Uma Plataforma para Gerenciamento de Identidades de Software como Serviço em uma Infraestrutura como Serviço

Maicon Stihler e Altair Olivo Santin

Programa de Pós-Graduação em Informática – Pontifícia Universidade Católica do Paraná (PUCPR)
Rua Imaculada Conceição, 1155 – Prado Velho – CEP 80215-901 – Curitiba – PR
{stihler,santin}@ppgia.pucpr.br

Abstract. *Users of software as a service (SaaS) do not exist on the infrastructure as a service (IaaS) domain; this complicates the accounting and auditing of resource consumption. Consequently, developers of SaaS applications are tasked with managing the mapping of identities between SaaS and IaaS. The traditional approaches to identity federation look at the problem at only one level (eg. SaaS), thus we propose a platform to allow the mapping of identities between multiple levels in a transparent fashion. The result is reduced complexity for developers, transparency for users, and a more accurate accounting and auditing of resource usage.*

Resumo. *Usuários de software como serviço (SaaS) não existem no domínio da infraestrutura como serviço (IaaS), o que complica a contabilidade e auditoria do consumo de recursos. Consequentemente, os programadores de aplicações SaaS têm a tarefa de gerenciar o mapeamento de identidades entre SaaS e IaaS. As abordagens tradicionais para a federação de identidades olham para o problema em apenas um nível (e.g. SaaS), portanto, propomos uma plataforma para permitir o mapeamento de identidades entre os vários níveis de uma forma transparente. O resultado é a redução de complexidade para os desenvolvedores, a transparência para os usuários, e a contabilidade e auditoria mais precisas do uso de recursos.*

1. Introdução

O amadurecimento da modalidade de computação em nuvem conhecida como Infraestrutura como Serviço (*Infrastructure as a Service*, *IaaS*), trouxe novas possibilidades para os desenvolvedores de Software como Serviço (*Software as a Service*, *SaaS*). A elasticidade computacional oferecida por um ambiente de *IaaS* permite, de mesmo modo, que uma aplicação em nível *SaaS* seja redimensionada conforme a demanda de seus usuários aumenta[Badger, Grance, Patt-Corner e Voas 2011] .

Essa flexibilidade, no entanto, oferece desafios que não são solucionados facilmente pelas propostas existentes na literatura. A concepção em camadas do modelo de serviços de computação em nuvem desvincula a lógica da aplicação da infraestrutura, mas cria dificuldades de mapeamento entre as identidades dos usuários da aplicação *SaaS* e as identidades dos usuários que existem no ambiente *IaaS*. Isto é,

um usuário válido em um ambiente não é reconhecido no outro. Essa desintegração tem impacto direto, por exemplo, na capacidade do desenvolvedor *SaaS* de implementar políticas de autorização dinâmicas para seus usuários. Além disto, o desenvolvedor da aplicação tem dificuldades para rastrear identidades em nível de *SaaS* e adaptar as políticas ao consumo de infraestrutura que o usuário faz em nível de *IaaS*. Adicionalmente, o controle de acesso em nível de *IaaS* não conhece a associação entre as políticas de acesso e o usuário que são controladas em nível de *SaaS*. Estas dificuldades para gerenciar identidades entre as camadas trazem prejuízos para outros controles no ambiente de computação em nuvem.

Considerando que a natureza de cobrança dos serviços de *IaaS* precisa ser contabilizada pelo consumo individual de recursos, tem-se uma limitação de granularidade nas abordagens aplicadas atualmente. Em geral a contabilização de consumo é feita pelo contratante e não pelo usuário. Essa contabilidade fina permitiria a aplicação de políticas de autorização específicas para cada usuário, assim como políticas de cobrança adequadas para cada perfil. Por exemplo, um usuário que consuma mais recursos do que o previsto inicialmente pode entrar em um esquema de cobrança diferenciado e ter seu acesso garantido por políticas de controle de uso dinâmicas.

Ainda que os planos de cobrança utilizados em *SaaS* sejam diferentes dos utilizados em *IaaS*, a individualização do consumo de recursos da infraestrutura decorrentes do uso de serviços (*SaaS*) abre novas possibilidades de cobrança e controle. Em abordagens tradicionais os custos de provimento do serviço é rateada entre os usuários, devido à incapacidade de identificar as parcelas de uso de maneira indidualizada.

Para equacionar a dificuldade de mapeamento de identidades entre domínios diferentes, vários trabalhos propõem a federação de identidades ou autenticação distribuída: Shibboleth[Morgan, Cantor, Carmody, Hoehn e Klingenstein 2004], OpenSSO [Oracle Corporation 2010], SAML [OASIS 2011] e OpenID [OpenID Foundation 2007]. No entanto, estas abordagens operam em um único nível de abstração, ou seja, as identidades são utilizadas em contextos homogêneos (e.g. aplicações web). Neste contexto, observamos que as aplicações *SaaS* costumam possuir mecanismos de segurança diferentes dos utilizados em sistemas *IaaS* (e.g. os sistemas de autenticação utilizados). Assim, as propostas existentes para federação de identidades não podem atender satisfatoriamente e de forma transparente os três níveis de abstração de computação em nuvem.

Para contornar as limitações citadas anteriormente é proposto neste trabalho uma plataforma interposta entre a camada de *IaaS* e a aplicação *SaaS*, ocultando do desenvolvedor de *SaaS* os detalhes de autenticação e contabilidade em nível de *IaaS*. Deste modo, uma aplicação *SaaS* ganha a possibilidade de rastrear a utilização de recursos de baixo nível (camada de *IaaS*) individualmente, através do mapeamento das

identidades de seus usuários e dos processos executados em nome destes no ambiente *IaaS*.

Este artigo está organizado da seguinte maneira: na Seção 2 é apresentada uma revisão dos principais trabalhos relacionados; a Seção 3 discute a arquitetura da plataforma proposta. A Seção 4 apresenta uma discussão sobre a implementação de um protótipo e as considerações finais são apresentadas na Seção 5.

2. Trabalhos Relacionados

Shibboleth e OpenSSO oferecem compatibilidade com o profile web da *Security Assertion Markup Language* (SAML). Um dos fundamentos de SAML é prover identidades federadas para permitir que organizações que utilizem diferentes mecanismos de autenticação e autorização possam interoperar.

No OpenSSO/Shibboleth, quando um usuário tenta acessar um recurso web com seu navegador, o *service provider* (SP) exige informações sobre o usuário para decidir se o acesso será permitido. A requisição é redirecionada para um *site* executando o software de identificação *(Identity Provider*, IdP*)* da organização responsável, onde o usuário efetua seu login. O IdP redireciona o navegador de volta ao recurso protegido, embutindo algumas informações (i.e. assertivas) que comprovam que o usuário está autenticado. O SP valida estas assertivas e coleta mais informações sobre o usuário no IdP. Os atributos são repassados à aplicação Web para que a decisão de autorização seja tomada.

O OpenID possui uma abordagem similar ao Shibboleth. Sua estrutura básica é composta pelos sites consumidores de credenciais, chamados de *relying party* (RP), e pelos OpenID *providers* (i.e. provedores de identidade). A diferença principal entre a abordagem das opções anteriores está no fato de que o usuário decide quais RPs devem receber suas informações. Nos casos anteriores, o provedor de identidades é quem decide, através de suas políticas de segurança, quais são as partes que devem ter acesso às credenciais do usuário. OpenID é centrado no usuário e seu foco tem sido a aplicação para autenticação distribuída em *sites* web [OpenID Foundation 2007].

Essa abordagem, OpenID, contudo não é adequada para o ambiente discutido na introdução. Pois, além da proposta ser limitada a recursos web, sua função é garantir o *single sign-on* (i.e. autenticação única) entre domínios de segurança diferentes. Ainda que seja possível efetuar a autenticação de acesso a aplicações *SaaS* com o uso destas soluções, o mapeamento entre as identidades *SaaS* e *IaaS* ainda é uma questão em aberto.

Um sistema muito popular para o gerenciamento de autenticação em nível de sistema operacional é o *Kerberos Authentication System* [Steiner, Neuman, e Schiller 1988]. Através de uma série de mensagens cifradas, uma aplicação pode verificar que uma requisição é feita em nome de um determinado usuário. O *Kerberos* altera o protocolo de autenticação de Needham e Schroeder [Needham e Schroeder 1978] para

reduzir o número de mensagens de autenticação, através do uso de *timestamps*, um serviço para eliminar a necessidade de utilização subsequente de senhas – o serviço de *ticket-granting*, e a possibilidade de utilizar servidores de autenticação que existam em domínios diferentes daquele da aplicação alvo [Neuman e Ts'o 1994].

O *Kerberos* permite a centralização das informações de autenticação e pode ser utilizado em um ambiente de *IaaS*. Contudo ele não está integrado a autenticação da aplicação *SaaS*. Em nossa proposta, o modelo apresentado pelo *Kerberos* é utilizado como parte da plataforma de autenticação. Utilizamos as credenciais de nível *SaaS* para obtermos credenciais de baixo nível (e.g. utilizando *Kerberos*), através de mapeamentos entre credenciais de níveis distintos.

3. Arquitetura Proposta

Antes de apresentar a arquitetura proposta é importante definirmos as entidades da proposta.

- Usuário *SaaS*: é o consumidor da aplicação desenvolvida a partir da infraestrutura de *IaaS*. Sua identidade apenas é reconhecida pela aplicação *SaaS*, não sendo possível rastrear suas atividades por meios convencionais no ambiente de *IaaS*, pois o ambiente de *SaaS* executa virtualizado sobre o *IaaS*.
- Contratante de *IaaS*: é aquele que utiliza os recursos de *IaaS* para oferecer uma aplicação como serviço.
- Usuário de *IaaS*: são as identidades reconhecidas em nível de *IaaS*, isto é, os usuários reconhecidos pelos sistemas operacionais virtualizados.

Além disto, neste texto serão considerados recursos e serviços como definidos abaixo:

- Recursos: são os meios computacionais disponibilizados como infraestrutura (e.g. espaço de armazenamento, tempo de processamento, banda de rede etc.).
- Serviço: é a aplicação *SaaS* disponibilizada aos usuários *SaaS*, utilizando os recursos oferecidos em nível de *IaaS*.

O contratante de *IaaS* utiliza os recursos contratados para oferecer um serviço aos usuários de *SaaS*. O monitoramento da quantidade de consumo realizada no provimento do serviço é essencial, tendo em consideração que as políticas de cobrança de ambientes de *IaaS* costumam ser baseadas no consumo (i.e. *pay-as-you-go*).

Além disso, os recursos contratados podem estar sujeitos a diferentes tarifas, dependendo do perfil do recurso utilizado. Isso ressalta a importância da individualização do consumo de recursos por parte dos usuários *SaaS*. A abordagem simplista de rateio dos custos com todos os usuários não é satisfatória; a individualização do consumo em nível de Serviço, contudo é uma tarefa bastante complicada devido à falta de integração no esquema de gestão de identidades de usuários em nível de *SaaS* e *IaaS*.

Baseados nesses pressupostos, propomos uma arquitetura para mapeamento de identidades de usuários *SaaS* para *IaaS*. Um objetivo importante é ocultar do contratante de *IaaS* as especificidades do mapeamento e os mecanismos de implementação, permitindo que o mesmo concentre seus esforços no gerenciamento dos usuários *SaaS*. Como podemos ver na Figura 1, que apresenta os principais componentes da arquitetura proposta, existe uma intersecção entre os ambientes *SaaS* e *PaaS*. Isto decorre do fato de que, apesar de ser provido por uma plataforma, o *gateway* do serviço e seus interceptadores operam em nível de *SaaS*.

O componente denominado de *Usuário SaaS* conduz a comunicação com o *serviço*, usando para isso um protocolo específico (e.g. HTTP, via navegador web). As requisições são enviadas ao *gateway do serviço* – interface pública do *serviço* oferecida pelo contratante de *IaaS*; O *serviço* é a aplicação *SaaS*.

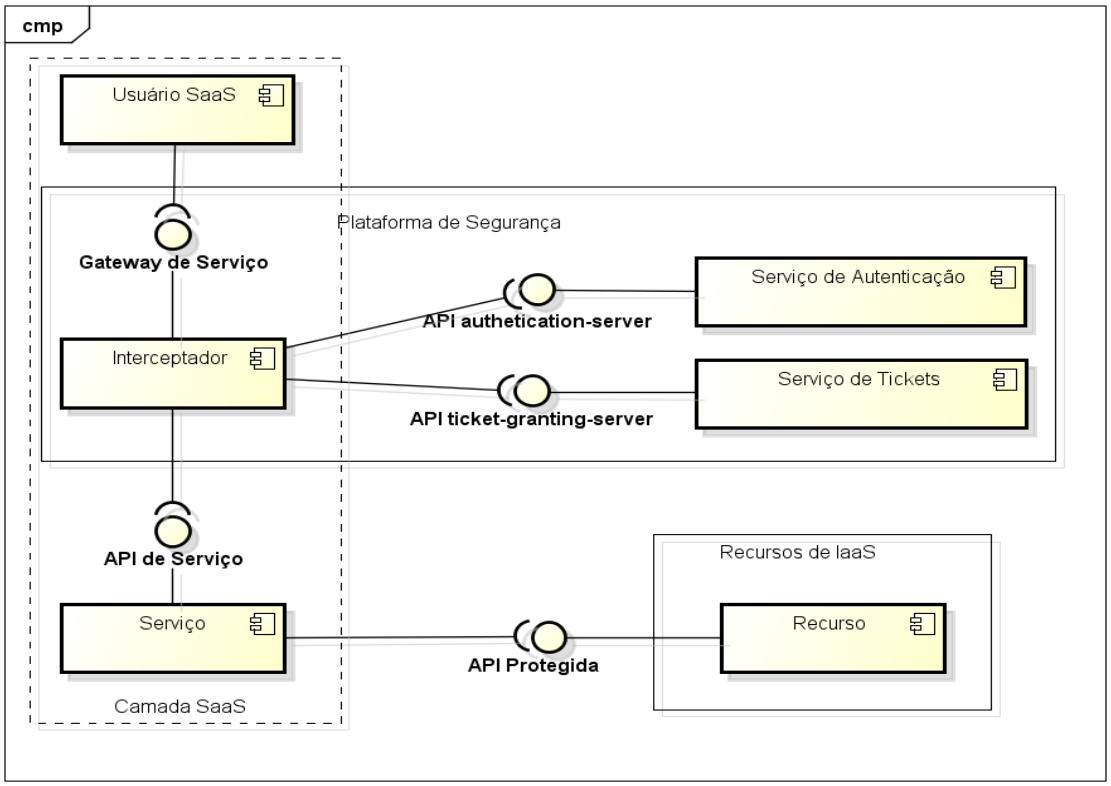

Figura 1: Componentes da Arquitetura

As requisições feitas pelo *cliente* devem conter as informações de autenticação requeridas pelo *serviço* (e.g. um par usuário/senha); estas informações são direcionadas ao *gateway do serviço* e são processadas por um *interceptador*. O interceptador extrai da requisição do *cliente* as informações de autenticação e desencadeia um procedimento de autenticação junto à infraestrutura. Isto é feito através do *serviço de autenticação*, no estilo do protocolo de autenticação *Needham-Schroeder*, usando a API *authentication-server*. Este processo verifica a existência de um mapeamento entre a credencial *SaaS* fornecida e uma credencial *IaaS*.

O *serviço de autenticação* utiliza informações de autenticação de alto-nível (i.e. usuários *SaaS*) para autenticar o acesso a recursos de *IaaS*. Uma vez comprovada a autenticidade do usuário *SaaS*, o *interceptador* utiliza a *API ticket-granting-server* para obter um *token* de identificação.

Este *token* de identificação é embutido na requisição do *serviço*; a requisição é então repassada à *API do Serviço* para o processamento esperado do *serviço*. O *serviço* deve então utilizar o *token* para acessar os recursos protegidos, utilizando para isso a *API Protegida* pelos mecanismos de segurança nativos do sistema operacional (e.g. o serviço pode instanciar novos processos, ou gravar informações em diretórios específicos). O sistema operacional pode então verificar a autenticidade da requisição e avaliar localmente a autorização do acesso.

3.1. Avaliação qualitativa da proposta

Com a utilização do esquema proposto, a primeira vantagem obtida é a centralização da atividade de gerenciamento do mapeamento de identidades no *serviço de autenticação*. Isto é, ao invés do contratante ser obrigado a administrar contas específicas para cada usuário *SaaS* em cada instância de sistema operacional criada, somente será necessário gerenciar uma base de autenticação central. Como resultado, o redimensionamento do *serviço* não incorre em custos administrativos extras para o contratante de *IaaS*; uma vez criado o mapeamento no *servidor de autenticação*, todas as máquinas virtuais instanciadas poderão receber requisições de *serviço* para a identidade cadastrada de maneira distribuída.

A segunda vantagem da proposta refere-se à possibilidade de rastrear as atividades específicas de cada usuário *SaaS* na infraestrutura. Como o sistema operacional geralmente oferece mecanismos para monitorar o consumo de um usuário local, o contratante pode utilizar tais mecanismos para contabilizar a utilização de recursos referentes ao usuário identificado pelo *token.*

Esta segunda possibilidade prove a capacidade de estabelecer políticas de autorização dinâmicas para cada usuário. Utilizando os princípios delineados no modelo de controle de uso [Park e Sandhu 2004], por exemplo, é possível disparar funções administrativas para entender a quantidade de recursos disponíveis para um usuário, ou executar tarefas de bilhetagem.

A unificação do sistema de identificação resulta em um controle fino para o contratante de *IaaS* que, tal qual os provedores de *IaaS*, pode oferecer um serviço elástico aos seus usuários. Como os usuários de *SaaS* estão isolados deste procedimento de gerenciamento unificado de credenciais, seu uso continua inalterado independente dos movimentos de redimensionamento da infraestrutura utilizada pelo serviço.

4. Implementação

Estudos preliminares foram efetuados para o desenvolvimento de um protótipo. Como ambiente de *IaaS*, o *middleware* adotado foi a versão de código aberto do software Eucalyptus [Eucalyptus Systems, Inc. 2011]. A tecnologia de virtualização utilizada é o

Xen Hypervisor [Citrix Systems 2011], sendo que as máquinas virtuais executam um sistema baseado em Linux, como por exemplo, a distribuição Debian [Debian Project 2011].

Como mencionamos anteriormente, o modelo para autenticação em nível de infraestrutura é baseado no protocolo de *Needham-Schroeder*, sendo que uma implementação do *Kerberos* é a solução mais apropriada; Para isso, é utilizada a distribuição de código aberto do Massachusetts Institute of Technology [Massachusetts Institute of Technology 2011].

O serviço, que desempenhará o papel de uma aplicação *SaaS*, será implementado com base no Apache Axis2 [Apache Foundation 2011], que é uma implementação da pilha de protocolos para serviços web. O Axis2 oferece a opção de implementação de interceptadores transparentes, de modo que podemos inspecionar a requisição SOAP [W3C 2011] para obtermos as informações de autenticação do usuário *SaaS*, e embutir cabeçalhos adicionais com o *token* obtido do serviço de autenticação (*Kerberos*).

O serviço, instanciado em uma máquina virtual, poderá então receber requisições dos usuários *SaaS*. O Interceptador irá efetuar a autenticação do usuário através das interfaces oferecidas pelo servidor *Kerberos*, embutindo o *token* obtido na mensagem SOAP destinada ao serviço. O serviço, por sua vez, utilizará esse *token* para instanciar um processo responsável por atender a requisição do usuário. Agentes de monitoramento instalados no sistema operacional virtualizado podem, então, utilizar o identificador do usuário para coletar os dados de consumo do processo criado.

Neste contexto, o serviço funciona como um despachante de requisições para processos criados em nome dos usuários *SaaS* – a criação de processos pode ser executada através do comando *ksu(kerberized version of the su*), que permite ao serviço assumir a identidade do usuário contida no *token* para fins de instanciar o processo. Através de utilitários como o *LiSt Open Files* (*lsof*) os agentes podem obter as métricas referentes a cada usuário (e.g. tempo de processamento, espaço utilizado em disco).

A Figura 2 apresenta uma visão geral dos componentes utilizados para implementar o protótipo da plataforma proposta. Um servidor executando o software Eucalyptus gerencia as máquinas virtuais em execução na infraestrutura (i.e. são os recursos *IaaS* da Figura 1). Cada instância destinada a atender requisições de serviço possui os componentes da Figura 2 pré-instalados. As credenciais dos usuários *SaaS* são cadastradas em uma base de identidades utilizadas pelo servidor *Kerberos*, que emite *tokens* de autorização utilizados pelo serviço para criar processos em nome dos usuários *SaaS*; O servidor *Kerberos* realiza o papel do servidor de autenticação e de *tickets* da plataforma de segurança da Figura 1.

O Cliente SOAP é a aplicação que realiza as requisições desejadas pelo Usuário *SaaS*. As requisições são capturadas por um módulo interceptador implementado no Apache Axis2, para realizar as validações necessárias (i.e. autenticação do usuário *SaaS*

e obtenção de *token Kerberos* para uso no sistema operacional). O Serviço instancia um processo para atender as requisições do usuário, e este processo de usuário efetua a interação com o sistema operacional para usar os recursos necessários. O processo de usuário executa sob a identidade fornecida pelo *token Kerberos*.

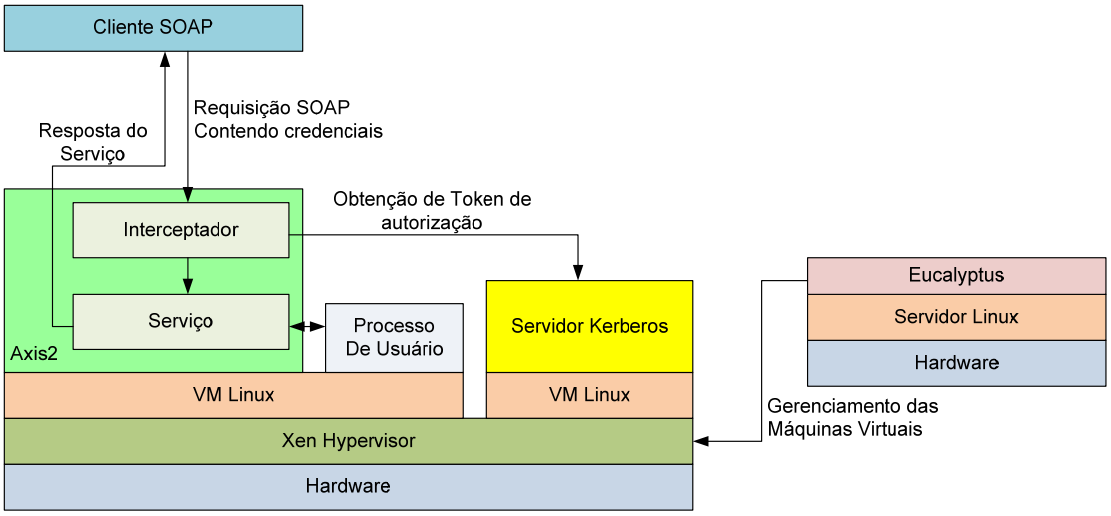

Figura 2: Componentes do Protótipo

5. Considerações Finais

Neste trabalho foi apresentada uma plataforma que permite o mapeamento entre identidades de usuários de aplicações *SaaS* e identidades em nível de *IaaS*. A integração das identidades provê ao desenvolvedor de aplicações *SaaS* a possibilidade de rastrear a utilização de recursos de seus usuários em baixo nível (nível de mecanismo), através da contabilidade individual (de granularidade fina).

Como resultado da proposta, o desenvolvedor obtém outras vantagens como a capacidade de aplicação de políticas dinâmicas de autorização baseadas no consumo de cada usuário; a cobrança diferenciada e individualizada para cada usuário *SaaS*, evitando o rateamento indiscriminado de custos de consumo.

A utilização de interceptadores entre o nível de *SaaS* e os sistemas em nível de *IaaS* permite a integração, de forma transparente ao usuário *SaaS*, dos sistemas de autenticação da aplicação e do sistema operacional virtualizado. Ou seja, o aumento no controle oferecido não incorre em nenhum inconveniente para o usuário final.

As discussões sobre a implementação de um protótipo sugerem que a proposta é realizável com componentes de software amplamente disponíveis. As tecnologias sugeridas são maduras e testadas em produção, o que sugere uma robustez inerente à proposta.

Referências

Apache Foundation (2011), Apache Axis2, Acessível em: http://axis.apache.org/axis2/java/core/, referenciado em 22 de Setembro de 2011.

Badger, L., Grance, T., Patt-Corner, R. e Voas, J. (2011), Cloud Computing Synopsis and Recommendations, disponível em: http://csrc.nist.gov/publications/drafts/800-146/Draft-NIST-SP800-146.pdf, Referenciado em 22 de Setembro de 2011.

Citrix Systems (2011), Inc. Xen Hypervisor. Disponível em: http://xen.org/products/xenhyp.html, referenciado em 22 de Setembro de 2011.

Debian Project (2011), Debian GNU/Linux. Disponível em: http://www.debian.org, referenciado em 22 de Setembro de 2011.

Eucalyptus Systems, Inc. (2011), Eucalyptus Cloud Platform. Disponível em: http://www.eucalyptus.com/, referenciado em 22 de Setembro de 2011.

Massachusetts Institute of Technology (2011), *Kerberos*: The Network Authentication Protocol, Disponível em: http://web.mit.edu/Kerberos/, referenciado em 22 de Setembro de 2011.

Morgan, R. L., Cantor, S., Carmody, S., Hoehn, W. e Klingenstein (2004), K. Federated Security: The Shibboleth Approach, EDUCAUSE Quarterly, vol. 27, no. 4 pp 12-17.

Needham, R. M. e Schroeder, M. D. (1978), Using encryption for authentication in large networks of computers, Commun. of the ACM. vol. 21, no. 12, pp 993-999. [Needham]

Neuman, B.C. e Ts'o, T. (1994), *Kerberos*: An Authentication Service for Computer Networks, IEEE Communications, vol. 32 no. 9, pp 33-38.

OpenID Foundation (2007), OpenID Authentication 2.0 Disponível em: http://openid.net/specs/openid-authentication-2_0.html, referenciado em 22 de Setembro de 2011.

Oracle Corporation (2010), OpenSSO Architecture Overview, documentação. http://wikis.sun.com/display/OpenSSO/OpenSSO+Architecture+Overview. Referenciado em 22 de Setembro 2011.

Organization for the Advancement of Structured Information Standards (OASIS) (2011), Security Assertion Markup Language, v. 2.0. http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf, 2005. Referenciado em 22 de Setembro de 2011.

Park, J. e Sandhu, R. (2004), The $UCON_{ABC}$ Usage Control Model. ACM Trans. Inf. Syst. Secur., vol. 7 no. 1, pp 128-174.

Steiner, J.G., Neuman, B.C., e Schiller, J.I. (1988), *Kerberos*: An Authentication Service for Open Network Systems. In Proceedings of the Winter 1988 Usenix Conference.

The World Wide Web Consortium (W3C) (2011), SOAP Version 1.2 Part 1 Messaging Framework (Second Edition). Disponível em http://www.w3.org/TR/soap12-part1, referenciado em 22 de Setembro de 2011.