Simplificando a segurança de sua aplicação com Java EE

Leonardo Zanivan @leonardopanga

JavaOne Latin America 2015 - SES16317

Palestrante

@leonardopanga
github.com/panga

- Especialista em Arquitetura de Software
- Entusiasta em segurança da informação
- Contribuidor em projetos OSS
- Membro do JCP
- Arquiteto na Trier Sistemas

Agenda

- Conceitos
- Java EE
- Exemplos
- Problemas
- JSR 375

Segurança da Informação

- O que é?
- O que proteger?
- Quanto custa?

Características Principais

- Confidencialidade
- Integridade
- Disponibilidade
- Autenticidade

ISO/IEC 27000:2014

www.praxiom.com/iso-27000-definitions.htm

Mecanismos de Proteção

- Controle de acesso
  - Autenticação e autorização
- Criptografia
  - Encriptação, hashing e certificado digital
- Governança
  - Políticas, administração e auditoria
- Escrever código seguro
  - Validar input, prevenir XSS e SQL Injection

Recomendações

- OWASP Top 10 Java EE https://goo.gl/EaWOvN
  “Os 10 riscos mais críticos para aplicações web baseadas em Java EE” - OWASP, 2010
- Tente não inventar a sua própria segurança
- Fique atento as normas regulatórias
  HIPPA, PCI, SOX, ISO/IEC 27000, etc.

SSL (Secure Sockets Layer)

"SSL (Secure Sockets Layer) is a standard security technology for establishing an encrypted link between a server and a client—typically a web server (website) and a browser; or a mail server and a mail client (e.g., Outlook)."

DigiCert, 2015.

Let's Encrypt

*"Let's Encrypt is a **free**, **automated**, and **open** certificate authority brought to you by the Internet Security Research Group (ISRG)"* ***Arriving September 2015***

```
$ apt-get install lets-encrypt
$ lets-encrypt run
```

- Emissão cert. automática
- Apache ou nginx auto conf.

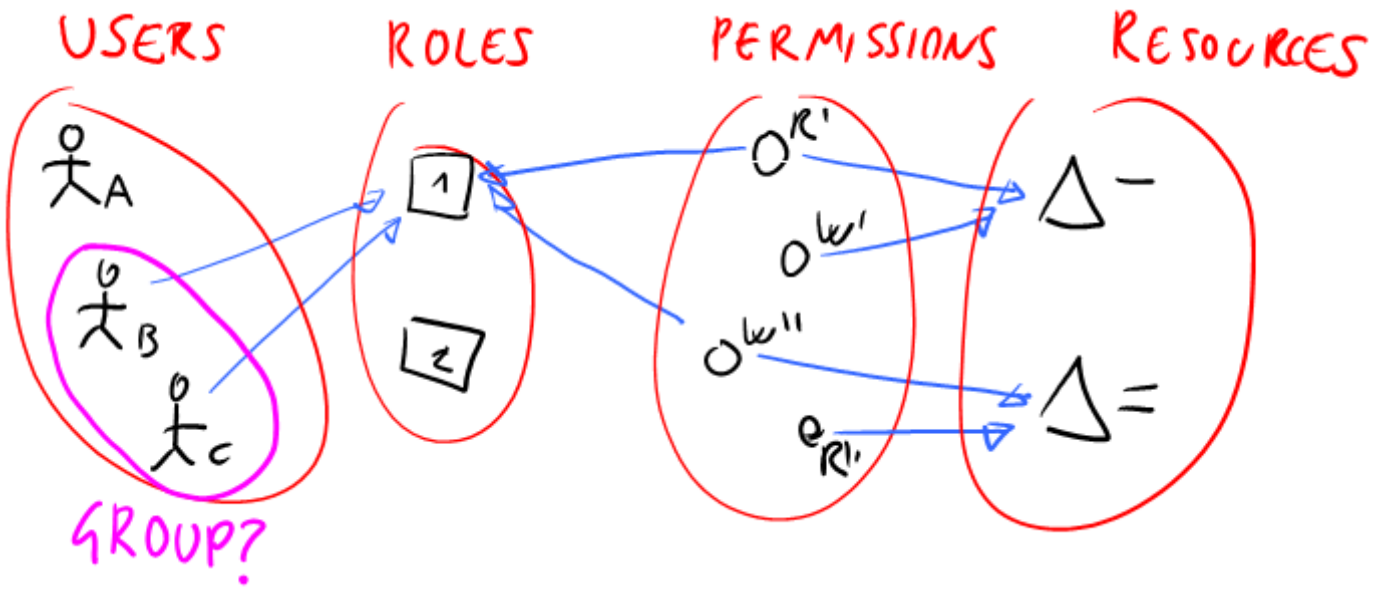
Iron-Clad Java: Building Secure Web Applications

Questionamentos

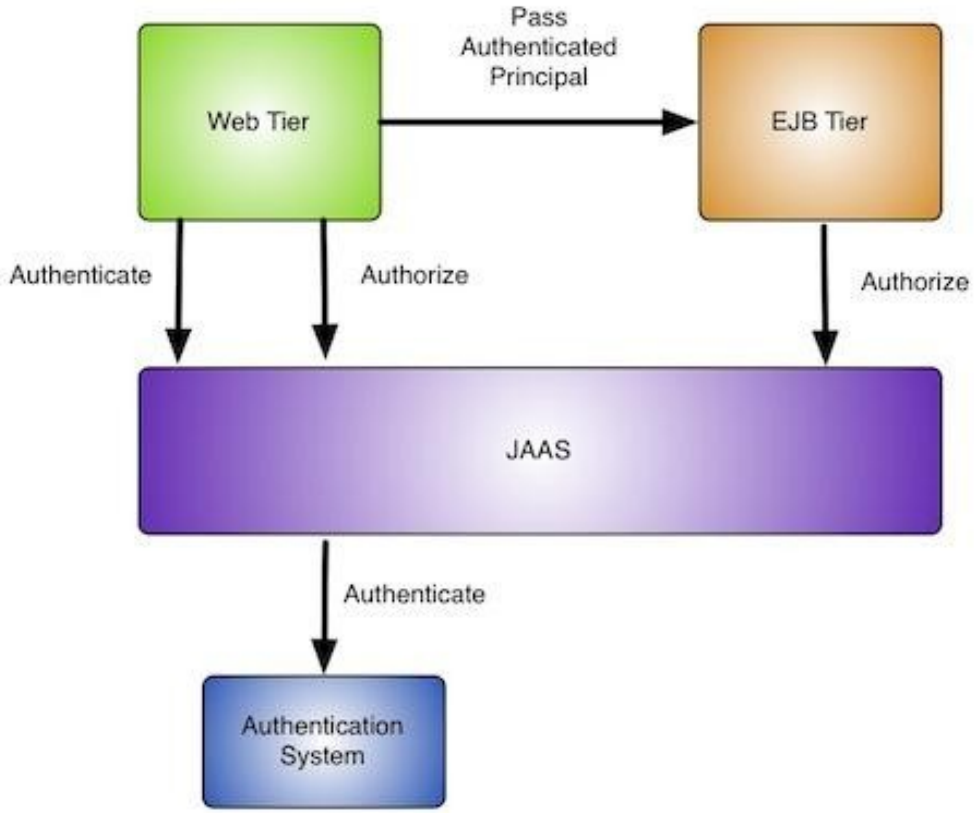
- Como o Java EE endereça a segurança?
- Java EE atende aos meus requisitos “complexos” de segurança?
- As especificações de segurança da plataforma Java EE são suficientes?

Segurança no Java EE

- Baseado em *Role-Based Access Control*
- Protege **URL**, **método EJB** ou **conteúdo**
- Utiliza conceito de módulos de autenticação
  - Database, LDAP, XML, etc.
- Pode ser declarativa (XML, annotations) ou programática (API)
- Configuração específica por **vendor**



RBAC - *Role-Based Access Control*



Fluxo de segurança no Java EE

Definições

- *Subject*: Usuário.
- *Role*: Controle de acesso.
- *Group*: Grupo de usuários com *roles* em comum.
- *Realm*: Repositório de usuários.
- *Principal*: Identidade (usuário, grupo ou role).
- *Credential*: Informações para autenticar um usuário.

SPEC de segurança no Java EE

- JAAS (J2EE 1.3)
  - *Java Authentication and Authorization Service*
- JACC (J2EE 1.4)
  - *Java Authorization Contract for Containers*
- JASPIC (Java EE 6)
  - *Java Authentication Service Provider Interface for Containers*

Autenticação BASIC

web.xml

```
<login-config>
    <auth-method>BASIC</auth-method>
    <realm-name>My realm</realm-name>
</login-config>
```

Autenticação FORM

web.xml

```
<login-config>
    <auth-method>FORM</auth-method>
    <form-login-config>
        <form-login-page>/login_form.jsp</form-login-page>
    <form-error-page>/login_error.jsp</form-error-page>
    </form-login-config>
</login-config>
```

login_form.jsp

```
<form action="j_security_check" method="POST">
    <input type="text" name="j_username" />
    <input type="password" name="j_password" />
</form>
```

Autorização Servlet Declarativa

web.xml

```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>Servlet</web-resource-name>
        <url-pattern>/SecureServlet</url-pattern>
    </web-resource-collection>
    <auth-constraint><role-name>admin</role-name></auth-constraint>
    <user-data-constraint>
       <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
<security-role>
    <role-name>admin</role-name>
</security-role>
```

Autorização Servlet Declarativa

SecureServlet.java

```
@WebServlet("/SecureServlet")
@ServletSecurity(
        @HttpConstraint(
                rolesAllowed = {"admin"},
                transportGuarantee = CONFIDENTIAL
        )
)
@DeclareRoles({"admin"})
public class SecureServlet extends HttpServlet {
}
```

Autorização Servlet Programática

SecureServlet.java

```
@WebServlet("/SecureServlet")
public class SecureServlet extends HttpServlet {

    protected void process(request, response) {
        if (request.isSecure()
                && request.getUserPrincipal() != null
                && request.isUserInRole("admin")) {
        }
    }
}
```

Autorização EJB Declarativa

ejb-jar.xml

```
<assembly-descriptor>
    <security-role>
    <role-name>admin</role-name>
    </security-role>
    <method-permission>
    <role-name>admin</role-name>
    <method>
        <ejb-name>AccountService</ejb-name>
            <method-name>withdraw</method-name>
        </method>
    </method-permission>
</assembly-descriptor>
```

Autorização EJB Declarativa

AccountService.java

```
@Stateless
@DeclareRoles({"admin"})
public class AccountService {

    @RolesAllowed({"admin"})
    public void withdraw(String account, BigDecimal amount) {
    }
}
```

Autorização EJB Programática

AccountService.java

```
@Stateless
public class AccountService {

    @Resource private EJBContext ejbContext;

    public void withdraw(String account, BigDecimal amount) {
        if (ejbContext.getCallerPrincipal() != null
                && ejbContext.isCallerInRole("admin")) {
        }
    }
}
```

Outras Especificações (API)

JAX-WS

```
@Context WebServiceContext context;
context.getUserPrincipal();
context.isUserInRole("role");
```

JAX-RS

```
@Context SecurityContext context;
context.getUserPrincipal();
context.isUserInRole("role");
```

JACC

```
(Subject)PolicyContext.getContext("javax.security.auth.Subject.container");
```

Outras Especificações (API)

JSF

```
@Inject ExternalContext context;
context.getUserPrincipal();
context.isUserInRole("role");
```

CDI

```
@Inject Principal principal;
```

WebSocket

```
public void onOpen(Session session) {
    session.getUserPrincipal();
}
```

Login portável com JASPIC

- Módulo SAM provido pela aplicação
- Registro via *ServletContextListener*
- Validação por *request*
- Retorna o usuário *Principal* e as *Roles*
- Integra com JAAS/JACC

https://github.com/javaee-samples/javaee7-samples

https://jaspic.zeef.com by Arjan Tijms

Em Produção....

- Segurança no Java EE é suficente?

- Interfaces de cadastro, login e administração
- Integração com login social (Google+, etc.)
- Integração com terceiros (SAML, OpenID)
- Multi-tenancy, Password Policy, Audit, 2FA ...

keycloak.jboss.org

KeyCloak

- Server roda no WF 8.2 ou EAP 6.4
- *PicketLink and Keycloak are merging!*
- Adaptadores JAAS para EAP/WF, Tomcat, Jetty
- Falta criar integração JAAS para Glassfish, WebLogic e outros, talvez com JASPIC?
- Elytron + KeyCloak no WildFly 10
- Não é um produto **ainda**

Problemas no JMS

- JMS não propaga nenhuma informação sobre segurança, não conhecemos o remetente e nem suas roles (*anonymous*)
- É preciso autenticar manualmente ao receber uma mensagem JMS

Problemas no WebSocket

- Uma sessão websocket autenticada não integra com EJB ou CDI (*anonymous*)
- Cadastrada issue na SPEC
  https://java.net/jira/browse/WEBSOCKET_SPEC-238

Workaround para JMS e WebSocket

- Biblioteca open-source para propagar a segurança do JMS e corrigir problemas do WebSocket nos containers JBoss/Wildfly

pom.xml

```
<dependency>
    <groupId>com.github.panga</groupId>
    <artifactId>jboss-security-extended</artifactId>
    <version>1.0.0</version>
</dependency>
```

Problemas no JASPIC

- Necessita de config.xml extra como *security domain* e *group to role* por *vendor* (jboss-web.xml, sun-web.xml)
- No JBoss/WildFly é necessário adicionar um security domain “dummy”
- Poucos *containers* estão certificados para o JASPIC 1.1 do Java EE 7

Java EE Security API - JSR 375

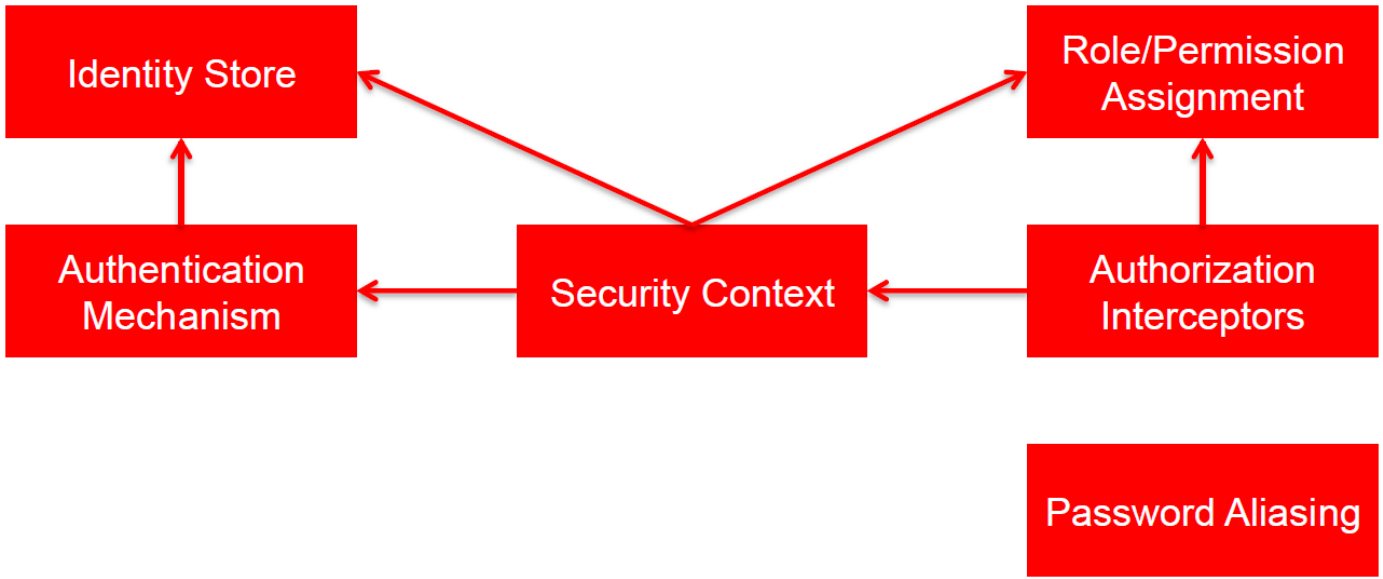
- Segurança foi o tema mais votado no Java EE 8 *community survey*
- Revisão da API de segurança no Java EE
- Padronização na implementação dos *vendors* para melhorar a portabilidade
- Suporte a PaaS/SaaS
- JASPIC como *first class citizen*

Escopo da JSR 375

EPIC	Description
Terminology	Establish terminology to enable accurate and concise communication
Authentication Mechanism	Simplify application-accessible authentication mechanisms
Identity Store	Standardize application-accessible identity store
Role/Permission Assignment	Standardize application-accessible role/permission assignment
Security Context	Standardize a platform-wide Security Context
Authorization Interceptors	Standardize platform-wide Authorization Interceptors
Password Aliasing	Standardize the API for using password aliases in configuration
Standardized Server Authentication Modules	Using the simplified Authentication Mechanism, standardize some additional ServerAuthModules

Veja o progresso e participe! https://java.net/projects/javaee-security-spec

Dependências entre as epics



Q&A

Obrigado!

@leonardopanga

github.com/panga