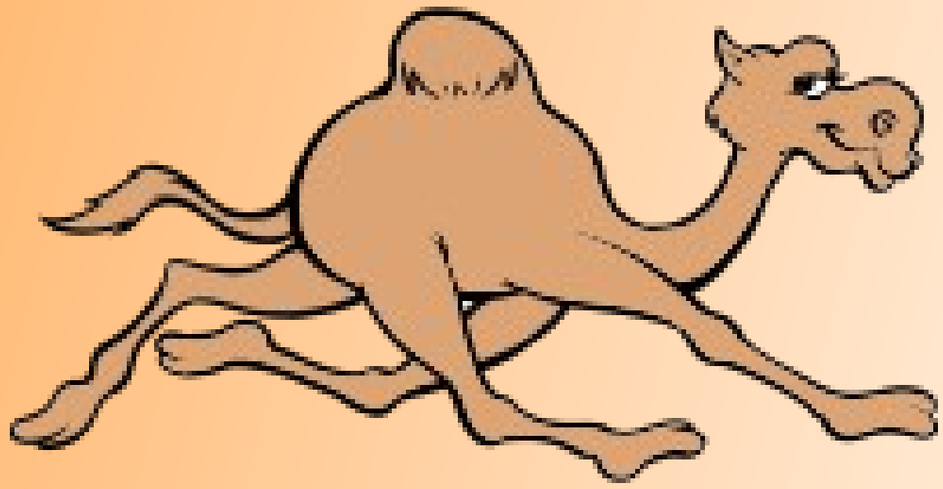
CamelOne 2013



Apache Karaf, ActiveMQ, Camel and CXF Security

Kenny Peeples
JBoss Technology Evangelist, Red Hat

Session Overview

Middleware Security for Apache CXF, Camel, ActiveMQ and Karaf as well as others continue to be an ongoing concern especially around Authentication, Authorization, Data at Rest and Data in Transit. The session will include a presentation and demonstrations of implementing Authentication (AuthN) and Authorization (AuthZ) as well as other security topics.

Speaker Bio

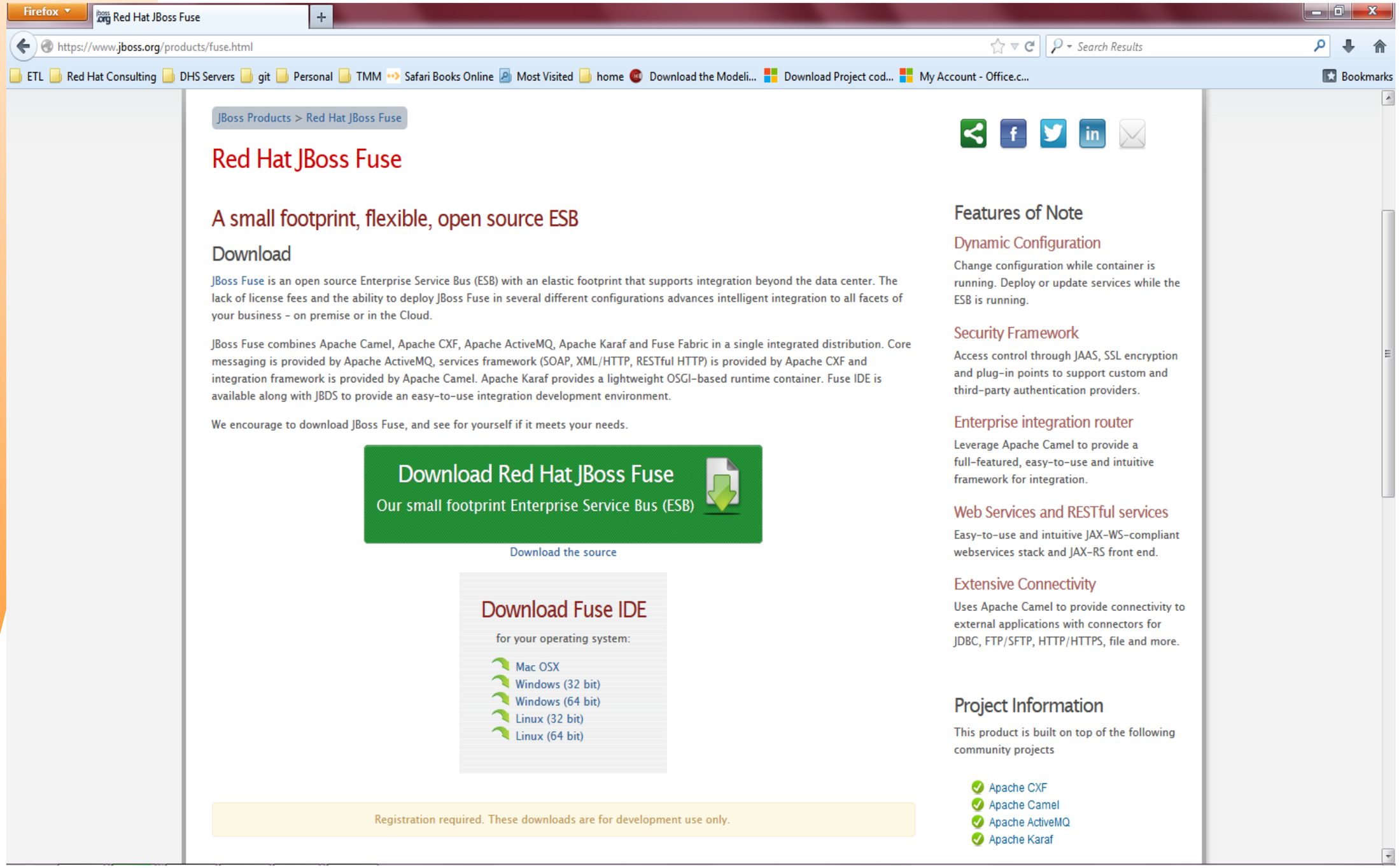
Kenneth has spent the last 7 years, with the last 3 being with Red Hat, working as a Consultant for the government for the Department of Defense, Department of Homeland Security and Intelligence Community. He designed, built and lead multiple projects including the Enterprise Security Federation Framework (ESF2), Information Assured Android Device (IA2D) and JBoss SCAP Content. During this time he has worked with Open Source Software with a focus on Red Hat and JBoss Products and Information Assurance. He also received his Security+ and Computer Hacking Forensic Investigator (CHFI) certifications.

Kenneth works in the Red Hat Middleware Product Business Unit as a JBoss Technology Evangelist with a focus on Fuse, SOA and EDS products.

Kenneth's Social Media Links
Website: http://people.redhat.com/kpeeples
Blogger: http://www.ossmentor.com

JBoss.org

Enterprise software downloads for developers

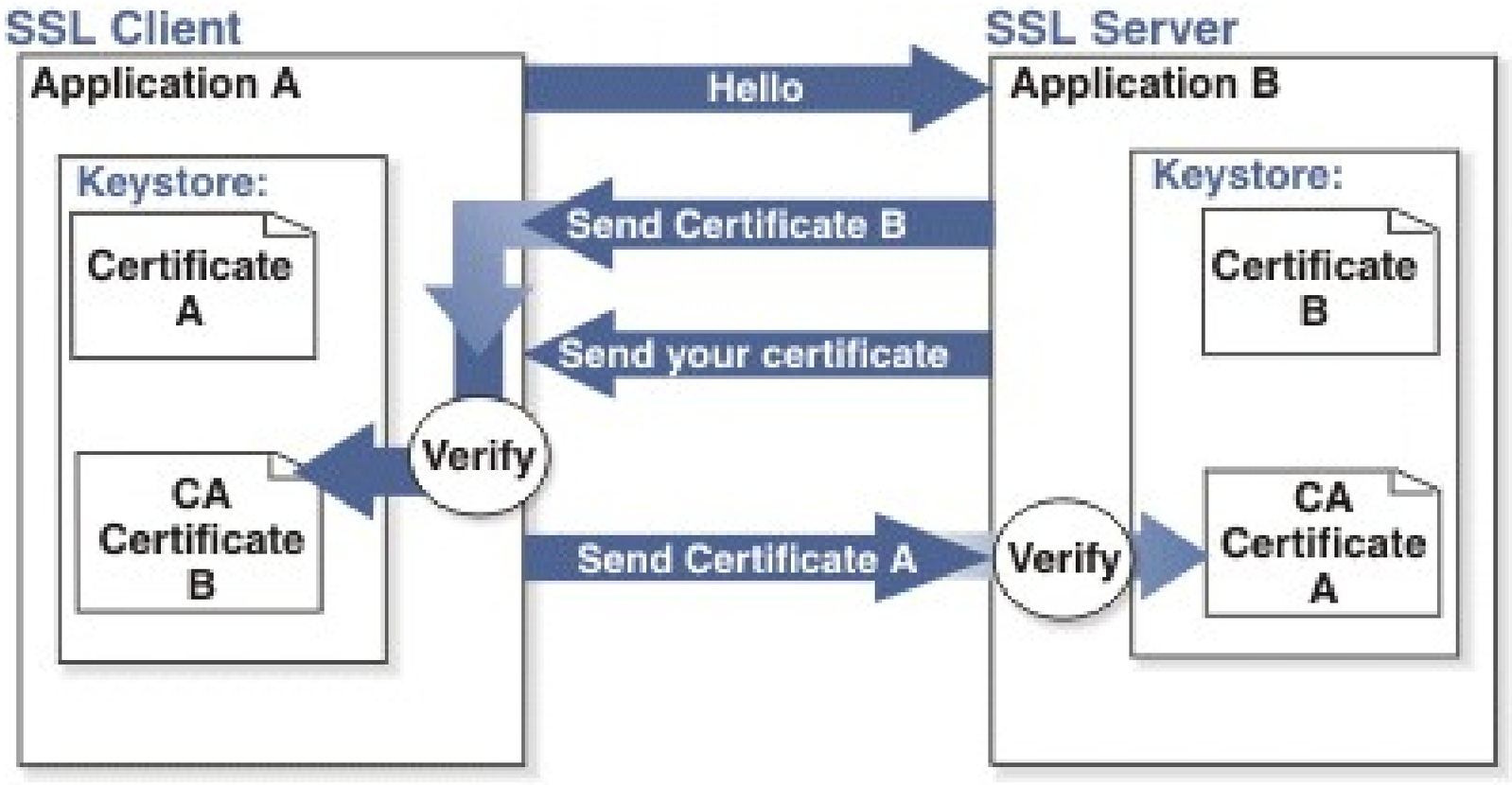


Middleware Security Concepts

- Authentication
- Authorization
- Auditing
- Encryption

2-way SSL

Mutual authentication or two-way authentication



Password Masking

When securing a container or its components it is undesirable to use plain text passwords in configuration files. One way to avoid this problem is to use encrypted property placeholders when ever possible. This is done in Fuse with the Java Simplified Encryption.

- http://www.jasypt.org/
- http://activemq.apache.org/encrypted-passwords.html
- https://access.redhat.com/site/documentation/en-US/JBoss_Fuse/6.0/html/Security_Guide/files/ESBSecurityEncryptProperties.html

What is JAAS?

The Java Authentication and Authorization Service (JAAS) can be used for two purposes:

- for authentication of users, to reliably and securely determine who is currently executing Java code, regardless of whether the code is running as an application, an applet, a bean, or a servlet; and
- for authorization of users to ensure they have the access control rights (permissions) required to do the actions performed.

http://docs.oracle.com/javase/7/docs/technotes/guides/security/jaas/JAASRefGuide.html

Apache ActiveMQ Security

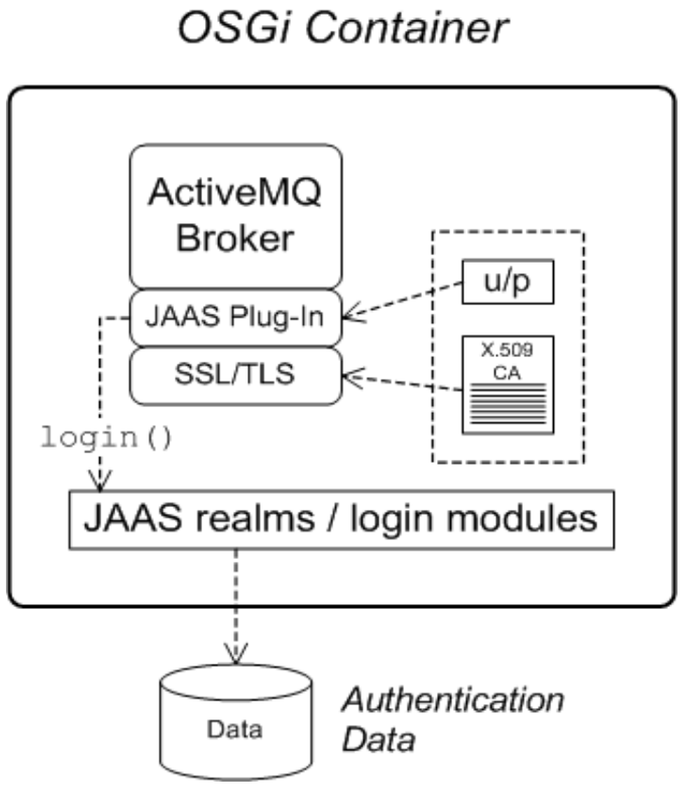
SSL/TLS security

Apache ActiveMQ supports the use of SSL/TLS to secure client-to-broker and broker-to-broker connections, where the underlying SSL/TLS implementation is provided by the Java Secure Socket Extension (JSSE).

JAAS security

Apache ActiveMQ also supports JAAS security, which typically requires clients to log on to the broker by providing username and password credentials.

https://access.redhat.com/site/documentation/en-US/JBoss_Fuse/6.0/html/Security_Guide/files/Arch-Architecture-ActiveMQ.html



Apache ActiveMQ Security (Continued)

To enable JAAS security in a broker, you install one of the supported JAAS plug-ins. Each of the JAAS plug-ins supports a different kind of credentials or implements a somewhat different login procedure.

The following JAAS plug-ins are currently supported by Apache ActiveMQ:

- jaasAuthenticationPlugin
- jaasCertificateAuthenticationPlugin
- JaasDualAuthenticationPlugin

Apache ActiveMQ Security (Continued)

Apache ActiveMQ provides a number of different JAAS login module implementations, which are used to define JAAS realms. The following JAAS login modules are currently implemented by Apache ActiveMQ:

- PropertiesLoginModule
- LDAPLoginModule
- GuestLoginModule
- TextFileCertificateLoginModule

Apache Camel Security

Camel offers several forms & levels of security capabilities that can be utilized on camel routes. These various forms of security may be used in conjunction with each other or separately.

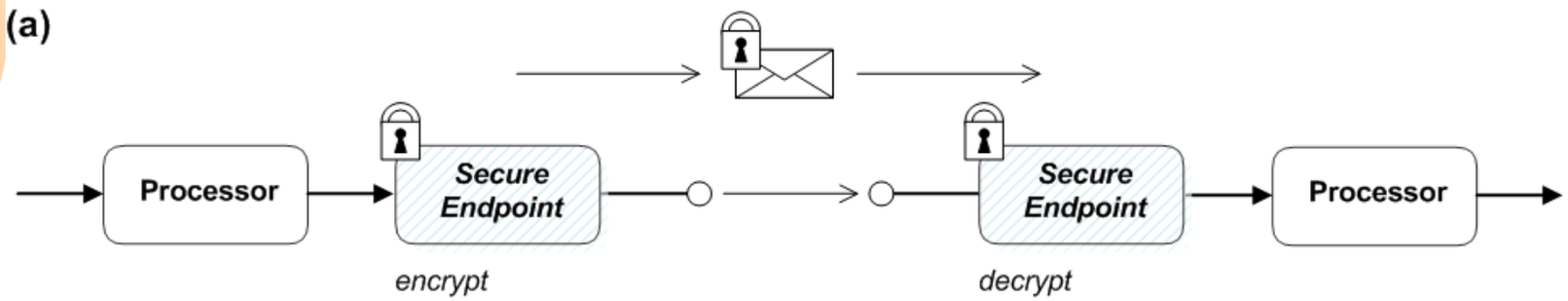
The broad categories offered are

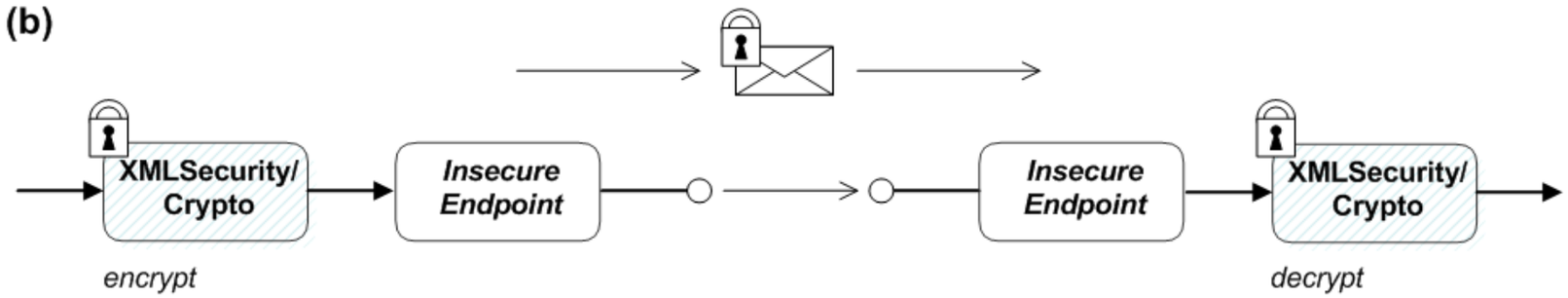
- Route Security
- Payload Security
- Endpoint Security
- Configuration Security

http://camel.apache.org/security.html

Apache Camel Security (Continued)

This figure shows an overview of the basic options for securely routing messages between Apache Camel endpoints.





https://access.redhat.com/site/documentation/en-US/JBoss_Fuse/6.0/html/Security_Guide/files/Arch-Architecture-Camel.html

Apache Camel Security (Continued)

As shown in the figure above, you have the following options for securing messages:

Endpoint security—part (a) shows a message sent between two routes with secure endpoints.

Payload security—part (b) shows a message sent between two routes where the endpoints are both insecure.

Apache Camel Security (Continued)

Some of the Camel components that currently support security are, as follows:

- JMS and ActiveMQ—SSL/TLS security and JAAS security for client-to-broker and broker-to-broker communication.
- Jetty—HTTP Basic Authentication and SSL/TLS security.
- CXF—SSL/TLS security and WS-Security.
- Crypto—creates and verifies digital signatures in order to guarantee message integrity.
- Netty—SSL/TLS security.
- MINA—SSL/TLS security.
- Cometd—SSL/TLS security.
- glogin and gauth—authorization in the context of Google applications.

Apache Camel Security (Continued)

Apache Camel provides the following payload security implementations, where the encryption and decryption steps are exposed as data formats on the marshal() and unmarshal() operations

- XMLSecurity data format

For more details, see http://camel.apache.org/xmlsecurity-dataformat.html.

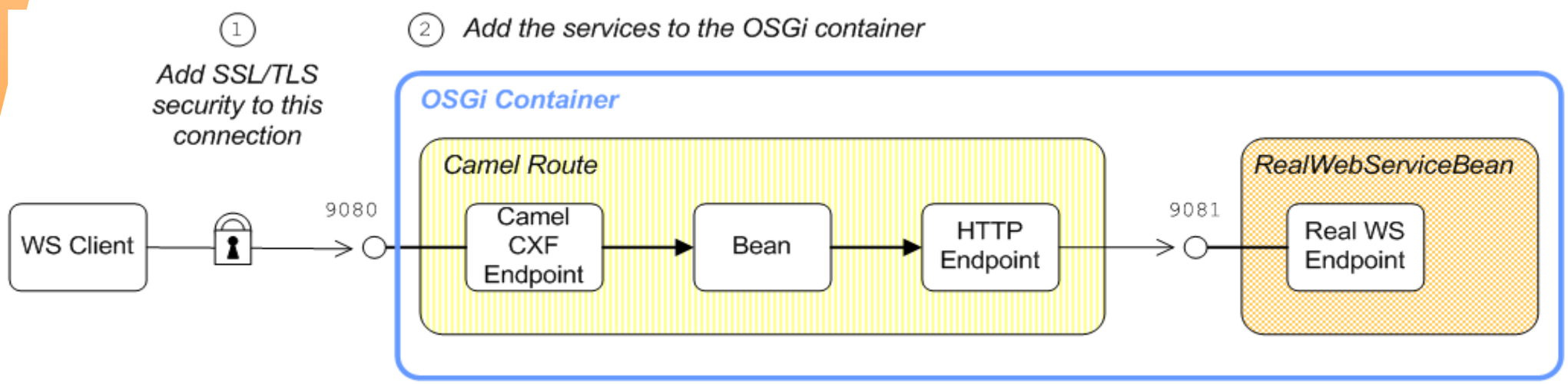
- Crypto data format

For more details, see http://camel.apache.org/crypto.html

Apache Camel Security Example

In order to explain how to secure a Camel CXF endpoint in OSGi, the tutorial below builds on an example available from the standalone distribution of Apache Camel, the Camel CXF proxy demonstration. The figure below gives an overview of how this demonstration works.

https://access.redhat.com/site/documentation/en-US/JBoss_Fuse/6.0/html/Security_Guide/files/CamelCXF-ProxyDemo.html



Karaf (OSGi) Container Security

JAAS realms - Red Hat JBoss Fuse supports a special mechanism for defining JAAS login modules (in either a Spring or a blueprint file)

karaf realm - The OSGi container has a predefined JAAS realm, the karaf realm. Red Hat JBoss Fuse uses the karaf realm to provide authentication for remote administration of the OSGi runtime, for the Web Console, and for JMX management.

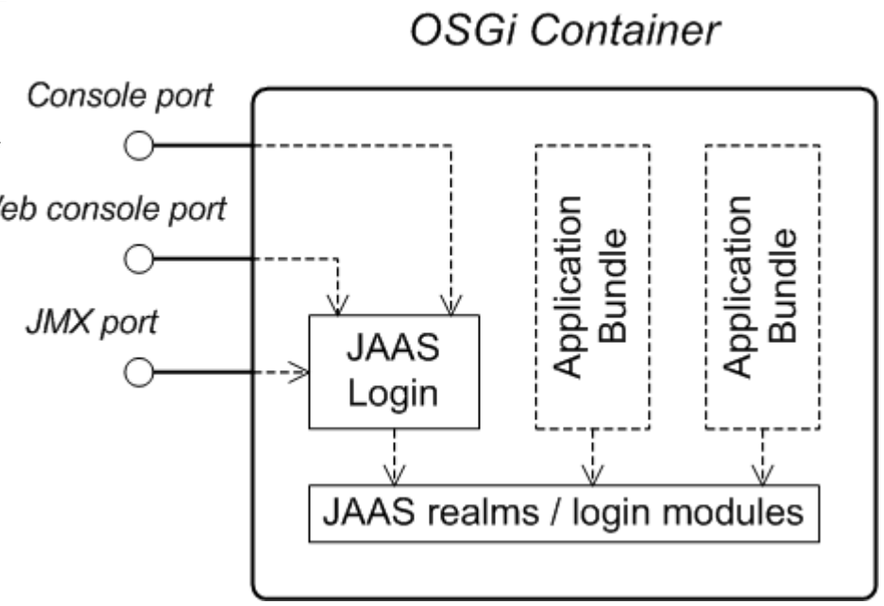
Console port - You can administer the OSGi container remotely either by connecting to the console port with a Karaf client or using the Karaf ssh:ssh command.

JMX port - You can manage the OSGi container by connecting to the JMX port. The JMX port is also secured by a JAAS login feature that connects to the karaf realm.

Application bundles and JAAS security - Any application bundles that you deploy into the OSGi container can access the container's JAAS realms.

https://access.redhat.com/site/documentation/en-US/JBoss_Fuse/6.0/html/Security_Guide/files/Arch-Architecture-Camel.html

.



Karaf Security Example

This tutorial below explains how to set up an X.500 directory server and configure the OSGi container to use LDAP authentication.

- install Apache Directory Server and Apache Directory Studio
- add user entries into the LDAP server
- add a group to manage security role
- configure Red Hat JBoss Fuse to use LDAP authentication
- configure JBoss Fuse to use roles for authorization
- configure an instance of Apache ActiveMQ to use LDAP authentication
- configure SSL/TLS connections to the LDAP server

https://access.redhat.com/site/documentation/en-US/JBoss_Fuse/6.0/html/Security_Guide/files/ESBLDAPTutorialOverview.html

Apache CXF Security

Securing CSF Services

- Secure Transports
- WS-* Security
- WS-Trust, STS
- SAML Web SSO
- Oauth
- Authentication with JAASLoginInterceptor and Kerberos
- Authorization
- Large Data Stream Caching

http://cxf.apache.org/docs/security.html

Apache CXF Example

An example of Apache CXF security is the same as the Camel Security example above

https://access.redhat.com/site/documentation/en-US/JBoss_Fuse/6.0/html/Security_Guide/files/CamelCXF.html

Picketlink

- IDM: Provide an object model for managing Identities (Users/Groups/Roles) and associated behavior using different identity store backends like LDAP and RDBMS.
- Federated Identity: Support SAMLv2, WS-Trust and OpenID.
- XACML: Oasis XACMLv2 implementation.
- Negotiation: Provide SPNego/Kerberos based Desktop SSO.

For latest information, please refer to http://www.picketlink.org

Picketlink - OASIS

OASIS (Organization for the Advancement of Structured Information Standards) is a non-profit consortium that drives the development, convergence and adoption of open standards for the global information society.

https://www.oasis-open.org/

Picketlink - SAML

Security Assertion Markup Language (SAML) is an XML-based open standard data format for exchanging authentication and authorization data between parties, in particular, between an identity provider and a service provider.

https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security

Picketlink - Fed

The following features are provided by the Fed component of PicketLink:

- Federated Authentication and SSO using Oasis SAML v2.0.
- Trusted Security System using a Security Token Server (STS) in an heterogeneous environment, using Oasis WS-Trust.
- Decentralized user driven Identity support via OpenID.

Picketlink - XACML

eXtensible Access Control Markup Language (XACML) is the standard that defines a declarative access control policy language implemented in XML and a processing model describing how to evaluate authorization requests according to the rules defined in policies.

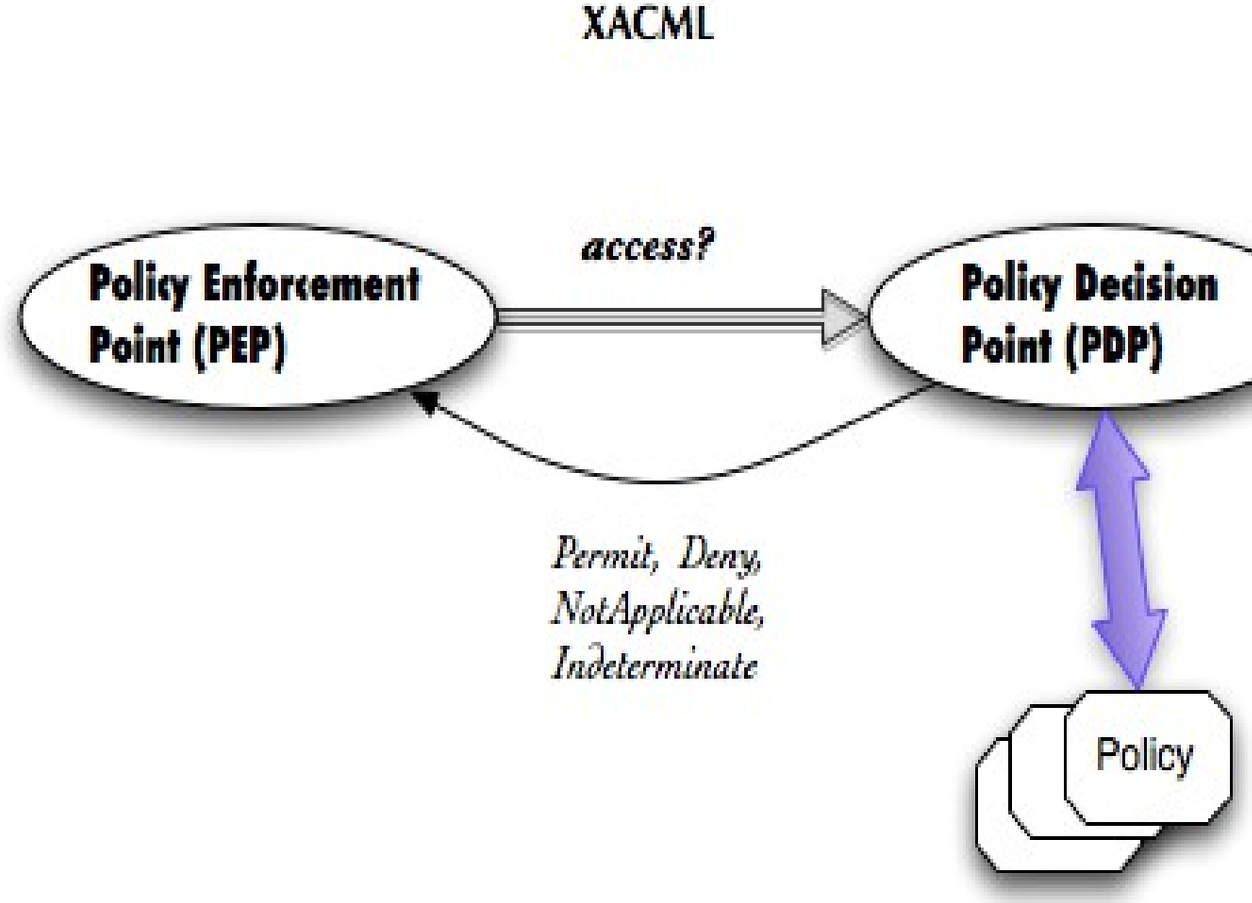
https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml

Picketlink - XACML

The following features are provided by the XACML component of PicketLink:

- Oasis XACML v2.0 library
- JAXB v2.0 based object model
- Multiple Database Integration for storing/retrieving XACML Policies and Attributes
- SAML v2.0 Profile of XACML v2.0
- RBAC Profile of XACML v2.0
- WSDL based SOAP PDP Service
- Servlet that accepts SOAP/SAML/XACML Payload

Picketlink - XACML



Access Control Best Practices

1. Know that you will need access control/authorization – spend time on it

2. Externalize the access control policy processing – do not hard code

3. Understand the difference between coarse grained and fine grained authorization – spend time during intitial design

4. Design for coarse grained authorization but keep the design flexible for fine grained authorization – loosely couple access control

5. Know the difference between Access Control Lists and Access Control standards – ACLS usually proprietary and not interoperable

6. Adopt Rule Based Access Control : view Access Control as Rules and Attributes – spend time on designing the rules and attributes

7. Adopt REST Style Architecture when your situation demands scale and thus REST authorization standards - Oauth or REST profile of XACML

8. Understand the difference between Enforcement versus Entitlement model – yes/no or what

http://anil-identity.blogspot.com/

Picketlink Security

Picketlink and Fuse Integration

- Camel - Route Security

  PicketLink provided Camel Route Policy Security Interceptor

- ActiveMQ

  PicketLink provided Security Classes

Spring Security

The camel-spring-security component provides role-based authorization for Camel routes. It leverages the authentication and user services provided by Spring Security and adds a declarative, role-based policy system to control whether a route can be executed by a given principal.

http://camel.apache.org/spring-security.html

Spring Security

The camel-spring-security component provides role-based authorization for Camel routes. It leverages the authentication and user services provided by Spring Security and adds a declarative, role-based policy system to control whether a route can be executed by a given principal.

http://camel.apache.org/spring-security.html

Shiro Security

Apache Shiro is a security framework that handles authentication, authorization, enterprise session management and cryptography. The camel shiro-security component allows authentication and authorization support to be applied to different segments of a camel route. Shiro security is applied on a route using a Camel Policy.

Security Examples on the security page:

- Applying Shiro Authentication on a Camel Route
- Applying Shiro Authorization on a Camel Route
- Creating a ShiroSecurityToken and injecting it into a Message Exchange
- Sending Messages to routes secured by a ShiroSecurityPolicy

http://camel.apache.org/shiro-security.html

Demonstrations

- Web Console Security
- ActiveMQ Broker, Producer, Consumer with SSL and JBDS

Questions