; Zhao, J.; Li, J.; Shen, P.; Yin, X.; Xiao, X. ; Hu, J. Analysis and research of the rsa algorithm. **Information Technology Journal**, p. 1818–1824, 2013.

Pessoa, R. M. **Infraware: Um middleware de suporte à aplicações sensíveis ao contexto**. 2006.

Schilit, B. N.; Adams, N. ; Want, R. Context-aware computing applications. **IEEE Workshop on Mobile Computing Systems and Applications**, 1994.

da Silva, B. P. R. A. **Planeamento e implementação de um sistema de gestão da segurança da informação**. Universidade do Porto, 2011.

Tanembaum, A. S.; Wetheral, D. **Segurança de redes**. In: Redes de Computadores, p. 479–481, São Paulo, Brasil, 2011. Prentice Hall.

Umaparvathi, M.; Varughese, D. D. K. Evaluation of symmetric encryption algorithms for manets. **Computational Intelligence and Computing Research (ICCIC), International Conference**, p. 1–3, 2010.

Verma, O. P.; Agarwal, R.; Dafouti, D. ; Tyagi, S. Performance analysis of data encryption algorithms. **Electronics Computer Technology (ICECT), 3rd International Conference**, v.5, p. 399–403, 2011.

Zibideh, W.; Matalgah, M. Energy consumptions analysis for a class of symmetric encryption algorithm. **Radio and Wireless Symposium (RWS)**, p. 268–270, 2014.

Zuquete, A. **Segurança em redes informáticas**. In: Criptografia, p. 25–30, São Paulo, Brasil, 2013. FCA - Editora de Informática.