

Autenticação com Biometria Usando a Plataforma Java

Fernando Lozano

www.lozano.eti.br

<fernando@lozano.eti.br>

Arquiteto de Soluções – Neki Technologies

www.neki.com.br

Linux.Java.Net Community

Sobre o Autor

- ▶ Consultor Independente com mais de 10 anos de atuação na área de TI, em Sistemas de Informação e Integração de Redes
- ▶ Detentor de diversas certificações profissionais da Microsoft, IBM, Sun, Red Hat e LPI
- ▶ Ex-conselheiro do Linux Professional Institute
- ▶ Colunista da Revista Java Magazine
- ▶ Colaborador do IBM Developer Works e NetBeans Magazine
- ▶ Community Manager do Java.Net
- ▶ Webmaster / Coordenador de Tradução da Free Software Foundation
- ▶ Autor do livro “Java em GNU/Linux”, ed. Alta Books

Patrocinador



“Missão: Otimizar Resultados”
Especialista em soluções open source
baseadas na plataforma Java



Consultoria em Desenvolvimento
e Infra-Estrutura de Sistemas



Capacitação e Mentoring
em Tecnologia e Processos

Parcerias Neki



Gerenciamento de Riscos
Soluções em Segurança e
Biometria



Professional Open Source

Agenda

- ▶ Fundamentos de autenticação biométrica
- ▶ Recursos para autenticação da plataforma Java
- ▶ Biometria em aplicações Java SE / Java EE (RMI)
- ▶ Biometria em aplicações Web Java EE

Fundamentos de Autenticação Biométrica

- ▶ As características biométricas (impressões digitais, padrão da retina, formato da face, etc) são convertidas em uma representação matemática (vetorial) conhecida como **minúcias**
- ▶ As minúcias obtidas pelo dispositivo biométrico são então comparadas em modo *fuzzy* com um conjunto de **templates** armazenado em um **diretório**
- ▶ Se houver as minúcias forem suficientemente semelhantes a um dos templates ocorre um *match* e a autenticação é considerada válida

Recursos de Segurança do Java EE

- ▶ Security Manager
- ▶ JCA – Java Cryptography Architecture
- ▶ JAAS – Java Authorization and Authentication System
- ▶ JACC – Java Authentication Contract for Containers
- ▶ Segurança declarativa para Servlets e EJBs
- ▶ Pouco usados na prática pelos desenvolvedores Java, por simples desconhecimento

Security Manager

- ▶ Recurso que permite a criação de *sandboxes*, como a utilizada para executar applets Java dentro de um navegador web
- ▶ Permite (ou não) a invocação de métodos de acordo com a origem do chamador
- ▶ Cada método sujeito a autorização deve interagir explicitamente com o Security Manager
- ▶ Métodos da API padrão do Java SE usam este recurso

JCA, JCE, JSSE, GSS, PKI

- ▶ Suportam vários algoritmos de criptografia simétrica e assimétrica
- ▶ Implementam os protocolos SSL e TLS para comunicação segura
- ▶ Permitem a criação e manipulação de certificados e assinaturas digitais, além de chaves públicas e privadas
- ▶ Suportam mecanismos baseados em tokens como o Kerberos

JAAS

- ▶ Permite criar módulos de login vinculados a diferentes bases de dados de usuários (arquivos, bancos de dados, diretórios LDAP, etc)
- ▶ Isola estes módulos da interface com o usuário (ou do protocolo de rede) para obter login, senha, tokens, biometria ou outras formas de credenciais
- ▶ Permite empilhar módulos no estilo do PAM do Unix

Módulos (JAAS) de Login Empilháveis

- ▶ Permitem a composição e reuso de módulos especializados como parte de uma política mais ampla de autorização
- ▶ Permite ainda configurar mecanismos de autenticação alternativos
- ▶ Por exemplo, um módulo de login verifica senhas contra um banco de dados, se isto não funcionar a senha é verificada contra uma base LDAP, e em seguida o login é autorizado apenas em horário comercial

JACC

- ▶ Define como um container Web ou EJB deve fornecer o contexto de uma requisição remota e sessão de usuário para um Security Manager
- ▶ Permite definir regras de autorização baseada nos argumentos da requisição (parâmetros HTTP ou argumentos de métodos remotos)
- ▶ Elimina a maioria dos casos em que é necessário inserir lógica programática de segurança
- ▶ Infelizmente o Security Manager tem escopo da JVM e não do deployment WAR / EJB-JAR / EAR

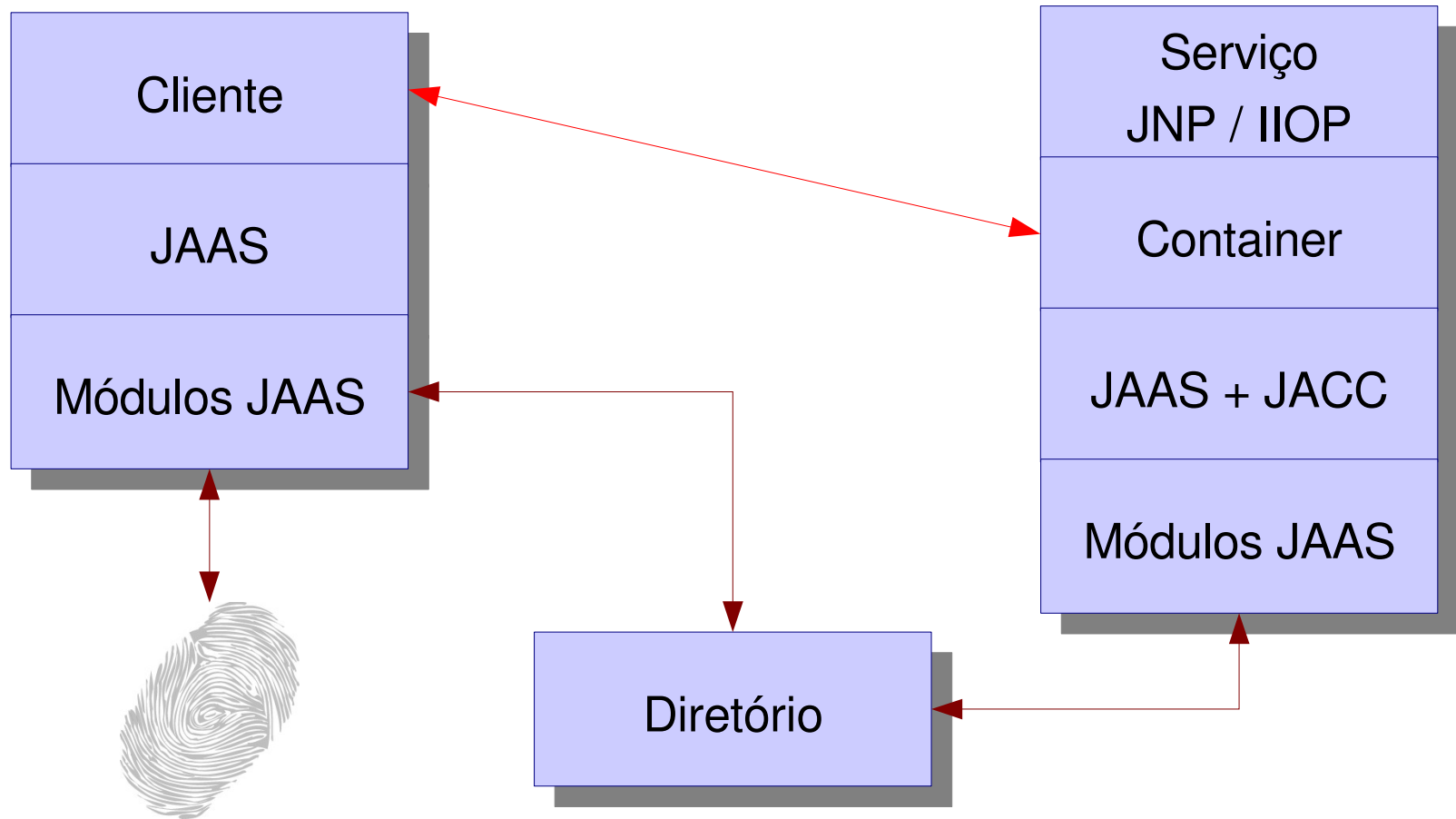
Infra-Estrutura x Aplicação

- ▶ Security Manager, JAAS e JCA são APIs de infraestrutura fornecidas pelo Java SE
- ▶ O JAAS é mandatório no Java EE 1.3, e o JAAC no Java EE 1.4
- ▶ Elas atendem à programação de containers: servidores Java EE e frameworks de interface gráfica como o NetBeans Platform ou o Eclipse RCP
- ▶ O programador de aplicações informação não deveria lidar diretamente com elas

Segurança Declarativa

- ▶ Estabelece a visão do programador de aplicação sobre os recursos de segurança
- ▶ Estipula que as regras de controle de acesso sejam definidas pela associação de recursos (URLs, métodos remotos) a grupos (roles) que autorizados a acessa-los
- ▶ Esta associação é especificada no descritor de deployment de um pacote WAR ou EJB-JAR
- ▶ O container tem liberdade em como configurar o mapeamento de usuários a roles

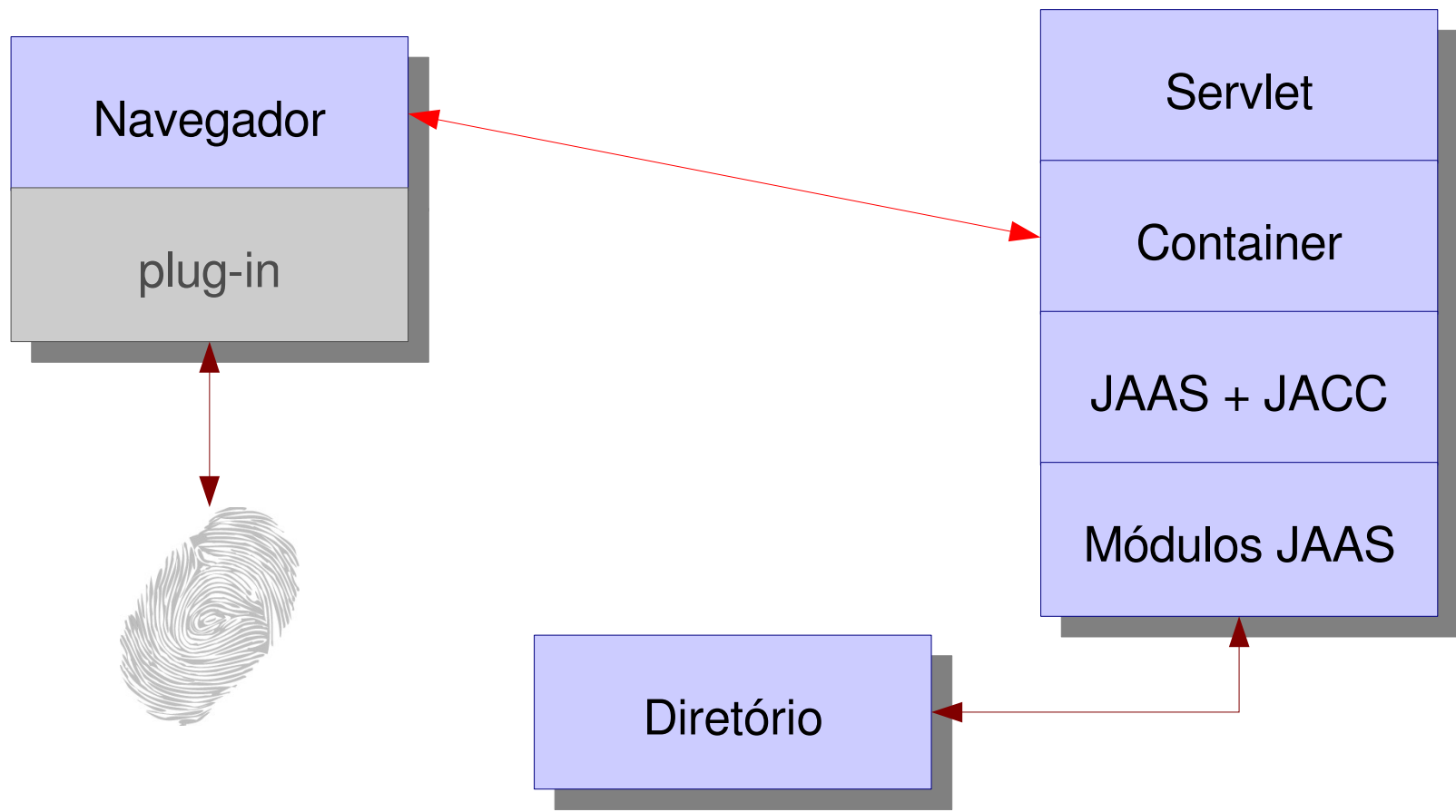
Juntando Tudo (RMI)



Biometria com RMI

- ▶ Um módulo JAAS no cliente faz a interface com o dispositivo de biometria
- ▶ Ele pode fazer o match das minúcias e então obter um token, que é enviado para o servidor
- ▶ Ou pode enviar as minúcias diretamente para o servidor
- ▶ Em ambos os casos, um módulo JAAS no servidor reconhece o token ou as minúcias como válidos

Juntando Tudo (Web)



Biometria com HTTP

- ▶ Não é suportada diretamente
- ▶ Depende de suporte específico no cliente (navegador)
- ▶ E também de módulos específicos no servidor (container)

Como Inserir Biometria na Web?

- ▶ Um programa / plug-in no cliente obtém as minúcias (por meio do dispositivo biométrico)
- ▶ A autenticação biométrica é entregue ao SO ou outro mecanismo baseado em token
- ▶ Um proxy web ou módulo JAAS no servidor tem que reconhecer o token ou minúcias como válidos
- ▶ O problema é instalar nas estações dos usuários o plug-in que lida com a biometria (e garantir sua interoperabilidade)

Biometria + Certificados Digitais

- ▶ O HTTP (via SSL/TLS) suporta certificados digitais X.400, baseados em PKI
- ▶ A autenticação baseada em certificados digitais é suportada pelo Java EE
- ▶ Assim o certificado substitui o token e elimina a necessidade de um módulo JAAS customizado
- ▶ Se o certificado for armazenado em um SmartCard (suportado pelos principais navegadores), o usuário tem liberdade de movimentos

Perguntas?

fernando@lozano.eti.br

www.lozano.eti.br