

UNIVERSIDADE FEDERAL DE SANTA CATARINA

Proposta de um modelo de biblioteca cliente para o Shibboleth:

Facilitando a adaptação das aplicações

Max Alexandre Zanelato

Florianópolis - SC

2014/2

UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

Proposta de um modelo de biblioteca cliente para o Shibboleth:

Facilitando a adaptação das aplicações

Max Alexandre Zanelato

Trabalho de Conclusão de Curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Ciências da Computação.

Florianópolis – SC

2014/2

Max Alexandre Zanelato

Proposta de um modelo de biblioteca cliente para o Shibboleth:

Facilitando a adaptação das aplicações

Trabalho de Conclusão de Curso apresentado
como parte dos requisitos para obtenção do
grau de Bacharel em Ciências da Computação.

Orientadora: Carla Merkle Westphall.

Coorientador: Rafael Weingärtner.

Banca examinadora

Carlos Becker Westphall

Daniel Ricardo dos Santos

DEDICATÓRIA

Dedico este trabalho a minha mãe Roseli Rabelo, que esteve comigo até o final, aos meus amigos, que me aturaram falando sobre o trabalho e as várias vezes que neguei convite para ficar escrevendo, a minha corretora ortográfica Ana Carolina Santoro, que sempre esteve ao meu lado, e a todos que participaram direta ou indiretamente possibilitando realizá-lo.

Sumário

1. Introdução	13
2. Objetivos	15
2.1. Objetivo Geral	15
2.2. Objetivos Específicos	15
3. Gerenciamento de Identidades	16
3.1. Identidade Digital	16
3.2. Definição de Gerenciamento de Identidades	19
3.3. Modelos	20
3.3.1. Tradicional.....	20
3.3.2. Centralizado	22
3.3.3. Federado.....	23
3.3.4. Centrado no Usuário	24
3.4. Autenticação Única (<i>Single Sign-On – SSO</i>)	25
3.5. Encerramento Único de Sessão	28
3.6. Requisitos e Fatores Críticos de Sucesso	30
3.7. Federação	33
3.8. Ferramenta: Shibboleth	35
3.8.1. Definição	35
3.8.2. Provedor de Identidades Shibboleth	36

3.8.3. Where Are You From	39
3.8.4. Provedor de Serviços	40
3.8.5. Funcionamento.....	42
3.8.6. Integração da Aplicação com Shibboleth SP	45
3.9. Tecnologias Relacionadas	48
3.9.1. SAML	48
3.9.2. CAS.....	53
3.9.3. Spring Security	55
4. Proposta de um modelo de biblioteca cliente para o Shibboleth: Facilitando a adaptação das aplicações	60
4.1. Descrição da proposta	60
4.2. Configuração.....	63
4.2.1. Configuração da Aplicação	63
4.2.2. Configuração do Ambiente.....	78
4.3. Execução	82
5. Considerações Finais	86
6. Trabalhos Futuros.....	88
7. Referências.....	89

Lista de Ilustrações

Figura 1: Modelo Tradicional	21
Figura 2: Modelo Centralizado	22
Figura 3: Modelo Federado	23
Figura 4: Modelo Centrado no Usuário	25
Figura 5: Funcionamento do perfil de autenticação única	27
Figura 6: Modelo de autenticação única.....	28
Figura 7: Exemplo de funcionamento do perfil <i>Single Logout (SLO)</i>	30
Figura 8: Componentes do Provedor de Identidades	37
Figura 9: Tela Inicial da Federação Chimarrão	40
Figura 10: Componentes do Provedor de Serviço.....	41
Figura 11: Passos do funcionamento do Shibboleth	43
Figura 12: Acesso direto ao recurso.....	47
Figura 13: Relação entre conceitos do SAML	49
Figura 14: Arquitetura SAML.....	51
Figura 15: Fluxo de autenticação no CAS	54
Figura 16: Dependência do framework Spring Security	56
Figura 17: Código a ser inserido para uso da extensão do Spring Security	59
Figura 18: Aplicação atuando como um provedor de serviços	61
Figura 19: Dependência do Spring Security	65
Figura 20: Exemplo de configuração do web.xml.....	68
Figura 21: Filtro para interceptar requisições	69
Figura 22: Arquivo xml de configuração da aplicação	70
Figura 23: Áreas segura e não segura do arquivo securityContext.xml	72

Figura 24: Código de erro SAML.....	72
Figura 25: Código de geração automático dos metadados da aplicação	73
Figura 26: Atributos para geração dos metadados da aplicação.....	74
Figura 27: Código para indicar os metadados do IdP	75
Figura 28: Bloco de logout local	76
Figura 29: Manipuladores do logout global	77
Figura 30: Comandos para geração de um certificado auto assinado	78
Figura 31: Informações requisitadas para geração de um certificado	79
Figura 32: Comando para extrair chave pública de uma URL.....	80
Figura 33: Declaração comentada no server.xml do Tomcat	80
Figura 34: Alterações no arquivo server.xml do Tomcat	81
Figura 35: Lista dos provedores de identidade.....	82
Figura 36: Página de autenticação do IdP	83
Figura 37: Atributos do usuário	84

Lista de Tabelas

Tabela 1: Exemplo de Identidade Digital	17
--	----

Lista de Reduções

AA	Attribute Authority
AAP	Attribute Acceptance Policies
ACS	Assertion Consumer Service
API	Application Programming Interface
AR	Attribute Requester
ARP	Attribute Release Policies
CAFe	Comunidade Acadêmica Federada
CAS	Central Authentication Service
CPF	Certidão de Pessoa Física
CTC	Centro Tecnológico
DS	Discovery Service
HS	Handle Service
HTTP	Hypertext Transfer Protocol
IdP	Identity Provider (Provedor de Identidades)
INE	Departamento de Informática e Estatística
JSP	JavaServer Pages
MVC	Model-View-Controller
RM	Resource Manager
RNP	Rede Nacional de Ensino e Pesquisa
SAML	Security Assertion Markup Language
SLO	Single Logout

SP	Service Provider (Provedor de Serviços)
SSL	Secure Sockets Layer
SSO	Single Sign-on (Autenticação única)
URL	Universal Resource Locator
VPN	Virtual Private Network
WAYF	Where Are You From

Resumo

As informações de indivíduos ou organizações são representadas eletronicamente por identidades digitais. Essas identidades possuem informações particulares a serem protegidas e não devem ser reveladas às pessoas não autorizadas.

Para que problemas de segurança, como acesso às informações não autorizadas, não aconteça ou que seja garantida a integridade, faz-se necessário a utilização de um mecanismo que realize esse controle. Um dos métodos para a resolução do problema seria a aplicação de um sistema de gerenciamento de identidades.

A tarefa de gerenciamento de identidades consiste na administração, organização interna e central de dados sobre terceiros, integrados a políticas, processos de negócio e tecnologias que permitem acesso a recursos e serviços locais de forma segura. O Shibboleth é uma opção de software para criar ambientes federados trocando informações entre provedores de serviços e provedores de identidades.

O Shibboleth trabalha com modelo de identidades federadas que estabelece uma relação de confiança entre os agentes da federação facilitando a troca de informações e acesso a recursos.

Palavras-chave: Provedor de Identidades. Integridade. Privacidade. Segurança. Gerenciamento de Identidades. Federação. Provedor de Serviço. Spring Security.

SAML.

1. Introdução

As empresas utilizam uma variedade de aplicações que facilitam as atividades rotineiras, tais como sistemas de e-mail, sistemas de gerenciamento de relações de clientes e sistemas de planejamento de recursos empresarial. Contudo, as empresas enfrentam um problema não trivial: o problema de gerenciamento de identidade de usuários.

De acordo com Jøsang *et al.* [5], o gerenciamento de identidades consiste de um sistema integrado de políticas, processos de negócios e tecnologias que permite às organizações e indivíduos o tratamento e manipulação de identidades.

Como HARRIS *et al.* [32] descreve, mecanismos de gerenciamento de identidades procuram facilitar o uso de credenciais por parte dos usuários, principalmente através de serviço de acesso único (*Single Sign-On*), que oferece a possibilidade de utilizar diversos serviços com apenas uma autenticação. As ferramentas de gerenciamento de identidades também auxiliam a criação de ambientes federados, que permitem o estabelecimento de relações de confiança entre diferentes organizações, assim usuários de uma organização podem acessar recursos externos sendo autenticados e autorizados no seu domínio de origem.

Um processo de autenticação entre federações exige que o mecanismo de gerenciamento de identidades apresente duas entidades distintas: o provedor de identidades, responsável por gerenciar os atributos que compõem uma identidade e realizar os processos de autenticação de um usuário e disseminação de atributos relacionados a uma identidade; e provedor de

serviços, responsável pelo processo de autorização do recurso solicitado, com base nos atributos recebidos do provedor de identidades.

A utilização de sistemas de gerenciamento de identidades possibilita que pessoas confirmem sua identidade e, então, dar um nível de acesso adequado a cada uma delas.

A gerência da segurança dos recursos geralmente está presente na camada do provedor de serviços e quando liberado pelo sistema, o usuário realiza o seu acesso. Em alguns ambientes característicos, os recursos se encontram fora do provedor de serviços do Shibboleth, então é necessário redirecionar o usuário ao recurso requisitado, como por exemplo uma aplicação ou serviço desenvolvido, que não utiliza o *plugin Apache Webserver*. O sistema pode se tornar vulnerável a ataques externos por deixar a responsabilidade de implementar todo o mecanismo de segurança nas mãos do desenvolvedor.

Este trabalho apresenta uma proposta que visa solucionar o problema destacado, criando um modelo para bibliotecas clientes que possam ser implementadas em diferentes linguagens. A adaptação de uma aplicação ao modelo de identidades federadas do Shibboleth se torna menos complexo e, conseqüentemente, padroniza-se as soluções aplicadas do lado do recurso a ser protegido.

2. Objetivos

2.1. Objetivo Geral

O objetivo geral desse trabalho é analisar as soluções existentes de gerenciamento de identidades e prover uma nova solução para um caso em específico que foge a arquitetura original.

2.2. Objetivos Específicos

Os objetivos específicos que podem ser listados para este trabalho são:

- Analisar a instalação e configuração do Shibboleth;
- Especificar possíveis problemas existentes na plataforma Shibboleth;
- Propor um modelo para sanar possíveis problemas existentes;
- Validar o modelo proposto com uma implementação em Java.

3. Gerenciamento de Identidades

O gerenciamento de acesso a recursos vem se tornando um grande problema com o crescente número de usuários e recursos computacionais que as organizações possuem. Esse cenário de crescimento e competitividade exige das organizações um gasto enorme de recursos para estabelecer e manter a segurança de seus negócios.

O gerenciamento de identidades é um campo da segurança da informação que se preocupa com o gerenciamento de usuários e seus dados envolvendo autenticação, autorização e liberação de atributos. O gerenciamento de identidades traz benefícios como redução de custos e esforços redundantes, otimizando o desempenho do usuário; acréscimo significativo na segurança da informação; e uso adequado e seguro do sistema.

3.1. Identidade Digital

Segundo BERTINO *et al.* [21], a identidade digital é um conjunto de características que podem estabelecer a identidade de um indivíduo. A **Tabela 1** apresenta pares de valores que poderiam ser utilizados para criar uma identidade digital.

Atributo	Valor
ID	101
Nome	Maria das Neves
Documento Pessoal	20401020

E-mail	maria@email.com
Matrícula	1010101

Tabela 1: Exemplo de Identidade Digital

Essa representação digital normalmente é composta por três diferentes tipos de dados: identificador, credenciais e atributos.

- **Identificadores:** Informação que identifica uma pessoa, local ou coisa de maneira exclusiva, única e não duplicada em um dado contexto. Impressões digitais e contas de e-mail são únicos, podem atuar como identificadores e podem ser considerados únicos em uma determinada identidade;
- **Credenciais:** É tipicamente uma coleção de atributos e asserções de identidade sobre um sujeito específico emitido por um provedor de identidades. O emissor é crucial para a decisão de aceitação da credencial e, possivelmente, a validade do conteúdo. No presente conteúdo, refere-se a integridade de assegurar que a credencial não tenha sido violada e que técnicas, como a assinatura digital, podem ser usadas para garantir integridade da credencial emitida;
- **Atributos:** Um conjunto de dados que descreve as características de um indivíduo. Inclui informações fundamentais para identificar indivíduos, suas preferências e as informações geradas como resultado de suas atividades. Alguns exemplos são nome de famílias, domicílios, idades, sexo, papel, títulos, afiliações e reputação.

A identidade digital consiste na combinação de subconjuntos chamados de identidades parciais, que estão relacionados a contextos específicos. Dependendo do contexto e da situação, uma pessoa pode ser representada por uma identidade parcial diferente. Por exemplo, se o contexto for uma universidade, então a identidade parcial seria composta pelo nome e disciplinas que um aluno cursa, já, se for uma empresa, a identidade parcial poderia ser representada pelas funções, privilégios e responsabilidades da pessoa.

Uma identidade digital normalmente passa pelo seguinte ciclo de vida – GOLDBERG *et al.* [20]:

- Fornecimento: O processo de registro do usuário e criação de sua identidade pode incluir uma prova de identidade e atribuição de privilégios. É o momento em que o usuário fornece suas informações pessoais e essas são relacionadas à identidade sendo criada.
- Propagação, uso e manutenção: Fase em que a identidade criada se encontra na maior parte do tempo. Essa identidade deve ser propagada para que possa ser utilizada nos sistemas em que for solicitada e deve ser mantida com um armazenamento seguro até que sua remoção seja solicitada.
- Destruição: Nesse momento a identidade é destruída e não poderá mais ser utilizada. É importante que não só a identidade, mas também contas relacionadas, atributos e informações atreladas, bem como privilégios sejam todos removidos.
- Auditoria: Pode ser necessária durante qualquer momento no ciclo de vida de uma identidade, por isso é importante que sejam gerados *logs* de toda operação realizada com ou sobre a identidade.

Como computadores e redes estão sujeitos a invasões e falhas de segurança, pode haver o roubo de identidades digitais causando prejuízos. Manter a privacidade, isto é, manter a habilidade do indivíduo proteger informações sobre si mesmo, como comentado no trabalho de GOLDBERG *et al.* [20], é uma tarefa complexa na Internet.

3.2. Definição de Gerenciamento de Identidades

O gerenciamento de identidades consiste em manter a integridade da identidade através dos seus ciclos de vida a fim de tornar as identidades e seus respectivos dados disponíveis para serviços de forma segura. A sua importância cresce conforme crescem os serviços que precisam utilizar autenticação e controle de acesso de usuários – Federação CAFe *et al.* [22].

Para análise de um sistema gerenciador de identidades, é importante ver quem são os atores. Os seguintes atores existem num gerenciamento de identidades:

- **Usuário:** Pode ser um indivíduo ou empresa que requisitará o serviço protegido, podendo acessá-lo após o processo de autenticação e autorização com base na análise de seus atributos;
- **Identidade:** Conjunto de características de um usuário, como nome, endereço e CPF;
- **Provedores de identidade (IdP – Identity Provider):** Responsáveis por associar uma identidade a um usuário, após um processo de autenticação. O usuário recebe uma credencial que é enviada, reconhecida e analisada pelos provedores de serviços garantindo ou não acesso ao recurso;

- Provedores de serviço (SP – Service Provider): Oferecem serviços ou recursos a usuários autorizados com base em seus atributos. São os parceiros de comunicação que interagem com outras organizações, quando usuários acessam determinados serviços, o SP precisa suportar o gerenciamento de identidades desses usuários.

3.3. Modelos

Há diferentes tipos de gerenciamento de identidades. Jøsang *et al.* [5] apresentou e explicou quatro tipos principais com relação à arquitetura: tradicional, centralizado, federado e centrado no usuário.

3.3.1. Tradicional

No modelo tradicional, que é o mais difundido atualmente, a identidade do usuário é gerenciada por cada provedor de serviço, que também atua como um provedor de identidade, conforme ilustrado na **Figura 1**, assim cabendo ao usuário criar sua identidade para cada serviço.

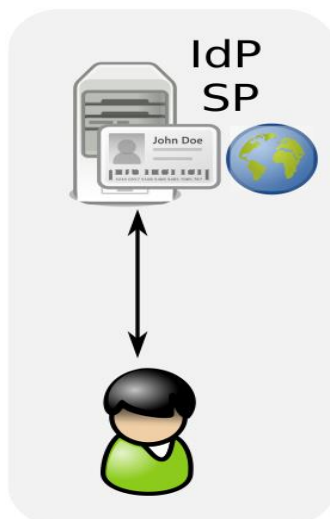


Figura 1: Modelo Tradicional

Fonte: Joni da Silva Fraga *et al.* [40]

Esse modelo, embora muito utilizado, é considerado pouco eficiente tanto do ponto de vista dos usuários quanto dos provedores de serviço. Gerenciar inúmeras identidades é algo custoso, pois é necessário fornecer um conjunto de atributos para cada servidor e, em diversas vezes, atributos que já foram informados para outro provedor.

O custo gerado para os usuários também reflete nos provedores de serviço, a tarefa repetitiva de sempre fornecer as mesmas informações no momento da criação de sua identidade, faz com que usuários não sejam tão fiéis no preenchimento de atributos cruciais para acessar o recurso fornecido pelo provedor. Cada provedor de serviço desse modelo é isolado, ou seja, não compartilham informações entre si.

3.3.2. Centralizado

O modelo centralizado surgiu como uma proposta de solução para a inflexibilidade do modelo tradicional. Centraliza a autenticação e autorização de diversos provedores de serviços em um único provedor de identidades, como ilustrado na **Figura 2**. Com esse modelo, surge o conceito de autenticação única (*Single Sign-On - SSO*), ou seja, permite aos usuários se autenticarem apenas uma vez no provedor de identidades e, então, acessar os serviços de modo transparente sem a necessidade de “reautenticação”.

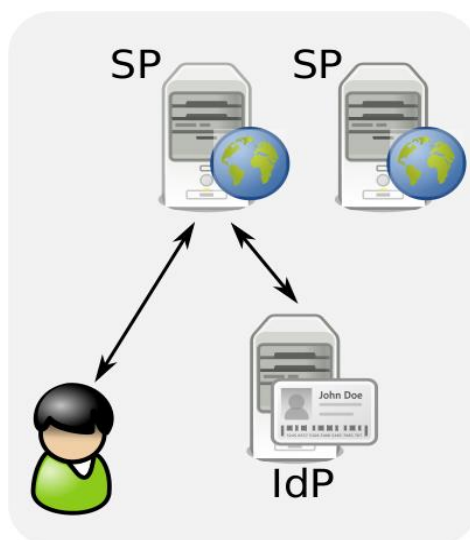


Figura 2: Modelo Centralizado

Fonte: Joni da Silva Fraga *et al.* [40]

Os riscos presentes nesse modelo podem levá-lo à falha. Como primeiro risco, pode-se citar o grande poder informacional concentrado no provedor de identidades que detém todos os atributos do usuário e pode utilizá-los de várias formas. Como segundo risco, pode-se mencionar a rigidez da arquitetura quando

há presença de políticas de segurança distintas. Por último, em caso de vários repositórios de atributos, requer alteração nas políticas de segurança em vigor.

Como vantagem desse modelo, temos a simplificação de gestão de infraestrutura técnica de autenticação, rapidez na resolução de problemas técnicos, diminuição da existência de hardware/software de suporte ao sistema de autenticação, redução na emissão e gestão de certificados de protocolos criptográficos, decréscimo de possíveis pontos de falha e de possíveis vulnerabilidades e simplificação na utilização para o usuário final.

3.3.3. Federado

Para aumentar o escopo utilizando o modelo centralizado, surgiu o modelo de identidade federada, que está fundamentado na distribuição do papel de entidade autenticadora a diversos servidores de identidade espalhados através de diversos domínios administrativos, como universidades, escolas, empresas ou governos. A **Figura 3** demonstra o modelo federado.

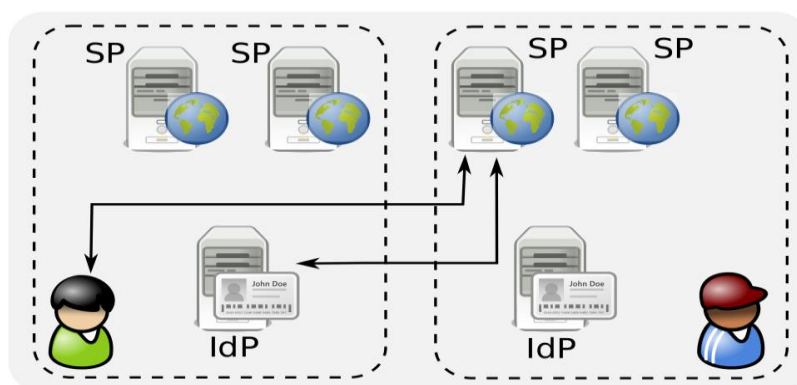


Figura 3: Modelo Federado

Fonte: Joni da Silva Fraga *et al.* [40]

Esse modelo visa aperfeiçoar a troca de informações relativas a uma identidade, evitando as inconsistências e redundâncias presentes no modelo tradicional, se baseando em uma relação de confiança chamada federação. Na federação há um acordo entre provedores de identidade e provedores de serviço que garante que uma identidade fornecida em um domínio passe a valer no outro, possibilitando a autenticação única.

Caso aconteça um ataque externo não permitido ao provedor de identidades, a arquitetura toda não será comprometida, porém, os usuários que utilizam tal provedor de identidade estarão com seus dados em risco.

3.3.4. Centrado no Usuário

O modelo centrado no usuário, diferente dos demais, é baseado na ideia de dar ao usuário total controle sobre suas identidades digitais, principalmente o controle sobre o tipo de informações que serão disponibilizadas a um determinado serviço. Apesar de ser um modelo diferente, suas principais implementações tem como base outros modelos já descritos Jøsang *et al.* [5].

As principais propostas de implementação desse modelo, mostrado na **Figura 4**, defendem a utilização de um servidor de identidade junto do usuário, como em um celular ou um *smartcard*, que ao receber a autenticação do usuário, é encarregado de liberar as informações do usuário para os provedores de serviço, respeitando totalmente as configurações de privacidade do usuário.

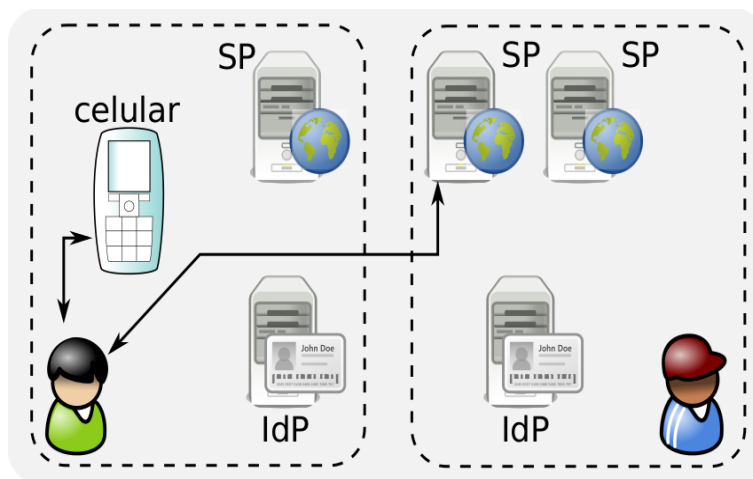


Figura 4: Modelo Centrado no Usuário

Fonte: Joni da Silva Fraga *et al.* [40]

3.4. Autenticação Única (*Single Sign-On – SSO*)

Uma das aplicações mais comuns de gerenciamento de identidades federadas é a autenticação única, que permite a um sujeito realizar o processo de autenticação uma única vez, junto ao seu provedor de identidades e desfrutar das credenciais obtidas para acessar diferentes serviços.

Uma autenticação única pode acontecer em cenários diferentes, como na web ou via aplicação. Shibboleth *et al.* [27] utiliza o cenário da web e assume que o usuário utilize um navegador comum para se comunicar tanto com o provedor de serviços quanto com o provedor de identidades. O perfil define, basicamente, quais ações um provedor de serviços deve fazer para iniciar um contexto de segurança com um sujeito com base em informações de segurança emitidas pelo provedor de identidades.

A **Figura 5** ilustra o funcionamento do perfil de autenticação única e os seguintes passos esquematizam a sua execução:

1. O usuário começa acessando um recurso protegido. O gerenciador de segurança do recurso monitorado determina se o usuário tem uma sessão ativa e, descobrindo que não, redireciona-o para o provedor de identidades a fim de começar um processo de autenticação única;
2. O usuário atinge o provedor de serviço que prepara uma requisição de autenticação e envia para o provedor de identidades. O software do provedor de serviço é geralmente instalado no mesmo servidor que o recurso;
3. Quando a requisição chega ao provedor de identidades, o provedor verifica se há uma sessão existente. Se tiver, vai para o próximo passo, senão o provedor de identidades questiona o usuário quanto as suas credenciais e o envia para o próximo passo.

Essa etapa é dependente da implementação, mas pode-se fazer uso dos documentos de metadados ou do perfil de descoberta de provedores de identidade Westin *et al.* [23];

4. Depois de autenticar o usuário, o provedor de identidades prepara uma resposta de autenticação e envia de volta ao provedor de serviço;
5. Quando o usuário entrega a resposta do provedor de identidades ao provedor de serviços, o SP validará a resposta, criará uma sessão para o usuário e construirá informações obtidas a partir da resposta (ex.: identificador do usuário) disponível para proteger o recurso. Depois disso, o usuário é enviado ao recurso;
6. Como no passo 1, o usuário está agora tentando novamente acessar o recurso protegido, mas dessa vez o usuário tem uma sessão com um

contexto de segurança válido, o recurso requisitado será entregue ao usuário.

Em uma continuação do cenário da **Figura 5**, se o sujeito for acessar outro provedor de serviço para o qual também não haja um contexto de segurança, os mesmos passos serão executados. No entanto, no passo 4, o provedor de identidades poderá responder ao provedor de serviços sem precisar pedir dados ao sujeito, utilizando o contexto de segurança criado quando o sujeito tentou acessar o primeiro provedor de serviço, como apresentado na **Figura 6**.

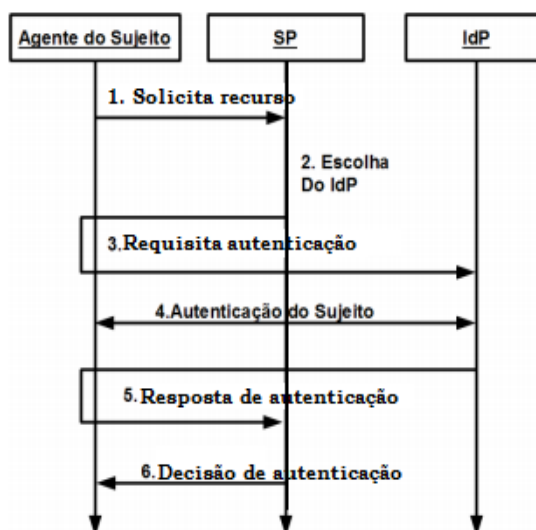


Figura 5: Funcionamento do perfil de autenticação única

Fonte: Joni da Silva Fraga *et al.* [40]

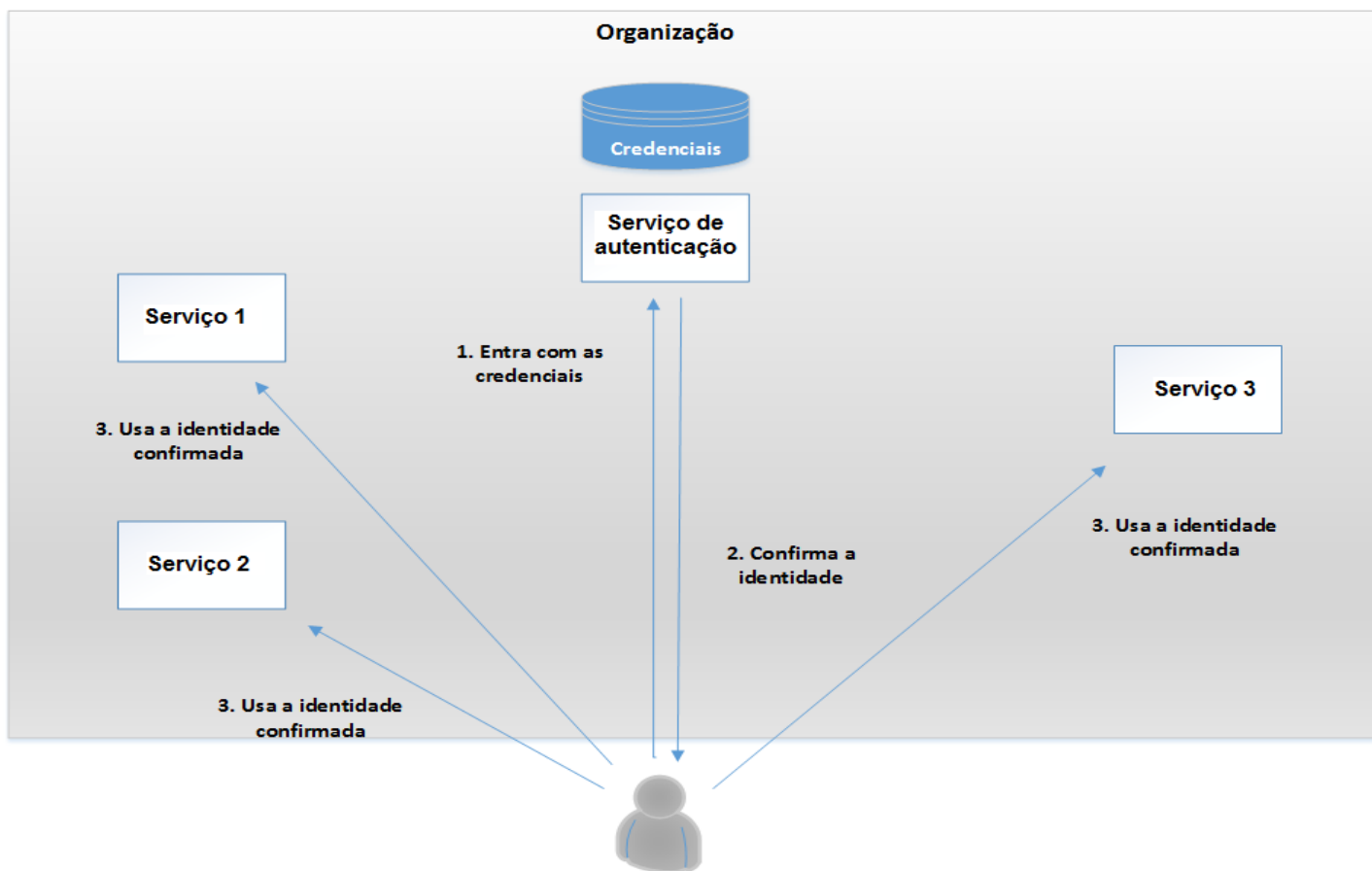


Figura 6: Modelo de autenticação única

3.5. Encerramento Único de Sessão

Quando há autenticação em um provedor de identidades, há o estabelecimento de conexão por meio de *cookie* de sessão, que são persistências temporárias que trocam informações de estado entre o lado cliente e o lado servidor. Além disso, quando é feito o acesso em um provedor de serviços, são consumidas asserções emitidas pelo provedor de identidades. Após a utilização, o indivíduo pode querer finalizar a sessão com um provedor de serviços específico ou pode finalizar todas as sessões de uma só vez. Para

o segundo caso, é definido um perfil de encerramento único de sessão (*Single Logout* – SLO) - Jøsang *et al.* [5].

A **Figura 7** demonstra o funcionamento de perfil de encerramento único (SLO). Em um cenário qualquer de requisição de encerramento, os seguintes passos são executados:

1. Em um determinado momento, o indivíduo indica para o provedor de serviços que deseja encerrar todas as sessões. O provedor de serviços em questão enviará uma mensagem de pedido de encerramento, definida no protocolo SLO, para o provedor de identidades responsável pela autenticação do indivíduo, através do agente do sujeito (por exemplo, o redirecionamento HTTP no navegador);
2. O provedor de identidade identifica todos os participantes através da sessão do sujeito;
3. O provedor de identidade envia uma mensagem para esses participantes pedindo o encerramento;
4. Os participantes encerram a sessão localmente e devolvem ao provedor de identidades mensagens de encerramento;
5. O provedor de identidade, por sua vez, retorna uma mensagem de resposta de encerramento para o provedor de serviço que iniciou a desconexão usando o mesmo mecanismo da requisição.

deve ser possível para um serviço obter qualquer informação sobre o usuário além daquelas consentidas.

Na interoperabilidade, as identidades dos usuários devem ser representadas em um formato comum de forma a permitir que a mesma possa ser compreendida e validada mesmo diante de múltiplos domínios administrativos e de segurança. A interoperabilidade é necessária para tratar vários aspectos de heterogeneidade entre membros da rede. Os membros da rede incluem as diversas plataformas computacionais, as várias políticas sancionadas e as diferentes tecnologias de segurança adotadas. Um suporte a essa heterogeneidade é essencial para garantir que a rede possa atender o maior número possível de participantes.

Por fim, cabe ao sistema gerenciar a criação, uso, manutenção e destruição de identidades e informações relacionadas às identidades. Deve oferecer as mesmas funções para identidades parciais para que possam ser utilizadas em diferentes contextos.

Chadwick *et al.* [7] elaborou uma lista com sete leis julgadas necessárias para que o sucesso seja alcançado:

- **Consenso e controle de usuário:** O sistema só deve liberar informações sobre um usuário após seu consentimento. A confiança do usuário no sistema é crucial para sua manutenção no mercado;
- **Revelação mínima para um uso restrito:** Deve-se obter a menor quantidade possível de informações do usuário (relacionada à identidade digital). Sistemas computacionais estão sujeitos a falhas e a ataques, o que pode revelar informações confidenciais de usuários a entidades não autorizadas. Assim deve-se manter a menor quantidade

possível de informação e removê-la do sistema assim que não for mais necessária;

- Justificação das partes: Somente as entidades envolvidas na relação devem receber informações relativas às identidades dos outros envolvidos;
- Identidades direcionadas: O sistema deve prover identificadores com as mesmas propriedades usados por entidades públicas e identificadores unidirecionais, usados por entidades privadas. Fica então a escolha do usuário a escolha de que identificador usar para cada provedor de serviços que for interagir, garantindo seu anonimato;
- Pluralismo de operadores e tecnologias: Um sistema de gerenciamento de identidades deve operar não importando a tecnologia usada em um provedor de identidades. Assim é necessário um identificador e um protocolo comum a todos para o transporte de identidades;
- Integração com o usuário: O usuário tem que ser tratado como parte integrante do sistema, pois a interação humana com os sistemas é o elo mais fraco e a segurança nessa comunicação é algo essencial. Os usuários devem se integrar ao sistema de modo que seja possível a pronta identificação de um ataque;
- Experiência consistente através de contextos: O usuário deve ter uma experiência simples e consistente de uso dos sistemas, mesmo que internamente estejam sendo utilizadas diferentes tecnologias e operadores.

3.7. Federação

Chadwick *et al.* [7] define uma federação como sendo um grupo de organizações (universidade, instituições, provedores de conteúdo, entre outros) que compartilham um conjunto de políticas e regras tais como:

- Mecanismos de segurança utilizados entre os servidores que interagem realizando a troca de atributos;
- Definição de atributos;
- Localização dos provedores de identidade;
- Tratamento das informações pessoais que são confidenciais;
- Escolha das organizações que podem participar, estabelecendo-se desta forma uma confiança com o objetivo de se atingir uma autenticação e autorização entre os vários domínios existentes.

Chadwick *et al.* [7] ainda afirma que um dos requisitos para a utilização de serviços federados é a existência de um componente de fornecimento de identidade numa instituição. Este componente é responsável pela autenticação dos usuários na respectiva instituição de origem e pelo envio de informação associada ao usuário com o objetivo de permitir o acesso aos serviços federados.

Uma federação pode ser criada em uma variedade de formatos e modelos confiáveis que provem um grupo de serviços para membros da federação. É necessário fornecer registro para as aplicações processadas e distribuir informações dos sócios da federação para os provedores de identidade e provedores de serviço.

Como exemplo de federação, alguns mencionados em *Federação café et al.* [22], pode-se citar:

- *InQueue* era uma federação operada pela Internet2 e foi encerrada em junho de 2007. A *InQueue* foi destinada a organizações que iniciavam trabalhos com o sistema Shibboleth e confiavam em um modelo federado;
- *Federation Éducation-Recherche* – França: Foi criada em 2009. Seu objetivo é incluir 203 universidades francesas e instituições de pesquisa com provedores de identidade Shibboleth juntamente com uma série de aplicativos e serviços Shibboleth protegidos;
- *InCommon* – Estados Unidos: Operada pela Internet2. O serviço de descoberta é uma interface de descoberta de provedores de identidade centralizados para todos os participantes InCommon;
- *SDSS* – Reino Unido: Criado em 2007. Utiliza tecnologia Shibboleth com foco na autorização interinstitucional. Seu foco é para o fornecimento em longo prazo de uma infraestrutura de qualidade de serviço para o Reino Unido;
- *SWITCHaai* – Suíça: Criada em 2004. Foi construída pela comunidade SWITCH para acesso a diversos sistemas *e-learning* e aplicações web para todos os universitários da Suíça e utiliza o sistema Shibboleth;
- *CAFe* – Brasil: Desenvolvida pelo grupo de trabalho RNP (Rede Nacional de Ensino e Pesquisa). É uma federação de identidade que reúne instituições brasileiras e utiliza Shibboleth. Seu funcionamento iniciou em 2007.

3.8. Ferramenta: Shibboleth

3.8.1. Definição

Segundo Shibboleth *et al.* [11], Shibboleth é um pacote de software baseado em padrões *open source* para web com autenticação única. O Shibboleth é um projeto da Internet2, uma união de duzentas universidades americanas e europeias em conjunto com indústrias e governo para desenvolver e distribuir aplicações e tecnologias avançadas de rede, acelerando a criação de tecnologias para a Internet.

A infraestrutura de autenticação e autorização é baseada em SAML (*Security Assertion Markup Language*), um framework que permite a criação e a troca de informações de segurança entre parceiros online e que utiliza o conceito de identidades federadas. Com SAML é possível criar uma estrutura segura que simplifica o gerenciamento de identidades e fornece ao usuário a autenticação única entre organizações diferentes pertencentes a uma mesma federação e que compartilham suas informações de identidade.

Através de um provedor de identidades Shibboleth, o usuário de um dado domínio, que pertença a uma federação, pode obter acesso a todos os outros provedores de serviços ligados a essa federação por meio de uma única autenticação, sem se preocupar em se identificar novamente com seus dados cadastrais (Aplicação do conceito de *Single Sign-On*).

No Brasil, a Comunidade Acadêmica Federada (CAFe), iniciativa da Rede Nacional de Ensino e Pesquisa (RNP), utiliza esse sistema para fazer o controle de identificação de seus usuários. Uma série de aplicações são compatíveis com

a arquitetura Shibboleth, como Google Apps, Moodle, Joomla e Media Wiki – Shibboleth *et al.* [26].

Os métodos de autenticação de usuário fornecem à aplicação um identificador (por exemplo, um e-mail) para realizar a autenticação. Esta simples forma de acesso não é suficiente para os sistemas modernos. Aplicações precisam de informações adicionais sobre os usuários para realizarem corretas decisões de autorização. Desse modo, o sistema Shibboleth fornece atributos dos usuários para as aplicações com a flexibilidade, segurança e a privacidade requerida em cenários federados.

3.8.2. Provedor de Identidades Shibboleth

O provedor de identidades é o local onde os usuários realizarão a autenticação e autorização via web. Portanto esse provedor mantém credenciais e atributos e faz o controle desses atributos, liberando-os somente para as instituições confiáveis.

Existem quatro componentes principais que estão representados na

Figura 8:

- *Handle Service (HS)*: É interoperável e permite que a instituição escolha o mecanismo de autenticação desejado. Faz a autenticação dos usuários em conjunto com o mecanismo de autenticação e cria o *handle token* para o usuário. O *handle token* é um transportador de credenciais que contém um ID ou *handle*;
- *Attribute Authority (AA)*: Quando um usuário requisita acesso a um recurso, o provedor de serviços mostra o *handle token* para o AA e

requisita atributos relativos ao usuário. O AA aplica políticas de privacidade sobre a liberação desses atributos e permite que o usuário especifique quais provedores de serviço podem acessá-los;

- *Serviço de Diretório*: Será fornecido pela instituição, sendo o local onde ficarão armazenados os dados dos usuários, ou seja, os seus atributos;
- *Mecanismos de Autenticação*: Não é fornecido pelo sistema Shibboleth. Deverá ser obtido por outras partes, tais como, *WebISO (Web Initial Sign-on)*, *PubCookie* ou *CAS (Central Authentication Service)*. Essas ferramentas permitem que o usuário faça autenticação utilizando um serviço central que possibilita a utilização de apenas um usuário e senha.

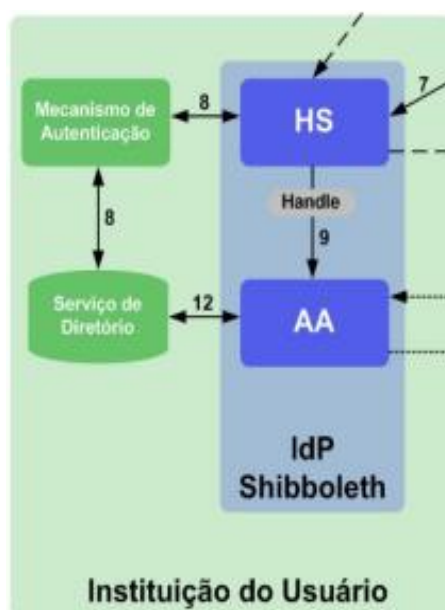


Figura 8: Componentes do Provedor de Identidades

Fonte: SBSEG *et al.* [41]

O provedor de identidades segue uma sequência de passos para que sua tarefa seja bem sucedida dentro da arquitetura Shibboleth:

1. O primeiro contato do usuário é o redirecionamento para o *Handle Service*, que consulta o mecanismo de autenticação para determinar se o usuário está autenticado (pode variar desde certificados digitais até páginas de login com senha);
2. Caso não esteja autenticado, o navegador do usuário será questionado sobre a autenticação;
3. A autenticação será feita;
4. Após a autenticação, será enviado para a *URL (Universal Resource Locator)* do provedor de serviço, um *handle* contendo a afirmação de autenticação;
5. Em seguida, o provedor de serviço envia uma requisição contendo o *handle* do usuário para o *Attribute Authority*;
6. O *AA* consulta o *ARP (Attribute Release Policies)* para a entrada de diretório correspondente ao *handle*;
7. Conforme regras das políticas consultadas, o provedor de identidades libera os atributos para o provedor de serviço.

O *Attribute Authority* mantém um grupo de políticas chamado *Attribute Release Policies (ARP)*, que administram a liberação de atributos do usuário para os provedores de serviço. Quando um usuário tenta acessar um recurso protegido pelo Shibboleth, o provedor de serviço deste recurso pergunta ao provedor de identidade sobre todos os atributos do usuário. O *AA* procura os atributos associados com o usuário, passa pelas regras do *ARP* e depois envia ao provedor de serviço somente os atributos permitidos nessa política.

Um *ARP* não pode mudar as informações dos atributos. Os *ARPs* são administrativamente mantidos e aplicados a todos os usuários para qual o provedor de identidade é autoritário. Todos os *ARPs* são especificados usando a mesma sintaxe e semântica. Cada *ARP* é formado de uma ou mais regras que especificam quais atributos e valores podem ser liberados para o provedor de serviço. A atribuição para os vários provedores de serviço é flexível e inclui algumas particularidades, como: uma regra deve afetar todos os provedores de serviço; os nomes exatos dos provedores de serviço para qual uma regra é aplicada; expressão regular no nome do provedor de serviço que deverá ser comparado para determinar se uma regra é aplicada; e aplicações individuais que podem atingir provedores de serviços.

3.8.3. Where Are You From

Where Are You From (WAYF) e *Discovery Service* (DS) representam o mesmo componente com nomes diferentes. É um componente da arquitetura Shibboleth responsável por permitir um usuário associar-se com uma instituição especificada. Ao solicitar um recurso, caso o usuário não esteja autenticado, o componente WAYF apresenta ao usuário uma lista de instituições que fazem parte da federação e o usuário deve selecionar em qual repositório de identidades ele está cadastrado. Após escolher a instituição, o usuário é redirecionado para o componente e iniciará o processo de autenticação. O serviço WAYF pode ser distribuído como parte de um provedor de serviço ou como código de terceira parte operado por uma federação.

Podemos citar como exemplo a Federação Chimarrão que é mantida para testes das instituições que desejam participar da federação CAFé. A **Figura 9**

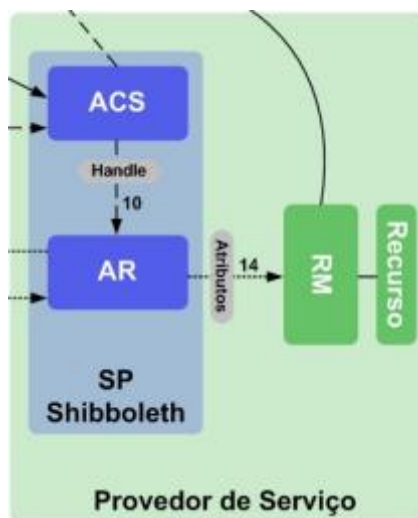


Figura 10: Componentes do Provedor de Serviço

Fonte: SBSEG *et al.* [41]

- *Assertion Consumer Service (ACS)*: É responsável por receber mensagens com o objetivo de estabelecer um contexto seguro. Pode ser definido como um recurso HTTP que processa mensagens SAML e retorna um *cookie* para o navegador, representando a informação extraída da mensagem;
- *Attribute Requester (AR)*: É o responsável por obter e repassar os atributos do usuário para o *Resource Manager (RM)*; e
- *Resource Manager (RM)*: É um componente que intercepta as requisições aos recursos e toma as decisões de controle de acesso baseadas nos atributos dos usuários.

Do ponto de vista do provedor de serviço, um navegador irá alcançar o *Resource Manager* com a requisição ao recurso protegido por Shibboleth. O *RM*

permite então que o *Assertion Consumer Service (ACS)* use o *Where Are You From (WAYF)* para adquirir o nome do provedor de identidade do usuário. Após autenticar o usuário, o *Handle Service* responde com uma afirmação de autenticação SAML *handle*, que o *ACS* entrega para o *Attribute Requester*. O *AR* usa este *handle* e o endereço fornecido do provedor de identidades para requisitar todos os atributos que são permitidos obter do usuário. Após receber os devidos atributos, o *AR* prepara algumas validações e análises baseadas nas *Attribute Acceptance Policies (AAPs)*. Estes atributos são entregues ao *RM*, que é responsável por usá-los para decidir se garante acesso ao recurso. As *AAPs* definem regras dos atributos recebidos do provedor de identidades. São utilizadas pela aplicação para decisões de controle de acesso e para fornecimento de filtros, determina quem pode acessar determinada informação.

3.8.5. Funcionamento

Na **Figura 11** são apresentados os passos de troca de mensagens para o processo de autenticação entre os componentes, provedor de identidades, provedor de serviços e agente (usuário). Cada passo é uma atividade realizada pelos componentes do sistema. Os componentes são distribuídos e trabalham para prover acesso seguro a recursos.

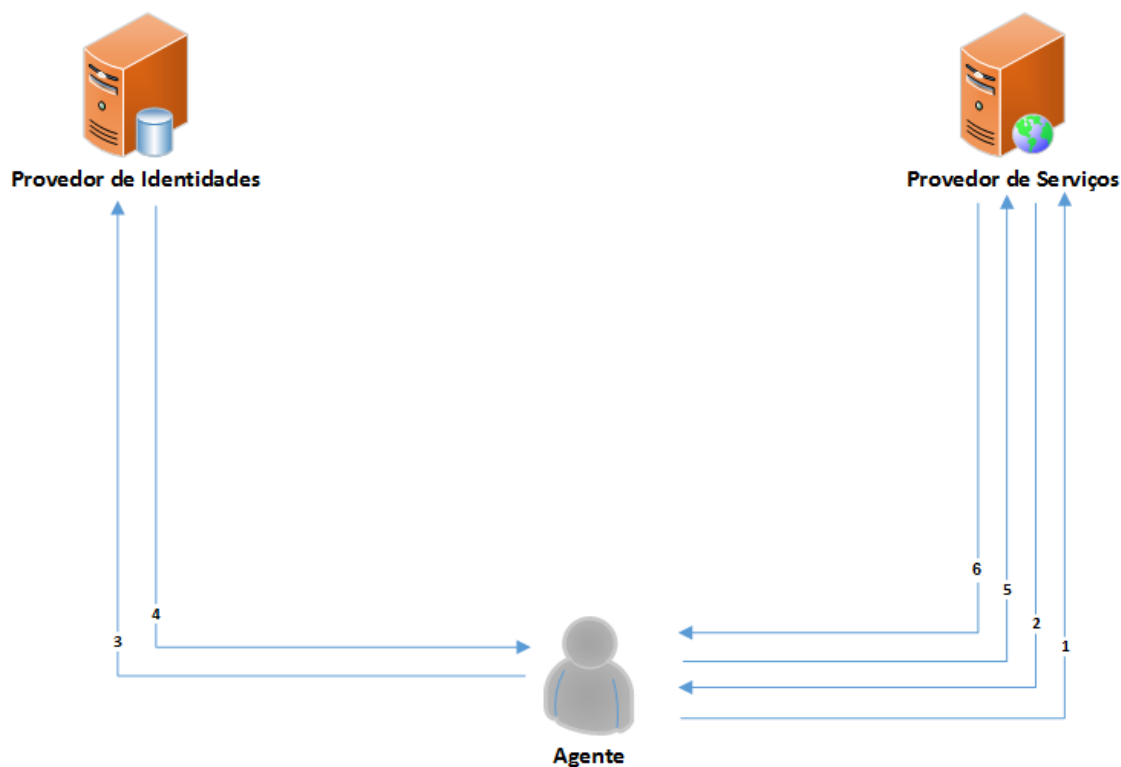


Figura 11: Passos do funcionamento do Shibboleth

Segundo o Shibboleth *et al.* [43], o seu funcionamento pode ser descrito na seguinte sequência de passos:

1. **O usuário tenta acessar um recurso protegido;**
2. **O provedor de serviços checa se há um contexto de segurança:** O provedor de serviços intercepta a requisição e verifica se ela possui um contexto de segurança. Se não possui tal contexto, então o provedor de serviços pede ao provedor de identidades que autentique o usuário, normalmente perguntando ao usuário de onde ele veio através do WAYF. Então ele redireciona o usuário para o provedor de identidades;
3. **Autenticação do usuário no provedor de identidades:** Se o usuário não estiver autenticado, o provedor de identidades questiona o usuário

pelos dados de autenticação. Caso o processo de autenticação finalize com sucesso, o provedor de identidades emite dados para autenticar o usuário no provedor de serviço;

4. **O provedor de identidades emite uma resposta ao provedor de serviço através do agente:** o provedor de identidades reúne um conjunto de atributos e os transforma, se necessário. A informação do usuário é empacotada em uma forma adequada para resposta, em uma asserção SAML (um documento XML), criptografada com a chave do provedor de serviço por razões de segurança e privacidade. O provedor de identidades responde com uma página HTML que tem formulários e um campo escondido que é asserção que foi criada;
5. **O agente entrega a resposta do provedor de identidades para o Assertion Consumer Service no provedor de serviço:** Na página que o provedor de identidades emitiu há uma instrução que solicita ao agente a realização de uma requisição para entregar os dados no ACS, então o navegador tem que enviar o formulário com a asserção que foi recebida para o ACS no provedor de serviços no carregamento da página;
6. **Recurso protegido é entregue ao usuário.**

O WAYF pode criar um *cookie* no navegador do usuário para que não seja necessária a apresentação da página de seleção da organização. E caso a organização do usuário utiliza serviço de autenticação única e o usuário já estiver com uma sessão de autenticação criada, a interface *web* de autenticação não precisa ser apresentada; e em muitos casos o usuário pode ter acesso ao recurso sem visualizar qualquer página intermediária.

3.8.6. Integração da Aplicação com Shibboleth SP

Wiki Shibboleth *et al.* [34] apresenta duas maneiras diferentes de integrar o Shibboleth SP com uma aplicação. Algumas formas podem ser mais elegantes, outras não. Algumas são simples e rápidas e outras são complicadas e tomarão muito tempo.

Muitas aplicações assumem controle completo sobre o processo de autenticação incluindo todas as interações com usuário e coleta de credenciais. O login único é a forma de tomar controle da aplicação e ensiná-la a confiar na autenticação do ambiente. No entanto, às vezes não há como ter esse grau de controle sobre a aplicação. Desse modo é necessário requisitar os atributos e utilizá-los na aplicação.

Ao contrário dos sistemas de autenticação que expõem uma API para controlar os processos de autenticação, o SP opera dentro do servidor web, isolado da aplicação colocando o manipulador da autenticação e informações de atributos em variáveis de ambiente da aplicação ou em *headers* em servidores web.

Uma aplicação que foi habilitada para confiar em uma informação HTTP, como "REMOTE_USER", está preparada para o uso do Shibboleth.

Para habilitar uma aplicação, é necessário executar dois passos:

1. Iniciar uma sessão Shibboleth: Shibboleth pode estabelecer automaticamente uma sessão sempre que uma determinada URL for acessada. Isso quer dizer que qualquer usuário que acessa esse recurso deve ser capaz de se autenticar em um IdP que é confiável

pelo SP. Os aplicativos também podem solicitar que uma sessão possa ser criada sob demanda, redirecionando o usuário para uma URL local vinculada;

2. Entregar variáveis de ambiente/*header* de autenticação e autorização em alguns pontos da aplicação: É o primeiro passo para a integração de dados. É um princípio do projeto Shibboleth SP que aplicações web nunca deveriam confiar ou exigir mecanismos de autenticação ou autorização personalizada pelo desenvolvedor. Assim o SP não tem nenhuma API de controle de autenticação. O provedor de serviços intercepta pedidos e define as variáveis de ambiente ou de *header* antes de passar o controle à aplicação. Se as aplicações são desenvolvidas nesse padrão, então mecanismos de atributo, autenticação e autorização são passados com pouca ou nenhuma modificação na aplicação.

O ambiente de integração Shibboleth, incluindo IdP, SP e aplicação, deve possuir o funcionamento da **Figura 12** quando o recurso protegido é requisitado.

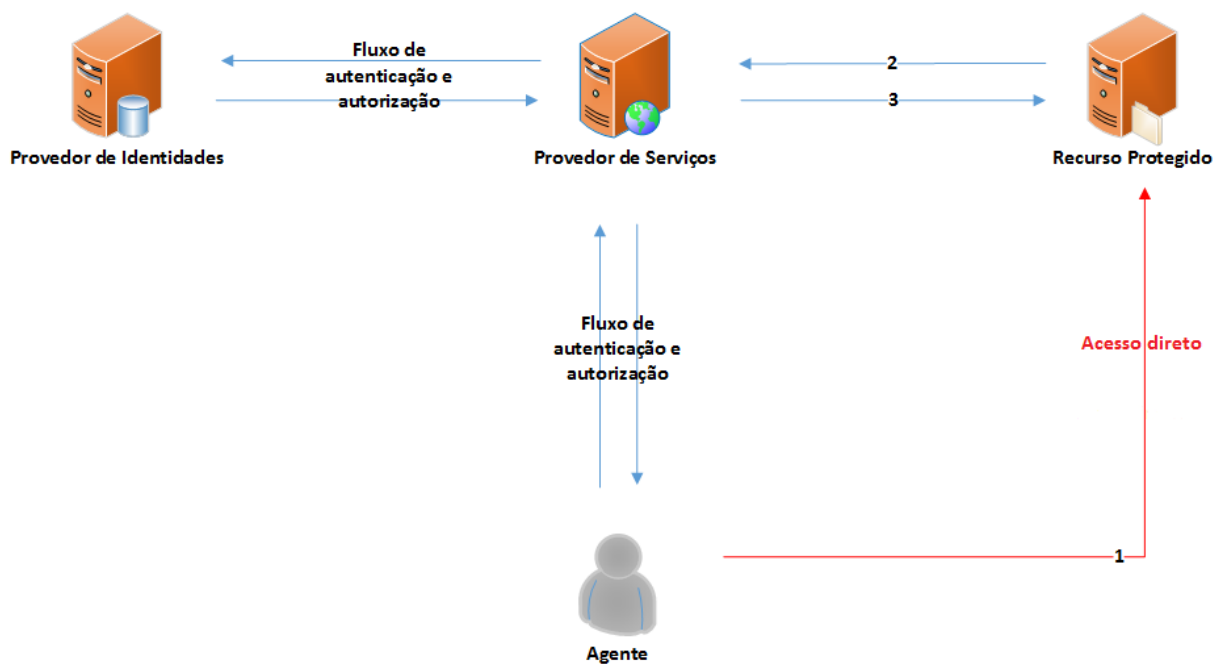


Figura 12: Acesso direto ao recurso

O usuário tenta acessar o recurso ou aplicação diretamente, contudo será direcionado ao provedor de serviço para que realize a autenticação e autorização de acesso.

1. O usuário acessa o recurso diretamente;
2. O recurso protegido direciona o usuário para o provedor de serviço para realizar a autenticação e autorização;
3. Autenticado e autorizado com base na política de atributos, o recurso é liberado para acesso.

O cenário da **Figura 12** inclui os principais componentes da arquitetura Shibboleth, que são o provedor de identidades, provedor de serviços e o recurso. Na seção **4.1. Descrição da proposta** será apresentada uma nova proposta na

qual será removido um dos componentes para melhor adaptar-se a situações características.

3.9. Tecnologias Relacionadas

3.9.1. SAML

O SAML é um padrão da OASIS que define formas de se usar o XML para representação de informações de autenticação e de atributos, definindo um protocolo para requisição e recebimento de credenciais de uma autoridade SAML – OASIS *et al.* [15].

Com SAML, um cliente faz uma requisição sobre um sujeito a uma autoridade SAML e a autoridade retorna asserções sobre a identidade do sujeito dentro de um domínio de segurança particular.

O padrão define a sintaxe e semântica não só das asserções XML, mas também dos protocolos de mensagens usados para trocar estas informações entre sistemas, dos mapeamentos para esta troca através de padrões de comunicação comuns e de perfis comuns que definem os casos de uso possíveis. A relação entre estes conceitos é ilustrada pela **Figura 13**.

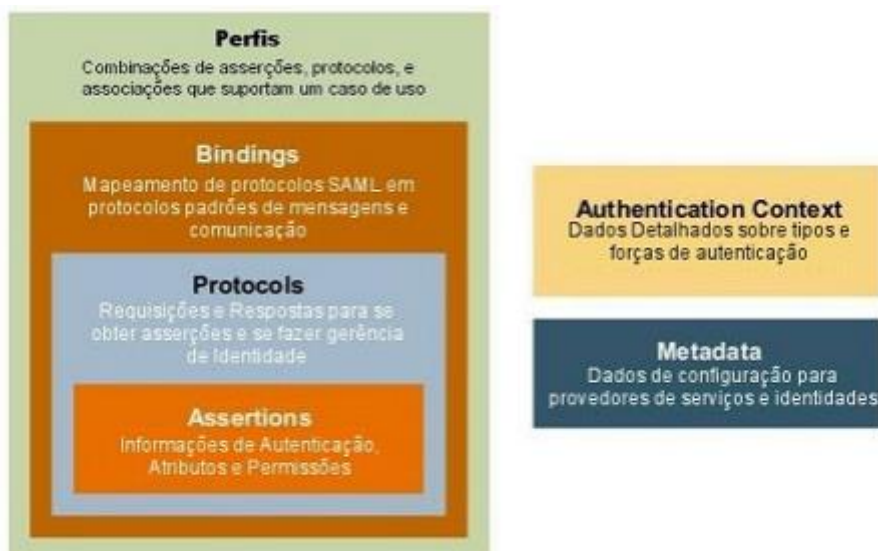


Figura 13: Relação entre conceitos do SAML

Fonte: OASIS *et al.* [44]

RND *et al.* [46] explica cada um dos componentes da **Figura 13**.

As asserções carregam instruções sobre um indivíduo que uma parte assertiva declara ser verdadeiro, em que a estrutura válida e o conteúdo de uma asserção são definidos pelo esquema XML de asserção SAML. As asserções normalmente são criadas por uma parte assertiva (também conhecida como provedor de identidade ou IdP) com base em uma solicitação de algum tipo de uma parte dependente (também conhecida como provedor de serviço ou SP), e contêm instruções que provedores de serviço usam para tomar decisões de controle de acesso. Três tipos de instruções são fornecidos pela SAML:

- Instruções de autenticação;
- Instruções de atributo;
- Instruções de decisão de autorização.

As instruções de autenticação declaram ao provedor de serviço que o principal certamente se autenticou no provedor de identidade em um dado momento, usando um método bem definido de autenticação. Outras informações sobre o método de autenticação usado pelo principal (chamado contexto de autenticação) podem ser incluídas em uma instrução de autenticação.

Uma instrução de atributo declara que um assunto está associado a certos atributos. Um atributo é simplesmente um par nome-valor. Partes dependentes usam atributos para tomar decisões de controle de acesso de baixa granularidade.

Uma instrução de decisão de autorização declara que um sujeito está autorizado a realizar a ação “A” no recurso “R” dada a evidência “E”.

Os protocolos descrevem como as asserções são empacotadas na solicitação SAML e elementos de resposta. Correspondendo aos três tipos de instrução, há três tipos de consulta SAML:

- Consulta de autenticação;
- Consulta de atributo;
- Consulta de decisão de autorização.

As ligações são um mapeamento de uma mensagem de protocolo SAML em formatos de sistema de mensagens padrão e/ou protocolos de comunicação. Por exemplo, a ligação SOAP SAML especifica como uma mensagem SAML é encapsulada em um envelope SOAP, sendo ele mesmo ligado a uma mensagem HTTP.

Os perfis descrevem em detalhes como as asserções, protocolos e ligações SAML se combinam para suportar um caso de uso definido.

As especificações SAML definem cinco componentes ilustrados pela **Figura**

14:

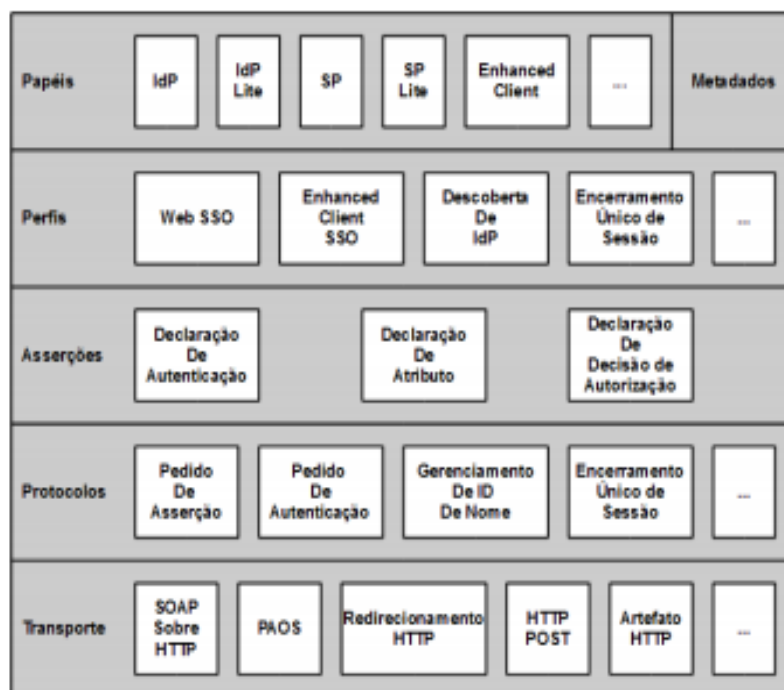


Figura 14: Arquitetura SAML

Fonte: OASIS *et al.* [45]

- Papéis: Cada entidade pode desempenhar vários papéis na infraestrutura SAML. Também são apresentados os metadados que descrevem essas entidades;
- Perfis: Agregam protocolos e asserções em fluxos de dados específicos para prover funcionalidades como gerenciamento de identidades e autenticação única;

- **Asserções:** São essenciais para o gerenciamento de identidades e de contextos de segurança. Carregam instruções sobre algo que uma parte assertiva declara como ser verdadeiro. É definido em XML (*eXtensible Markup Language*). Três tipos de instrução são fornecidas pela SAML:
 - **Asserções de autenticação:** Declara ao provedor de serviço que aconteceu a autenticação no provedor de identidade em um dado momento;
 - **Asserções de atributo:** Declara que um assunto está associado a certos atributos. Um atributo é simplesmente um par nome-valor;
 - **Asserções de decisão de autorização:** Declara que um sujeito está autorizado e poderá realizar uma ação “A” no recurso “R” dada a evidência “E”;
- **Protocolos:** São usados para requerer e transferir asserções entre as entidades. Descrevem como as asserções são empacotadas na solicitação SAML e elementos de resposta. Há três tipos de consulta SAML correspondendo aos três tipos de instrução:
 - Consulta de autenticação;
 - Consulta de atributo;
 - Consulta de decisão de autorização;
- **Transporte:** Esses protocolos podem ser mapeados em diferentes mecanismos de transporte.

3.9.2. CAS

O CAS (*Central Authentication Server*) é um sistema de autenticação originalmente desenvolvido pela Universidade de Yale para fornecer uma maneira confiável para uma aplicação autenticar um usuário. Tornou-se um projeto Jasig em dezembro de 2004 – CAS *et al.* [28]. Jasig é um consórcio global de instituições de ensino e afiliadas comerciais que patrocinam projetos de software de código aberto que beneficiem o ensino superior – HARRIS *et al.* [31].

Como exemplo de CAS, podemos citar alguns sistemas da Universidade Federal de Santa Catarina que utilizam esse tipo de autenticação: os usuários da universidade podem acessar webmail, wireless, VPN, idUFSC e outros serviços com apenas um identificador de usuário e uma senha comum a todos.

Vantagens para se utilizar o CAS como provedor de *single sign-on* – CAS *et al.* [28]:

- Possui um protocolo aberto e bem documentado;
- Um componente de servidor *open-source* Java;
- Bibliotecas de clientes para Java, .Net, PHP, Perl, Apache, uPortal, entre outros;
- Fácil integração com aplicações uPortal, Sakai, BlueSocket, TikiWiki, Mula, Liferay, Moodle e outros;
- Documentação da comunidade e apoio à implementação;
- Comunidades de desenvolvedores ativos.

O protocolo CAS envolve no mínimo três partes: um navegador web cliente, a aplicação que faz a requisição de autenticação e o servidor CAS. Pode

também envolver um serviço de *back-end* como um servidor de banco de dados
– CAS *et al.* [30]. Pode-se analisar na **Figura 15** o fluxo de autenticação do CAS:

1. Um usuário acessa um recurso web protegido;
2. O navegador é redirecionado ao servidor CAS;
3. O servidor CAS envia o formulário de autenticação ao usuário;
4. O usuário responde o formulário com suas credencias;
5. O servidor CAS cria um *cookie* de sessão e redireciona o navegador para o servidor web, com um *ticket* de uso único;
6. A aplicação valida o *ticket* contatando o servidor de eventos que devolve o identificador do indivíduo;
7. O usuário acessa o recurso solicitado.

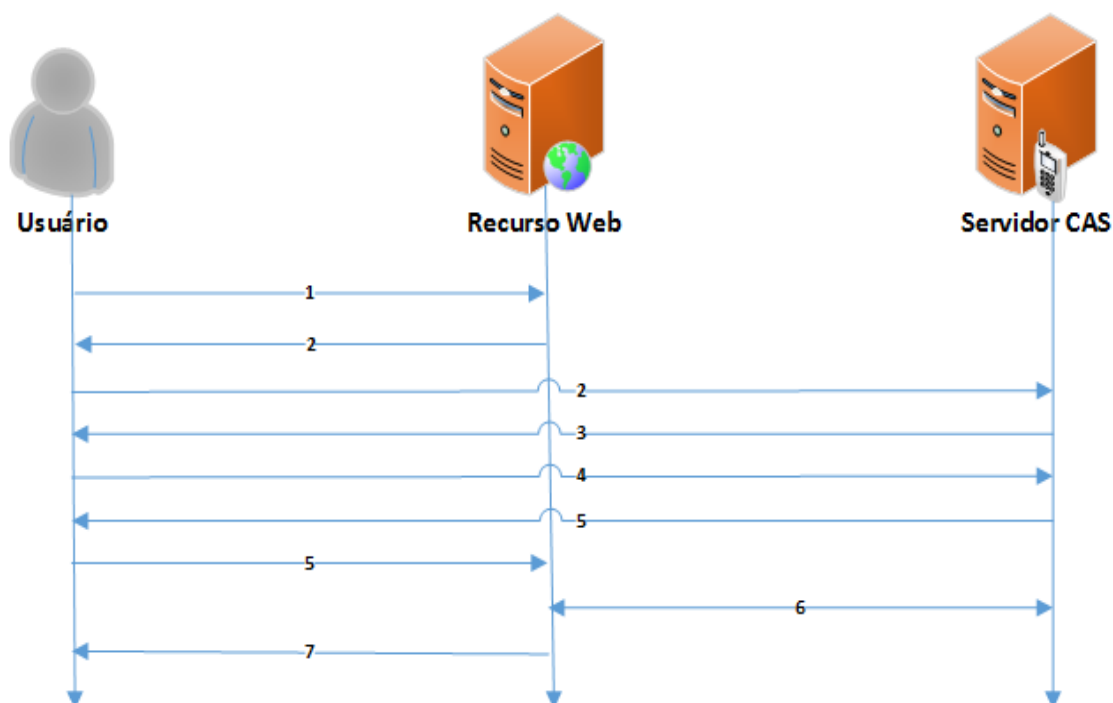


Figura 15: Fluxo de autenticação no CAS.

O protocolo CAS inclui uma série de recursos de segurança que são projetados para limitar o dano causado por uma possível falha de segurança em um aplicativo cliente. Uma das opções para que isso não aconteça é implementar o protocolo *single sign out*. Essa abordagem armazena um *ticket* URL de cada serviço do usuário durante a sessão de *single sign-on*. Quando o usuário solicita a saída, o servidor CAS emite uma HTTP POST para todos os serviços autenticados. Clientes do CAS analisarão a requisição e terminarão a sessão local com usuários, assim garantindo que ao efetuar o logout, nenhuma sessão em uma das aplicações em que o usuário se autenticou fique ativa, dando margem para ataques por roubo de identificadores de sessões.

3.9.3. Spring Security

De acordo com Spring Security *et al.* [35], Spring Security é um *framework* que fornece autenticação e autorização para aplicações Java. Em todos os projetos do Spring, o diferencial está na facilidade de utilização para estender e atender às exigências de cada desenvolvedor.

As características dessa ferramenta são:

- Suporte abrangente e extensível para autenticação e autorização;
- Proteção contra ataques como fixação de sessão, *clickjacking*¹, etc;
- Integração de API Servlet;
- Integração opcional com o Spring Web MVC;
- Entre outras opções.

¹ O *clickjacking* ("furto de click") é uma técnica informática fraudulenta que rouba os cliques do mouse e repassa ao atacante do site.

A maneira recomendada para começar a usar o Spring Security em um projeto é através de um sistema de gerenciamento de dependências. O trecho da **Figura 16** pode ser copiado e colado em uma construção.

```
<dependencies>
  <dependency>
    <groupId> org.springframework.security </groupId>
    <artifactId> spring-security-web </artifactId>
    <version> 3.2.5.RELEASE </version>
  </dependency>
</dependencies>
```

Figura 16: Dependência do framework Spring Security

Fonte: Spring Security Extension *et al.* [33]

O *framework* centraliza em um único documento XML todas as configurações, dispensando configurações do container e tornando a aplicação mais abstrata e segura.

O Spring Security utiliza-se de um filtro HTTP para interceptar todas as requisições HTTP recebidas e conferir suas permissões de acesso. As configurações de autenticação e autorização são feitas em um arquivo XML contendo o contexto da aplicação.

De uma maneira geral, o fluxo de autenticação e autorização segue os passos a seguir:

1. Usuário submete suas credenciais através do seu navegador;
2. Mecanismo de autenticação coleta os detalhes:
 1. Um objeto de requisição de autenticação é construído;

2. O objeto é enviado a um manipulador de autenticações;
 3. O manipulador de autenticações passa a requisição através de uma cadeia de provedores de autenticação;
 4. O(s) provedor(es) de autenticação solicitarão os detalhes do serviço para prover um novo objeto com os detalhes;
 5. O novo objeto construído será usado para construir um objeto completo de autenticação.
3. Se o mecanismo de autenticação receber de volta um objeto completo de autenticação, será uma requisição válida. Nesse caso, a autenticação será colocada em um contexto de segurança. Por outro lado, se for rejeitada a requisição, o mecanismo de autenticação solicitará ao usuário que tente novamente;
 4. O interceptador de segurança autoriza ou gera exceções Java;
 5. As exceções são traduzidas para erro de HTTP, como por exemplo ERROR Code 403.

O Spring Security também inclui um pacote que manipula autenticação e autorização SAML. Como descrito em *Spring Security Extension et al.* [33], a especificação SAML 2.0 combinada com o *framework* Spring Security permite a combinação perfeita de mecanismos de autenticação e autorização em uma única aplicação. Todos os produtos que suportam SAML 2.0 podem ser usados em conjunto com a extensão do Spring Security SAML, inclusive o *Shibboleth*.

Algumas das características da extensão do Spring Security SAML são:

- Dar suporte a vários perfis SAML (Single Sign-on, Single sign-on-titular-de-chave, logout único, proxy);

- IdP e SP inicializados em Single Sign-on;
- Seleção do IdP e o perfil de descoberta do provedor de identidades;
- Interoperabilidade de metadados e gerenciamento de confiança;
- Geração automática de metadados do SP;
- Carga de metadados de arquivos, URLs e URLs baseadas em arquivo;
- Processamento e recarga automática de metadados com muitos provedores de identidade;
- Suporte para contextos de autenticação com logs de evento;
- Suporte para HTTP-POST, HTTP-Redirect, SOAP, PAOS e construção de artefatos;
- Personalização dos metadados de ambos SP e IdP.

A maneira mais indicada de usar a extensão do Spring Security SAML é adicionando no gerenciamento de dependências do projeto o trecho de código da **Figura 17**.

```
<dependencies>
  <dependency>
    <groupId> org.springframework.security.extensions </groupId>
    <artifactId> spring-security-saml </artifactId>
    <version> 1.0.0.RELEASE </version>
  </dependency>
</dependencies>
<repositories>
  <repository>
    <id> spring-milestones </id>
    <name> Spring Milestones </name>
```

```
<url> http://repo.spring.io/milestone </url>

<snapshots>
  <enabled> false </enabled>
</snapshots>
</repository>
</repositories>
```

Figura 17: Código a ser inserido para uso da extensão do Spring Security

Fonte: Spring Security Extension *et al.* [33]

A extensão também pode ser usada em aplicações que não estão protegidas. Tanto aplicações individuais quanto aplicações *multi-tenancy*² podem ser adaptadas com Spring Security.

A extensão pode ser incorporada dentro da aplicação fazendo-a atuar como um provedor de serviços ou pode atuar separadamente funcionando como um *proxy* transmitindo informações para a aplicação.

A extensão habilita aplicações atuarem como provedor de serviços em federações baseada em perfis de single sign-on e logout único do protocolo SAML 2.0. A aplicação que utiliza a extensão do Spring Security SAML para se comunicar com o provedor de identidades Shibboleth não necessita do servidor web Apache e o plug-in do Shibboleth, já que a extensão faz todo o trabalho.

² É a arquitetura na qual uma única instância de uma aplicação serve vários clientes. Cada cliente é chamado de inquilino.

4. Proposta de um modelo de biblioteca cliente para o Shibboleth: Facilitando a adaptação das aplicações

4.1. Descrição da proposta

No contexto atual, o provedor de serviços armazena os recursos a serem requisitados pelo usuário e a autenticação criando um contexto seguro. Na nova proposta, a aplicação descarta o uso do *plugin* do Apache chamado SP do Shibboleth e passa a englobar suas funções.

O provedor de serviços gerencia a proteção do recurso requisitado trocando asserções de segurança com o provedor de identidade para a avaliação de acesso. Para que o provedor de serviços possa ser descartado, é necessário que a aplicação manipule as mesmas asserções permitindo a atribuição das funções do SP à aplicação por meio de bibliotecas pré-existentes.

Para melhor visualização, a **Figura 18** ilustra o novo contexto. Observa-se que não há mais a necessidade de um módulo do SP Shibboleth, já que a aplicação gerencia a segurança.

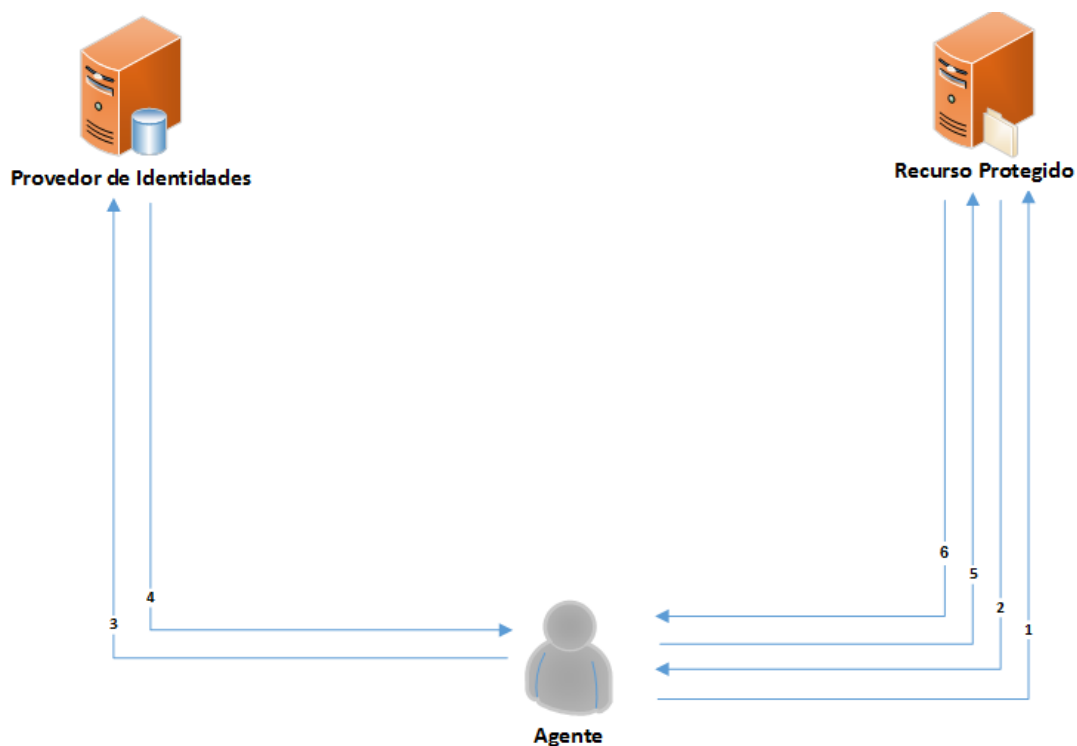


Figura 18: Aplicação atuando como um provedor de serviços

O seu funcionamento pode ser descrito na seguinte sequência de passos:

1. **O usuário tenta acessar um recurso protegido;**
2. **A aplicação checa se há um contexto de segurança:** A aplicação intercepta a requisição e verifica se ela possui um contexto de segurança. Se não possui, a aplicação pede ao provedor de identidades que autentique o usuário, normalmente perguntando ao usuário de onde ele veio através do WAYF. Então o usuário é redirecionado para o provedor de identidades;
3. **Autenticação do usuário no provedor de identidades:** Se o usuário não estiver autenticado, o provedor de identidades questiona o usuário

pelos dados de autenticação. Caso o processo de autenticação finalize com sucesso, o provedor de identidades emite dados para autenticar o usuário na aplicação;

4. **O provedor de identidades emite uma resposta à aplicação através do agente:** o provedor de identidades reúne um conjunto de atributos e os transforma, se necessário. A informação do usuário é empacotada em uma forma adequada para resposta, em uma asserção SAML (um documento XML), criptografada com a chave da aplicação por razões de segurança e privacidade. O provedor de identidades responde com uma página HTML que tem formulários e um campo escondido que é asserção que foi criada;
5. **O agente entrega a resposta do provedor de identidades para a aplicação:** Na página que o provedor de identidades emitiu há uma instrução que solicita ao agente a realização de uma requisição para entregar os dados para a aplicação, então o navegador tem que enviar o formulário com a asserção que foi recebida para a aplicação no carregamento da página;
6. **Aplicação libera seu conteúdo de acesso.**

A aplicação protegida terá o comportamento desejado utilizando a extensão atual do Spring Security SAML com Java 1.8 em ambiente Web.

A segunda proposta é extrair atributos armazenado no provedor de identidades para que possam ser utilizados no contexto da aplicação.

A terceira e última proposta é implementar o *single logout* global, já que o Shibboleth não implementa. Pode-se usar a extensão do Spring Security SAML para a manipulação do SLO através de sessões no servidor da aplicação.

4.2. Configuração

Nessa seção será apresentada a configuração da aplicação para que possa englobar as funções do provedor de serviços e a configuração do ambiente onde a aplicação estará inserida.

Para a construção da aplicação web, foi utilizado JDK 1.8 em uma estrutura de gerenciamento de dependências Maven para importar as bibliotecas do *framework* do Spring Security utilizando IDE Eclipse Luna. A aplicação será executada numa máquina virtual Linux com Tomcat 7.

4.2.1. Configuração da Aplicação

A aplicação deve ser configurada para executar com os módulos Spring Security, Spring MVC, OpenSAML e extensão Spring Security SAML. O Spring Security cuida da segurança da aplicação, o Spring MVC organiza para maior entendimento e controle da programação, o OpenSAML manipula as asserções SAML e a extensão do Spring Security SAML integra a aplicação em um sistema de gerenciamento de identidades.

Começamos construindo uma aplicação Maven Web, adicionando a ela dependências dos módulos do Spring e SAML. Os seguintes passos mostram como criar uma aplicação web e a inserção das dependências:

1. No Eclipse, é necessário criar o projeto em File – New – New Maven Project e selecionar a arquitetura. Nesse caso, deseja-se uma aplicação web, então a arquitetura a ser adicionada será “maven-archetype-webapp”;

Adicionar as dependências da **Figura 19** no arquivo pom.xml da aplicação.

```
<dependencies>
  <dependency>
    <groupId>org.springframework.security.extensions</groupId>
    <artifactId>spring-security-saml2-core</artifactId>
    <version>1.0.0.RELEASE</version>
    <scope>compile</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-config</artifactId>
    <version>3.1.2.RELEASE</version>
    <scope>compile</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
    <version>3.1.2.RELEASE</version>
    <scope>compile</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-beans</artifactId>
    <version>3.1.2.RELEASE</version>
    <scope>compile</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
```



```
<artifactId>spring-context</artifactId>

<version>3.1.2.RELEASE</version>

<scope>compile</scope>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-aop</artifactId>
  <version>3.1.2.RELEASE</version>
  <scope>compile</scope>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-web</artifactId>
  <version>3.1.2.RELEASE</version>
  <scope>compile</scope>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-webmvc</artifactId>
  <version>3.1.2.RELEASE</version>
  <scope>compile</scope>
</dependency>
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>jsp-api</artifactId>
  <version>2.0</version>
  <scope>provided</scope>
</dependency>
</dependencies>
```

Figura 19: Dependência do Spring Security

É importante ressaltar que todas as versões devem ser compatíveis. Adicionando as dependências, a aplicação estará apta a utilizar os módulos.

2. O arquivo `web.xml` também deve ser alterado para conter o `DispatcherServlet` do Spring Security usado para registrar manipuladores que processam as requisições HTTP. É preciso definir o mapeamento dos filtros que delegam a chamada para uma determinada classe, como demonstra a **Figura 20**. Esses dois elementos servem para interceptar todas as requisições pelas bibliotecas do Spring Security.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">

  <display-name>Spring Security SAML</display-name>
  <description>Sample application demonstrating Spring security
SAML integration.</description>

  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>
      /WEB-INF/securityContext.xml
    </param-value>
```

```
</context-param>

<servlet>
    <servlet-name>saml</servlet-name>
    <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
    <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
    <servlet-name>saml</servlet-name>
    <url-pattern>/saml/web/*</url-pattern>
</servlet-mapping>

<filter>
    <filter-name>springSecurityFilterChain</filter-name>
    <filter-
class>org.springframework.web.filter.DelegatingFilterProxy</filter-
class>
</filter>
<filter-mapping>
    <filter-name>springSecurityFilterChain</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>

<listener>
    <listener-
class>org.springframework.web.context.ContextLoaderListener</listene
r-class>
```

```
</listener>

<welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
</welcome-file-list>

<error-page>
    <exception-type>java.lang.Exception</exception-type>
    <location>/error.jsp</location>
</error-page>
</web-app>
```

Figura 20: Exemplo de configuração do web.xml

O listener “ContextLoaderListener” é carregado quando iniciamos e paramos a interface “ApplicationContext” que é a raiz do Spring. “ContextLoaderListener” determina quais configurações serão utilizadas, procurando por uma tag <context-param> e por “contextConfigLocation”. No exemplo da **Figura 20** definimos que o caminho se encontra em “/WEB-INF/securityContext.xml”.

A interceptação das requisições acontecem através dos filtros. É importante que o filtro “springSecurityFilterChain” do Spring Security seja o primeiro filtro, caso a aplicação tenha mais de um. Isso garante a maior segurança na aplicação.

Para interceptar as requisições, é necessário incluir o filtro, como na **Figura 21**.

```

<filter>
  <filter-name>springSecurityFilterChain</filter-name>
  <filter-class>
    org.springframework.web.filter.DelegatingFilterProxy
  </filter-class>
</filter>
<filter-mapping>
  <filter-name>springSecurityFilterChain</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>

```

Figura 21: Filtro para interceptar requisições

Quando trabalha-se com Spring Security, o filtro “DelegatingFilterProxy” delegará para o filtro “FilterChainProxy”, que foi criado com o arquivo security.xml. O “FilterChainProxy” permite ao Spring Security aplicar qualquer número de filtros *servlets* para requisições de *servlets*.

O Servlet *saml* está fazendo papel de *front controller* na aplicação Spring MVC, ou seja, recebe as requisições e envia às lógicas corretas.

Por possuir sua própria configuração XML, o Spring define seu próprio XML com várias opções para configuração da aplicação dentro de “src/main/webapp/WEB-INF”. Nesse arquivo informa-se o uso de anotações do Spring MVC e também é necessário informar o local onde colocaremos os arquivos JSP. A **Figura 22** demonstra o código fonte do arquivo.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```

<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"

       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">

    <context:component-scan base-
package="org.springframework.security.saml.web"/>

    <bean
class="org.springframework.web.servlet.view.InternalResourceViewReso
lver">
        <property name="prefix" value="/WEB-INF/jsp"/>
        <property name="suffix" value=".jsp"/>
    </bean>

</beans>

```

Figura 22: Arquivo xml de configuração da aplicação

Para que a aplicação esteja em um contexto seguro, o Spring centraliza sua segurança toda em um único arquivo XML no qual definirá suas políticas através de *beans*, que pode ser definido como uma classe Java que expõe propriedades seguindo uma convenção de nomenclatura simples para os métodos *get* e *set*. Nesse arquivo informaremos todos os elementos do gerenciamento de identidades, seus métodos de autenticação, autorização, seu

círculo de confiança e outras propriedades. Para melhor entendimento, o arquivo será fragmentado com as partes de extrema importância para o resultado final.

Devemos informar pontos seguros da aplicação e pontos sem autenticação. A **Figura 23** demonstra como se faz a especificação.

```
<!-- Unsecured pages -->
<security:http security="none" pattern="/favicon.ico" />
<security:http security="none" pattern="/images/**" />
<security:http security="none" pattern="/css/**" />
<security:http security="none" pattern="/logout.jsp" />

<!-- Security for the administration UI -->
<security:http pattern="/saml/web/**" access-denied-
page="/saml/web/metadata/login">
    <security:form-login login-processing-url="/saml/web/login"
        login-page="/saml/web/metadata/login" default-
target-url="/saml/web/metadata" />
    <security:intercept-url pattern="/saml/web/metadata/login"
        access="IS_AUTHENTICATED_ANONYMOUSLY" />
    <security:intercept-url pattern="/saml/web/**"
        access="ROLE_ADMIN" />
    <security:custom-filter before="FIRST"
        ref="metadataGeneratorFilter" />
</security:http>

<!-- Secured pages with SAML as entry point -->
<security:http entry-point-ref="samlEntryPoint">
    <security:intercept-url pattern="/**"
        access="IS_AUTHENTICATED_FULLY" />
</security:http>
```

```
<security:custom-filter before="FIRST"
    ref="metadataGeneratorFilter" />
<security:custom-filter after="BASIC_AUTH_FILTER"
    ref="samlFilter" />
</security:http>
```

Figura 23: Áreas segura e não segura do arquivo securityContext.xml

Os filtros do módulo SAML precisam ser habilitados como parte das configurações do Spring Security. Eventuais erros durante o processamento de mensagens SAML são geralmente propagadas como exceções de Servlet para o *container* Java. Para capturar essas exceções, configura-se um arquivo de erro no web.xml indicando o código da **Figura 24**.

```
<error-page>
    <exception-type>javax.servlet.ServletException</exception-type>
    <location>/error.jsp</location>
</error-page>
```

Figura 24: Código de erro SAML

Os metadados SAML estão presentes em um documento XML que contém informações necessárias para interação com provedores de identidade e provedores de serviço. O documento contém o endereço dos *endpoints*, informações sobre vinculações, identidades e chaves públicas. Os metadados serão gerados pela aplicação e enviados ao provedor de identidades no qual

deseja-se habilitar o *single sign-on*. A aplicação também necessitará de acesso aos metadados do provedor de identidades.

Para a geração automática dos metadados pela aplicação, o código da **Figura 25** pode ser adicionado no arquivo de configuração do Spring Security.

```
<security:custom-filter before="FIRST" ref="metadataGeneratorFilter"/>
```

Figura 25: Código de geração automático dos metadados da aplicação

O filtro invocará a função manipuladora automaticamente como parte da URL na qual está sendo processado o Spring Security.

É necessário especificar um *entityId* único identificando a aplicação. Se não informar, a aplicação gerará automaticamente com parâmetros de requisições HTTP. A **Figura 26** mostra o código a ser inserido para a correta geração dos metadados da aplicação.

```
<!-- Filter automatically generates default SP metadata -->
<bean id="metadataGeneratorFilter"
class="org.springframework.security.saml.metadata.MetadataGeneratorFilter">
  <constructor-arg>
    <bean
class="org.springframework.security.saml.metadata.MetadataGenerator">
      <property name="extendedMetadata">
        <bean
class="org.springframework.security.saml.metadata.ExtendedMetadata">
```

```

                                <property                name="idpDiscoveryEnabled"
value="true" />
                                </bean>
                                </property>
                                <property                name="entityBaseURL"
value="https://tccmax.lrg.ufsc.br/spring-security-saml2-sample" />
                                </bean>
                                </constructor-arg>
                                </bean>

```

Figura 26: Atributos para geração dos metadados da aplicação

O atributo identificador da aplicação como SP foi especificado e também foi habilitado o módulo que tem função de localização do IdP.

Para que a aplicação saiba para onde redirecionar a resposta de autenticação, é necessário que a aplicação conheça os metadados do servidor, para isso, adiciona-se o código da **Figura 27**.

```

<!-- IDP Metadata configuration - paths to metadata of IDPs in circle of trust is
here -->
    <bean id="metadata"
class="org.springframework.security.saml.metadata.CachingMetadataManager">
        <constructor-arg>
            <list>
                <bean
class="org.opensaml.saml2.metadata.provider.HTTPMetadataProvider">
                    <constructor-arg>

```

```

        <value
type="java.lang.String">https://idp.lrg.ufsc.br/idp/profile/Metadata/SAML</value>
        </constructor-arg>
        <constructor-arg>
            <value type="int">15000</value>
        </constructor-arg>
        <property name="parserPool" ref="parserPool" />
    </bean>
</list>
</constructor-arg>
</bean>

<!-- SAML Authentication Provider responsible for validating of received
      SAML messages -->
<bean id="samlAuthenticationProvider"
      class="org.springframework.security.saml.SAMLAuthenticationProvider">
</bean>

```

Figura 27: Código para indicar os metadados do IdP

A configuração do *bean* busca os metadados na URL especificada com um *timeout* de 15 segundos.

A extensão do Spring Security oferece dois tipos de logout:

- Logout local: Destrói a sessão local e fará o logout do usuário, enquanto estiver ativa, não precisa se autenticar novamente no IdP;
- Logout global: Destrói a sessão local e a sessão do IdP. No encerramento da sessão no IdP também destruirá sessões de outros SPs que o usuário estiver logado.

Para o logout local, basta adicionar o bloco da **Figura 28**, já o logout global é necessário passar manipuladores de SAML 2.0, como na **Figura 29**.

```
<security:http>
    <security:logout      logout-url="/j_logout"      logout-success-
url="/logout.jsp"/>
</security:http>
```

Figura 28: Bloco de logout local

```
<!-- Logout handler terminating local session -->
    <bean id="logoutHandler"
        class="org.springframework.security.web.authentication.logout.S
ecurityContextLogoutHandler">
        <property name="invalidateHttpSession" value="false" />
    </bean>

    <!-- Override default logout processing filter with the one
processing SAML
        messages -->
    <bean                                id="samlLogoutFilter"
class="org.springframework.security.saml.SAMLLogoutFilter">
        <constructor-arg index="0" ref="successLogoutHandler" />
        <constructor-arg index="1" ref="logoutHandler" />
        <constructor-arg index="2" ref="logoutHandler" />
    </bean>

    <!-- Filter processing incoming logout messages -->
```

```
<!-- First argument determines URL user will be redirected to
after successful
    global logout -->
<bean id="samlLogoutProcessingFilter"
class="org.springframework.security.saml.SAMLLogoutProcessingFi
lter">
    <constructor-arg index="0" ref="successLogoutHandler" />
    <constructor-arg index="1" ref="logoutHandler" />
</bean>
```

Figura 29: Manipuladores do logout global

Após a configuração, basta utilizar a classe gerada pelo Spring que atuará como controladora.

A classe utilizará anotações para fazer o mapeamento entre URLs das requisições com métodos que devem ser executados. *@Controller* indica que a classe é uma controladora, é uma anotação do Spring MVC. Essa mesma anotação indica que seus métodos são ações. Pode-se criar qualquer método dentro dessa classe, contudo os métodos que não estão mapeados com a anotação *@RequestMapping* não atendem a requisições. Essa anotação recebe um parâmetro que indica qual será a URL utilizada para invocar o método. A anotação *@Autowired* fornece controle sobre a injeção de dependências.

Essa classe é responsável por delegar a função de criação, gerenciamento e remoção de partes do sistema de gerenciamento de identidades, como por exemplo, a criação de metadados para a aplicação.

4.2.2. Configuração do Ambiente

A aplicação necessita de uma camada segura e de confiança para trocar mensagens entre provedores de identidade e provedor de serviço, para isso é necessário estabelecer um canal SSL duplamente autenticado configurando portas de redirecionamento, protocolos e outros parâmetros. Assinaturas são tipicamente construídas usando criptografia assimétrica e infraestrutura de chaves públicas assinadas por uma autoridade de certificação confiável.

A verificação das assinaturas são executadas em duas fases. A assinatura primeiro é checada por validade comparando seu *hash* como parte da assinatura e subseqüentemente é verificada quanto a sua cadeia.

O primeiro passo para estabelecer o canal seguro, é a geração de um certificado que será utilizado para identificar o servidor onde está a aplicação e de uma *keystore* para armazenar o certificado utilizando os comandos da **Figura 30** com a ferramenta *keytool* presente na pasta *bin* do JDK.

```
keytool -keystore Tomcat.jks -genkey -keyalg rsa -alias  
tccmax.lrg.ufsc.br  
  
keytool -keystore Tomcat.jks -selfcert -alias tccmax.lrg.ufsc.br
```

Figura 30: Comandos para geração de um certificado auto assinado

O primeiro comando cria a *keystore* Tomcat.jks com um certificado tccmax.lrg.ufsc.br dentro. Para que seja auto assinado, o segundo comando é executado. Durante a criação do certificado, dados são requisitados, conforme

Figura 31, para evitar problemas, o CN precisa ter o mesmo nome do servidor, ou seja, **tccmax.lrg.ufsc.br** que está em destaque.

```
Enter keystore password: password
Re-enter new password: password
What is your first and last name?
  [Unknown]: tccmax.lrg.ufsc.br
What is the name of your organizational unit?
  [Unknown]: home
What is the name of your organization?
  [Unknown]: home
What is the name of your City or Locality?
  [Unknown]: Sao Paulo
What is the name of your State or Province?
  [Unknown]: SP
What is the two-letter country code for this unit?
  [Unknown]: BR
Is CN=Loiane Groner, OU=home, O=home, L=Sao Paulo, ST=SP, C=BR
correct?
  [no]: yes

Enter key password for
  (RETURN if same as keystore password): password
Re-enter new password: password
```

Figura 31: Informações requisitadas para geração de um certificado

Para que a aplicação confie no provedor de identidades, a chave pública deve estar presente no servidor da aplicação. Para extrair uma chave pública, pode-se baixar o *SSL Extractor utility* (disponível em GitHub *et al.* [47]) e utilizar o comando da **Figura 32**.

```
java -jar sslextractor-0.9.jar https://idp.lrg.ufsc.br/idp/ 443
```

Figura 32: Comando para extrair chave pública de uma URL

O certificado será armazenado como extensão *.cer* e poderá ser importado para a *keystore*.

É necessário alterar as portas padrões do Tomcat de HTTP para HTTPS facilitando as configurações das aplicações, pode-se começar a configurar o servidor para ouvir a porta 80 e redirecionar para a 443. Para realizar a alteração, o arquivo *server.xml* presente na pasta *conf* do diretório de instalação do Tomcat deve ser aberto e localizado a declaração da **Figura 33**.

```
<!--  
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"  
    maxThreads="150" scheme="https" secure="true"  
    clientAuth="false" sslProtocol="TLS" />  
-->
```

Figura 33: Declaração comentada no *server.xml* do Tomcat

Depois de localizar a declaração, o código deve estar sem *tags* de comentário e alterado de acordo com a **Figura 34** referenciando a *keystore* criada no início. Uma porta 80 foi adicionada também redirecionando para a porta segura 443.

```
<Connector port="443" protocol="HTTP/1.1" SSLEnabled="true"
    maxThreads="150" scheme="https" secure="true"
    clientAuth="false" sslProtocol="TLS"
    keystoreFile="/var/lib/tomcat7/webapps/tomcat.jks"
    keystorePass="changeit" />

<Connector port="80" protocol="HTTP/1.1"
    connectionTimeout="20000"
    URIEncoding="UTF-8"
    redirectPort="443" />

<Connector executor="tomcatThreadPool"
    port="8080" protocol="HTTP/1.1"
    connectionTimeout="20000"
    redirectPort="443" />
```

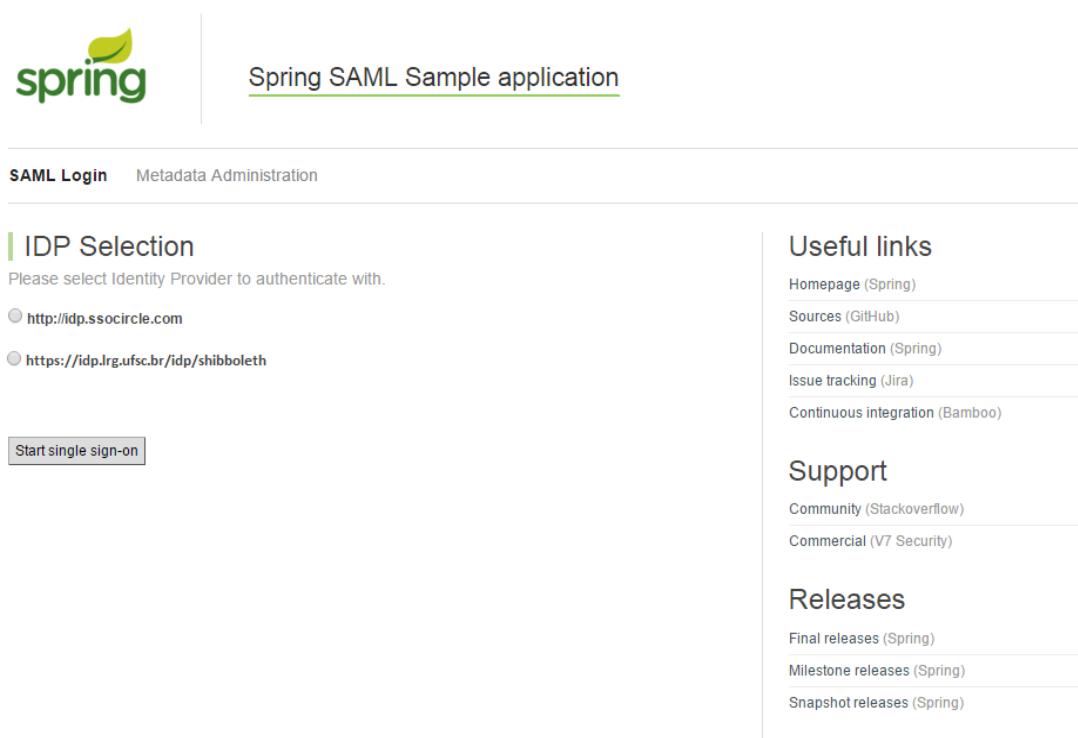
Figura 34: Alterações no arquivo server.xml do Tomcat

Após as alterações, é necessário reiniciar o Tomcat.

4.3. Execução

Após todas as configurações feitas, a aplicação já está pronta para executar em um ambiente seguro atuando como provedor de serviços trocando mensagens SAML com o provedor de identidades.

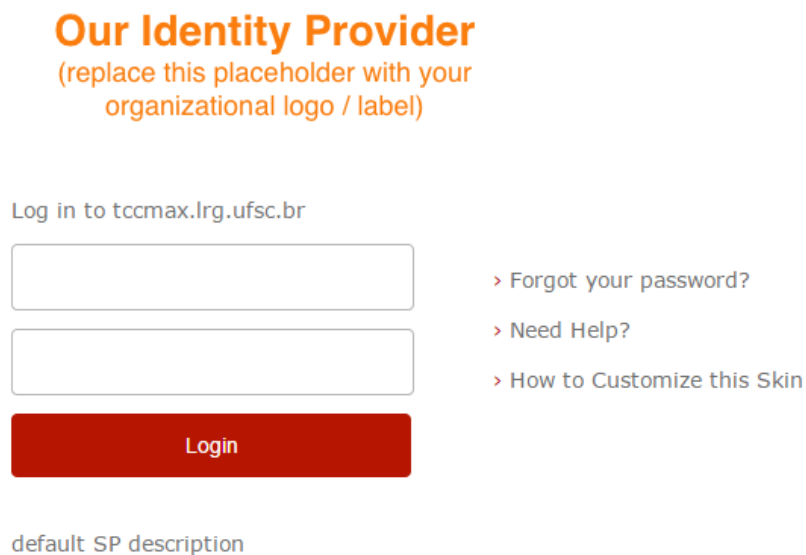
Acessando o endereço da aplicação <https://tccmax.lrg.ufsc.br/spring-security-saml2-sample/>, a aplicação direcionará para o *Discovery Service*, onde estará listado os provedores de identidade disponíveis para autenticação, como demonstra a **Figura 35**.



The screenshot displays the 'Spring SAML Sample application' interface. At the top left is the Spring logo. The main heading is 'Spring SAML Sample application'. Below this, there are two navigation links: 'SAML Login' and 'Metadata Administration'. The central section is titled 'IDP Selection' and contains the instruction 'Please select Identity Provider to authenticate with.' There are two radio button options: 'http://idp.ssocircle.com' and 'https://idp.lrg.ufsc.br/idp/shibboleth'. A 'Start single sign-on' button is located below these options. On the right side, there are three sections: 'Useful links' with links to Homepage (Spring), Sources (GitHub), Documentation (Spring), Issue tracking (Jira), and Continuous integration (Bamboo); 'Support' with links to Community (Stackoverflow) and Commercial (V7 Security); and 'Releases' with links to Final releases (Spring), Milestone releases (Spring), and Snapshot releases (Spring).

Figura 35: Lista dos provedores de identidade

Selecionando o provedor configurado com os metadados e clicando no botão “Start single sign-on”, a aplicação redirecionará para a página de autenticação do provedor de identidades, como mostra a **Figura 36**.



Our Identity Provider
(replace this placeholder with your
organizational logo / label)

Log in to tccmax.lrg.ufsc.br

- > [Forgot your password?](#)
- > [Need Help?](#)
- > [How to Customize this Skin](#)

default SP description

Figura 36: Página de autenticação do IdP

Entrando com o login e senha corretos, o recurso é liberado para acesso. Nesse caso mostra os atributos no usuário vindos do provedor de identidades para serem utilizados na aplicação. A **Figura 37** ilustra os atributos.

Authenticated user

Overview of the authenticated user's data.

General information

Name:	_09bc2907a83e685f8f57a97dffdf5a6a
Principal:	_09bc2907a83e685f8f57a97dffdf5a6a
Name ID:	_09bc2907a83e685f8f57a97dffdf5a6a
Name ID format:	urn:oasis:names:tc:SAML:2.0:nameid-format:transient
IDP:	https://idp.lrg.ufsc.br/idp/shibboleth
Assertion issue time:	2014-10-29T00:36:05.866Z

Principal's SAML attributes

Subject confirmation

Method:	urn:oasis:names:tc:SAML:2.0:cm:bearer
In response to:	a25je2efhjhd5f317be88gjhd673
Not on or after:	2014-10-29T00:41:05.866Z
Recipient:	https://tccmax.lrg.ufsc.br/spring-security-saml2-sample/saml/SSO

Authentication statement

Authentication instance:	2014-10-29T00:36:05.859Z
Session validity:	
Authentication context class:	urn:oasis:names:tc:SAML:2.0:ac:classes>PasswordProtectedTransport
Session index:	_bcf05a729eed8271015158acc1d1c6d4
Subject locality:	150.162.198.58

Conditions

Not before:	2014-10-29T00:36:05.866Z
Not on or after:	2014-10-29T00:41:05.866Z
Audience restriction:	https://tccmax.lrg.ufsc.br/spring-security-saml2-sample/saml/metadata

Global Logout Local Logout

Figura 37: Atributos do usuário

E por fim, clicando em “Global Logout”, a aplicação faz o logout destruindo a sessão local e a sessão do provedor de identidades. Desse modo, outros provedores de serviço que também estivessem utilizando a sessão, seriam

invalidados. Por outro lado, se quisesse somente destruir a sessão local, seria indicado clicar em “Local Logout”.

5. Considerações Finais

Cada vez mais se torna necessária a utilização de identidades digitais, nos mais diversos cenários. E para dar suporte a essa utilização é necessário que existam sistemas de gerenciamento de identidades seguros e confiáveis.

Entre essas características de segurança, a privacidade é uma das mais preocupantes, por afetar diretamente os usuários dos serviços e pelo grande número de incidentes recentes que podem ser encontrados envolvendo vazamento de informações pessoais.

Junto ao problema de segurança, há informações de atributos que podem ser reaproveitadas sem a necessidade da intervenção do usuário reduzindo a redundância na interação com o usuário. Além disso, o usuário deve ter total controle de seu acesso, ou seja, quando solicitar a saída, sua sessão deverá ser encerrada em todos os módulos.

A solução proposta abrange esses três problemas de forma segura baseando-se em modelos já existentes e que possam complementar a arquitetura Shibboleth.

O Shibboleth ainda é uma arquitetura muito robusta que possui algumas limitações. Podemos citar como exemplo a liberação de atributos para o provedor de serviços utilizando apenas uma única política de liberação. Outra limitação se trata sobre o controle dos atributos, pois quando estão sendo liberados, o usuário não consente explicitamente quais atributos estão sendo liberados.

O entendimento do funcionamento e a configuração do ambiente em um servidor web Apache com o módulo Shibboleth SP se torna mais complexo por

não ter documentação bem definida e explícita. A adaptação e a integração com aplicações nativas em outras linguagens também se torna desfavorável devido a esse mesmo fator.

Para contornar algumas dessas insuficiências, surgiram bibliotecas que manipulam mecanismos de autenticação e autorização, que é o caso do Spring Security, deixando a configuração e a integração com a arquitetura Shibboleth mais simplificada e unificada. A maior parte da configuração é feita em um arquivo *.xml* com os elementos de segurança de forma declarativa. A fácil integração com aplicações Java nativas auxilia muito o trabalho do desenvolvedor.

6. Trabalhos Futuros

Com base no trabalho desenvolvido, algumas vertentes de trabalhos futuros podem ser identificadas, como o cenário 1 proposto na seção 3.8.6. Integração da Aplicação com Shibboleth SP que apenas foi citado e não implementado ou uma criação complexa de uma estrutura de federação completa com vários provedores de identidades e muitos provedores de serviços, já que foi abordado um cenário característico.

A exploração da autorização também é extremamente importante para proteção e gerenciamento de perfis que utilizam os recursos do gerenciamento de identidades.

7. Referências

- [1] REINERT, Claiton Enilson. **Estudos sobre gerenciamento de identidades e acesso à web**. 115f. (Trabalho de Conclusão de Curso) – Bacharelado em Ciência da Computação, Universidade do Vale do Itajaí, Itajaí, 2005.
- [2] HOVAV, Anat and Berger Ron. **Tutorial: Identity Management Systems and Secured Access Control**. Communications of the Association for Information Systems, Vol. 25, article 42, 2009.
- [3] Joni da Silva Fraga, Michelle Wingham, Emerson Ribeiro de Mello, Davi Böger, Marlon Gueiros. **Gerenciamento de Identidades Federadas**. Minicursos do Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais. SBSeg'2010, Fortaleza, CE, Brazil, Outubro 2010.
- [4] SANTOS, Alfredo. **Gerenciamento de Identidades – Segurança da Informação**. Rio de Janeiro: Brasport, 2007.
- [5] Jøsang, A. e Pope, S. (2005). **User centric identity management**. In AusCERT Asia Pacific Information Technology Security Conference 2005.
- [6] Carmody, S., Erdos, M., Hazelton, K., Hoehn, W., Morgan, B., Scavo, T., e Wasley, D. (2005). **Incommon technical requirements and information**. vol. 2005.

[7] Chadwick, D. (2009). **Federated identity management. Foundations of Security Analysis and Design V**, pages 96–120.

[8] Camenisch, J. e Pfitzmann, B. (2007). **Security, Privacy, and Trust in Modern Data Management, chapter Federated Identity Management**, pages 213–238. Springer Verlag.

[9] Bhargav-Spantzel, A., Camenisch, J., Gross, T., e Sommer, D. (2007). **User centrlicity: a taxonomy and open issues**. Journal of Computer Security, 15(5):493–527.

[10] IBM (2005). **Federated Identity Management and Web Services Security with IBM Tivoli Security Solutions**. IBM, second edition.

[11] Shibboleth. **What's Shibboleth**. Disponível em <<http://shibboleth.net/about/>>. Acesso em 30 de agosto de 2013.

[12] BONETTI, T. M. **Integração entre Portais de Serviço de Grade e Shibboleth usando o GridShib**. Relatório Final de Iniciação Científica (PIBIC - CNPq) - Departamento de Informática e Estatística, Universidade Federal de Santa Catarina. Florianópolis, 2010.

[13] CORDOVA, A. S. **Aplicação Prática de um Sistema de Gerenciamento de Identidades**. Trabalho de Conclusão de Curso (Bacharelado em Ciências da

Computação) - Centro de Ciências Tecnológicas da Terra e do Mar, Universidade do Vale do Itajaí. Itajaí, 2006.

[14] OASIS. **About Us**. Disponível em: <<http://www.oasis-open.org/org>>. Acesso em 31 agosto 2013.

[15] OASIS. **Security Assertion Markup Language (SAML) 2.0 Technical Overview**. Disponível em: <<http://www.oasis-open.org/committees/download.php/11511/sstc-saml-tech-overview-2.0-draft-03.pdf>>. Acesso em 31 agosto 2013.

[16] Dashboard. **Shibboleth**. Disponível em <<https://wiki.shibboleth.net/confluence/dashboard.action>>. Acesso em 31 de agosto de 2013.

[17] TODOROV, D. **Mechanics of User Identification and Authentication**. Boca Raton: Auerbach Publications, 2007.

[18] WINDLEY, P. **Digital Identity**. Cambridge: O'Reilly Media, Inc., 2005.

[19] Damiani, E., di Vimercati, S. D. C., e Samarati, P. (2003). **Managing multiple and dependable identities**. In IEEE Internet Computing, pages 29–37. IEEE.

[20] Ian Goldberg, David Wagner, and Eric A. Brewer. **Privacy Enhancing Technologies for the Internet**. In COMPCON '97, pages 103–109, February 1997.

[21] BERTINO, Elisa and TAKAHASHI, Kenji. **Identity Management - Concepts, Technologies, and Systems**, 2011, Artech House

[22] Federação CAFe. **Projetos Especiais: Implantação de um provedor de identidades**. Página 11. Disponível em <<http://esr.rnp.br/publicacoes/seguranca/cafe?download=ad61ab143223efbc24c7d2583be69251>>. Acesso em 29 de setembro de 2013.

[23] OASIS (2005f). **Profiles for the OASIS SAML V2.0**. Organization for the Advancement of Structured Information Standards (OASIS).

[24] Westin, A. **Privacy and Freedom**. New York: Atheneum, 1967.

[25] Tsukada, Y., et al., “**Anonymity, Privacy, Onymity, and Identity: A Modal Logic Approach**”, *Proc. 11th IEEE Int. Conf. Computacional Science and Engineering*, August 2009, pp. 42-51.

[26] Shibboleth. **Documentation Shibboleth**. Disponível em <<https://wiki.shibboleth.net/confluence/display/SHIB2/ShibEnabled>>. Acesso em outubro de 2013.

[27] Shibboleth. **How Shibboleth Works: Basic Concepts**. Disponível em <<http://shibboleth.net/about/basic.html>>. Acesso em outubro de 2013.

[28] CAS. CAS about. Disponível em <<http://www.jasig.org/cas/cas2-architecture>>. Acesso em outubro de 2013.

[29] Shibboleth. **Flows and Config**. Disponível em <<https://wiki.shibboleth.net/confluence/display/SHIB2/FlowsAndConfig> >. Acesso em outubro de 2013.

[30] CAS. **CAS Wiki**. Disponível em <http://en.wikipedia.org/wiki/Central_Authentication_Service>. Acesso em outubro de 2013.

[31] Jasig. **What is Jasig**. Disponível em <<http://www.jasig.org/about>>. Acesso em novembro de 2013.

[32] HARRIS, S. CISSP All-in-One Exam Guide. 4. edição. New York: McGraw Hill, 2008.

[33] Spring Security Extension. **Spring Security Quick Start**. Disponível em <<http://projects.spring.io/spring-security/>>. Acesso em setembro de 2014.

[34] Wiki Shibboleth. **NativeSPEnableApplication**. Disponível em <<https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPEnableApplication>>. Acesso em setembro de 2014.

[35] Spring Security. **Projetos**. Disponível em <<http://projects.spring.io/spring-security/>>. Acesso em outubro de 2014.

[36] Wikipédia. **Spring Security**. Disponível em <http://en.wikipedia.org/wiki/Spring_Security>. Acesso em outubro de 2014.

[37] Spring. **Referência Spring Security**. Disponível em <<http://docs.spring.io/spring-security/site/docs/3.2.5.RELEASE/reference/htmlsingle/>>. Acesso em outubro de 2014.

[38] Tomcat. **SSL Configuration HOW-TO**. Disponível em <<http://tomcat.apache.org/tomcat-7.0-doc/ssl-howto.html>>. Acesso em outubro de 2014.

[39] Spring Security. **Spring Security SAML Extension**. Disponível em <<http://docs.spring.io/autorepo/docs/spring-security-saml/1.0.x-SNAPSHOT/reference/htmlsingle/>>. Acesso em outubro de 2014.

[40] Joni da Silva Fraga, Michelle Wingham, Emerson Ribeiro de Mello, Davi Böger, Marlon Gueiros - "**Gerenciamento de Identidades Federadas**", em

Minicursos do Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais - SBSeg'2010, Fortaleza, CE, Brasil, Outubro 2010.

[41] SBSEG. **Uma aplicação de privacidade no gerenciamento de identidades em nuvem com uApprove.**

<<http://www.peotta.com/sbseg2011/resources/downloads/wticg/91986.pdf>>.

Acesso em outubro de 2013.

[42] UFRGS. **Federação Chimarrão.** <<http://www.chimarrao.ufrgs.br>>. Acesso em outubro de 2013.

[43] Shibboleth. **How Shibboleth Works: Basic Concepts.** Disponível em <http://shibboleth.net/about/basic.html>>. Acesso em dezembro de 2014.

[44] OASIS. **SAML.** Disponível em <http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0-cd-02_html_73dc0b55.gif>. Acesso em outubro de 2013.

[45] OASIS. **SAML.** Disponível em <<http://t2.gstatic.com/images?q=tbn:ANd9GcR3GZkbsViTPM9eI3SCB5e8jzAmXP5TFFT4W4tOmiLv3zJ0AfpSA>>. Acesso em outubro de 2013.

[46] RND. **Wireless Authentication using Feide.** Disponível em <<https://rnd.feide.no/doc/feide-eduroam.pdf>>. Acesso em dezembro de 2014.

[47] GitHub. **A simple Java application which connects to an SSL/TLS port and extracts all certificates presented by the server to PEM formatted files.**

Disponível em <<https://github.com/vschafer/ssl-extractor>>. Acesso em dezembro de 2014.