Authenticating REST services in OpenEdge

Controlling business logic access via OE Realm & LDAP

Paul Koufalis, White Star Software
pk@wss.com

Peter Judge, Progress Software
pjudge@progress.com
Abstract

- Slides-only précis of the REST Security Workshop presented earlier
- It covers securing REST services using
  - OERealm authentication
  - LDAP authentication
  - Roles authorization
  - Cross Origin Resource Sharing (CORS)
Who is this Paul Koufalis fellow?

- Progress DBA and UNIX sysadmin since 1994
- Expert consulting related to technical aspects of Progress and OpenEdge
- Wide range of experience
  - Small 10 person offices to 1500+ concurrent users
  - AIX, HPUX, Linux, Windows…if Progress runs on it, I’ve worked on it
And what about that Peter Judge person?

Peter Judge pjudge@progress.com
Software Architect
@ Progress since 2003
Integration-y stuff: HTTP-Out, Corticon
OE Best Practices / OERA / AutoEdge / CCS
4GL since 1996
Agenda

- Introduction to PAS/OE & REST
- What is a REST security architecture?
- What's LDAP?
- URL-based authorization
- CORS
- Considerations for a production environment
- Logging for fun & profit
REST-based ABL services
What is REST?

- **RE**presentational **S**tate **T**ransfer: an architecture for designing networked applications
- In English: Uses HTTP URLs and verbs to provide web services
  - GET [http://myweb.com/stuff/rest/customers/24](http://myweb.com/stuff/rest/customers/24) will return customer #24
  - GET [http://myweb.com/stuff/rest/customers](http://myweb.com/stuff/rest/customers) will return all customers
  - PUT [http://myweb.com/stuff/rest/customers/24](http://myweb.com/stuff/rest/customers/24) will update customer #24

- Some features:
  - **Client-Server**: pull- or request-based
  - **Stateless**: each request must contain all information necessary to understand the request
  - **Cacheable**
  - **Uniform interface**: all resources accessed via generic interface
What is REST?

- **Practically**, JSON payloads to and fro over HTTP, using a particular style of URL
- JSON content types
  - application/json
  - application/vnd.[vendor]+json

- Example request

```plaintext
GET /users/PeterJudge-PSC HTTP/1.1
Host: api.github.com
Accept: application/vnd.github.v3+json
User-Agent: OpenEdge-HttpClient/0.3.0
```
REST – Example Response

HTTP/1.1 200 OK
Cache-Control: public, max-age=60, s-maxage=60
Content-Encoding: gzip
Content-Type: application/json; charset=utf-8

Server: GitHub.com
{
  "login": "PeterJudge-PSC",
  "id": 2736095,
  "url": "https://api.github.com/users/PeterJudge-PSC",
  "html_url": "https://github.com/PeterJudge-PSC",
  "repos_url":
      "https://api.github.com/users/PeterJudge-PSC/repos",
  "type": "User",
  "hireable": true,
  "created_at": "2012-11-06T15:35:09Z",
  "updated_at": "2015-03-04T20:44:24Z"
}
OpenEdge REST Services

- Each deployed OE REST web application has
  - A web application name
  - One or more REST services
  - A security configuration (user authentication and [URL] authorization)
  - A connection to an AppServer [ABL service]

- Example OpenEdge REST web application:
  
  http://host:port/webAppName/rest/

  deployment site defined part  OE defined part
RESTful Service and URL Design

- A REST service
  - Has a service-name that appears in the URL
  - A service-name contains one or more REST resources
  - Each resource has a unique URL path within the service
  - Each resource URL path can have optional input parameters and/or options

- Example

  http://host:port/webAppName/rest/<service-name>/<resource-path>

  developer designed part
You Choose What Goes Where

- Routing
  - HTTP method + URI => class + method
  - Alt. => persistent proc + internal proc

- Mapping HTTP message into ABL arguments
  - URL parameters
  - Headers
  - Cookies
  - Server Context
Securing Your OE REST Web Service

- Security provided in layers
  - **SSL/TLS** to encrypt data-in-transit between client and web server and between web server and AppServer
  - Web application (Tomcat) **session management**
  - Spring Security for **authentication**
    - Can also provide **authorization**
  - OpenEdge AppServer for application level **authorization**
Authentication and Authorization

- **Authentication**
  - Identifies a user, using factors
    - Something the user knows (e.g. password)
    - Something the user has (e.g. security token)
    - Something of the user (e.g. biometric)
  - Verify that a user is who they say they are
    - We need to be able to trust this fact, as do others

- **Authorization**
  - What services can the user access?
  - What data can the user see and/or modify?
    - Multi-tenancy
    - Record-level, field-level
Spring Security

- OpenEdge supplements the Java container’s security with the industry-recognized Spring security framework
- Spring Security is a customizable authentication and access control framework
  http://projects.spring.io/spring-security/
- It is one of the industry standards for securing Spring-based applications
Web Application Authentication Models

- **Anonymous**: No user authentication or login session
- **HTTP Basic Authentication** — Client sends base64 encoded user name/password to web application in each http request
  - HTTP header: Authorization
- **HTTP Form Authentication** — Client logs in and out the web application once per session
  - **Login**: The client obtains user credentials and POSTs them to the web application
    - URI: /static/auth/j_spring_security_check
    - Body: j_username=xxxx&j_password=yyyy&submit=Submit+Query
    - Cookie: JSESSIONID
  - **Logout**: The client uses a GET request to log out
    - URI: /static/auth/j_spring_security_logout
Common Authentication Providers

- In-memory — user accounts in configuration file
- Local file — user accounts in clear-text file
- Container — user account authenticated by Java container
- LDAP / AD — user accounts in a Directory Service (11.2.1+)
- OERealm — OpenEdge AppServer service (11.3+)
Configuring Authentication Providers in OE REST: web.xml

```xml
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>
  <!-- USER EDIT: Select which application security model to employ
  /WEB-INF/oeablSecurity-basic-local.xml
  /WEB-INF/oeablSecurity-anonymous.xml
  /WEB-INF/oeablSecurity-form-local.xml
  /WEB-INF/oeablSecurity-container.xml
  /WEB-INF/oeablSecurity-basic-ldap.xml
  /WEB-INF/oeablSecurity-form-ldap.xml
  /WEB-INF/oeablSecurity-basic-oerealm.xml
  /WEB-INF/oeablSecurity-form-oerealm.xml
  /WEB-INF/oeablSecurity-form-saml.xml
  /WEB-INF/oeablSecurity-basic-saml.xml

  -->
  /WEB-INF/oeablSecurity-anonymous.xml
  </param-value>
</context-param>
```
What is PASOE?
Application Server Platform

- A single delivery platform for all Progress Web-based products
- Not only the application but also the web server to support it
  - Created from Apache Tomcat 7.0.55 distribution
- Designed for secure operation
  - Spring Security Framework included
  - Realms and roles defined to implement access control
High-level features

- Secure
- Simple
  - Administration, scalability, application migration, deployment
  - AppServer connection and operating STATES
- Customer Extensible
  - Open REST APIs for customer developed metrics, monitoring, and administration
  - Installation tailoring
- Better analysis tools
  - Built-in metrics gathering, current state queries
- Faster and optimizes resources
  - Runs same ABL application and client load with less memory and CPU consumption
PASOE Architectural Concepts and Terms

AppServer Web Stack

- **Tomcat**
  - AIA
  - SOAP (WSA)
  - Mobile [REST]

Unified broker

Agent
- 1 ABL Session

PASOE Web Stack

- **PASOE**
  - PAS extensions
  - OEPAS extensions

- **APSV (AIA)**
  - SOAP (WSA)
  - Mobile [REST]

Session Manager

- 1 named AppServer
- 1 named PASO Instance

MSAgent
- 1 ABL Session

1 oeabl web application

1 named PASO Instance
Architecture: Components

Classic AppServer Components:
- AdminServer
- Client
- NameServer
- AppServer

State-Aware
- 50 Agents

Stateless
- 50 Agents

State-Free
- 50 Agents

PASOE Components:
- Session Manager
- PASOE
- MSAgent

- APSV (AIA)
- SOAP (WSA)
- REST/Mobile

50 ABL Sessions
What is a REST security architecture?
Components of a security architecture

- Security tokens & Security Token Service
- Authentication flows
- Configuring Spring Security
- Flow of a CLIENT-PRINCIPAL through an application
What is a Security Token?

- A transportable block of data that can be used as proof of user identity by any systems or applications that have a trust relationship with the originator of the security token
  - Exists for same reason passports do: so that a gatekeeper doesn’t have to ask you for everything every time you want to pass
- Enables Single Sign On (SSO)
  - Authenticate once and allow access many times across (ABL) processes
- Secure, time sensitive and data-integrity protected

- CLIENT-PRINCIPAL = ABL security token
- Sets current identity in any connected db or AVM session

- A security token service (STS) is the service component that builds, signs, and issues security tokens [1]

Authentication flows between client, STS and AppServer

Client

PAS/OE

Security Token Service (Spring Security)
LoginUser() LogoutUser() ValidateUser

Domain Services (ABL)
  security-policy:set-client()
  CustomerSvc:ReadAllCustomers()

Single Point AuthN
  ValidateUser()
  ValidatePassword()
  GetAttributes()
Authentication flows between client, STS and AppServer

Client
GET /customers

PAS/OE

Security Token Service (Spring Security)
- LoginUser()
- LogoutUser()
- ValidateUser

Domain Services (ABL)
- security-policy:set-client()
- CustomerSvc:ReadAllCustomers()

Single Point AuthN
- ValidateUser()
- ValidatePassword()
- GetAttributes()
Authentication flows between client, STS and AppServer

Client
GET /customers
Authorization: mary/letmein

PAS/OE

Security Token Service (Spring Security)
LoginUser() LogoutUser() ValidateUser

Domain Services (ABL)
security-policy:set-client()
CustomerSvc:ReadAllCustomers()

Single Point AuthN
ValidateUser()
ValidatePassword()
GetAttributes()
Authentication flows between client, STS and AppServer

Client
- GET /customers
- Authorization: mary/letmein

PAS/OE

Security Token Service
- LoginUser()
- LogoutUser()
- ValidateUser

Domain Services
- security-policy:set-client()
- CustomerSvc:ReadAllCustomers()

Single Point AuthN
- ValidateUser()
- ValidatePassword()
- GetAttributes()
Authentication flows between client, STS and AppServer

Client
GET /customers
Authorization: mary/letmein

PAS/OE

Security Token Service
LoginUser() LogoutUser() ValidateUser()

Domain Services (ABL)
security-policy:set-client()
CustomerSvc:ReadAllCustomers()

Single Point AuthN
ValidateUser()
ValidatePassword()
GetAttributes()
Authentication flows between client, STS and AppServer

Client
GET /customers
Authorization: mary/letmein

PAS/OE

Security Token Service
LoginUser() LogoutUser() ValidateUser

Domain Services
security-policy:set-client()
CustomerSvc:ReadAllCustomers()

Single Point AuthN
ValidateUser()
ValidatePassword()
GetAttributes()
Flow of CLIENT-PRINCIPAL through an application

- STS passes credentials into AVM using CURRENT-REQUEST-INFO
  - CLIENT-PRINCIPAL available in activation event procedure
- Credentials can be asserted on session or per-database
  - SECURITY-POLICY:SET-CLIENT(<client-principal>)
  - SET-DB-CLIENT(<client-principal>, <database>)
- Token is validated against an existing domain
  - Domain name & access code must match for successful authentication
What Are Domains?

- A group of users with a common set of:
  - Roles and responsibilities
  - Level of security
  - Data access privileges

- Configured in db meta-schema:
  - Authentication systems
  - Tenants

```
_sec-authentication-domain

_Domain-name
_Domain-type
_Domain-description
_Domain-access-code
_Domain-runtime-options
_Tenant-name
_Domain-enabled
```
Authentication using an OpenEdge Realm
OE Realm details

- Class-based implementation of the Progress.Security.Realm.IHybridRealm interface
- SPA does **NOT** create, seal or otherwise operate on CLIENT-PRINCIPALs
- Requires integer type user id
  - ValidateUser(input <character>) returns <integer>
- Roles are Just Another Attribute

**Single Point of Authentication**
- ValidateUser()
- ValidatePassword()
- GetAttributes()
ValidateUser() Example

METHOD PUBLIC INTEGER ValidateUser( INPUT p0 AS CHARACTER ):  
DEFINE VARIABLE cName AS CHARACTER NO-UNDO.  
DEFINE VARIABLE cDomain AS CHARACTER NO-UNDO.

cName = ENTRY(1, p0, '@').  
IF NUM-ENTRIES(p0, '@') GE 2 THEN  
cDomain = ENTRY(2, p0, '@').

FIND ApplicationUser WHERE  
    ApplicationUser.LoginName = cName AND  
    ApplicationUser.LoginDomain = cDomain  
NO-LOCK NO-ERROR.

IF AVAILABLE ApplicationUser THEN  
    RETURN ApplicationUser.Id.

RETURN ?.  
END METHOD.
GetAttribute() Example

METHOD PUBLIC CHARACTER GetAttribute( INPUT p0 AS INTEGER, INPUT p1 AS CHARACTER ): 
DEFINE VARIABLE cAttribute AS CHARACTER NO-UNDO.

CASE p1:
  WHEN "ATTR_ROLES" THEN
    DO:
      FIND ApplicationUser WHERE
      ApplicationUser.Id EQ p0
      NO-LOCK NO-ERROR.
      FOR EACH GrantedRole WHERE
        GrantedRole.Grantee EQ LC(ApplicationUser.LoginName) + '@' + LC(ApplicationUser.LoginDomain)
        NO-LOCK:
          cAttribute = cAttribute + ',,' + GrantedRole.RoleName.
      END.
    END.
  END CASE.

RETURN cAttribute.
END METHOD.
LDAP Authentication
LDAP Essentials

- Single manageable location for enterprise configurations
- LDAP (Lightweight Directory Access Protocol) is a wire protocol and client API
- LDAP Directory Services widely used for single point of administration
  - Most commonly recognized as a Single Point of Authentication (SPA)
- A directory service is hierarchical store of schema defined objects and object attributes
  - Examples: OpenLDAP, Windows Active Directory, Apache DS
  - No two production sites will have the same hierarchy (of users & groups)
Key Directory Service Terms and Concepts

- Distinguished Name (DN)
  - The path to a specific data object
  - Root DN: the name of the object hierarchy's root data object
    example: dc=acme,dc=com
  - Fully qualified DN: full path to the object from the root DN to the object
    example: dn=ldapserver1,ou=IT,dc=acme,dc=com
  - Relative DN:
    example: dn=ldapserver (child object of: ou=IT,dc=acme,dc=com)

- Search root: the fully qualified DN of the data object at which to begin a descending search for one or more data objects
Key Directory Service Terms and Concepts (cont'd)

- Directory Services require logging in to search information
- Security policies prevent read/write of another user’s password attribute
- Passwords are stored as salted one-way hashes

To test a user account’s password for login

1. You have to login with a fully qualified DN that has search privileges
2. Search to find the user’s account and retrieve its fully qualified DN
3. Logout
4. Login using the user account’s fully qualified DN and password
5. Retrieve user attributes - primarily the Groups (i.e. Role) they are a member of
6. Logout
<table>
<thead>
<tr>
<th>Required Information From Directory Service Admin</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Network address and ports of the Directory Service</td>
<td>&quot;foo.com&quot; 389</td>
</tr>
<tr>
<td>2. ROOT DN of the directory service</td>
<td>&quot;dc=foo, dc=com&quot;</td>
</tr>
<tr>
<td>3. DN &amp; password of an account with ‘query’ privilege</td>
<td>&quot;uid=admin, ou=ds admins, ou=IT, dc=foo, dc=com&quot;</td>
</tr>
<tr>
<td>4. LDAP DN of the object where the user object search will start</td>
<td>&quot;ou=users, ou=employees, dc=foo, dc=com&quot;</td>
</tr>
<tr>
<td>5. LDAP user account object’s attribute name that holds the user’s login ID</td>
<td>&quot;uid&quot;</td>
</tr>
<tr>
<td>6. LDAP DN of the object where the search for LDAP user groups (roles) will start</td>
<td>&quot;ou=groups,dc=foo,dc=com&quot;</td>
</tr>
<tr>
<td>7. LDAP group object’s attribute name whose value will be the role name inserted into the user’s login token</td>
<td>&quot;uniqueMember&quot; or &quot;roleOccupant&quot;</td>
</tr>
<tr>
<td>8. LDAP Group attribute holding the Role/Group name</td>
<td>&quot;cn&quot;</td>
</tr>
</tbody>
</table>
Example directory
Configuring Spring Security LDAP: Directory access

```
<ldap-server id="PrimayLDAP"
url="ldap://localhost:10389/dc=XXX,dc=YYY"
manager-dn="uid=admin,ou=system"
manager-password="secret" />
```
Configuring Spring Security LDAP: Directory access

```xml
<ldap-server id="PrimayLDAP" url="ldap://localhost:10389 /dc=XXX,dc=YYY"
  manager-dn="uid=admin,ou=system"
  manager-password="secret" />
```
Configuring Spring Security LDAP: Authentication Manager

```xml
<authentication-manager id="oeablApplicationAuth">
  <ldap-authentication-provider
    server-ref="PrimayLDAP"
    role-prefix="ROLE_"
    group-search-base="ou=groups,ou=sports,ou=tenants,dc=pugchallenge,dc=org"
    group-role-attribute="cn"
    group-search-filter="(roleOccupant={0})"
    user-search-base="ou=users,ou=sports,ou=tenants,dc=pugchallenge,dc=org"
    user-search-filter="(uid={0})"
    />
</authentication-manager>
```
Configuring Spring Security LDAP: Authentication Manager

```
<authentication-manager id="oeablApplicationAuth">
    <ldap-authentication-provider
        server-ref="PrimaryLDAP"
        role-prefix="ROLE_"
        group-search-base=
            "ou=groups,ou=sports,ou=tenants,dc=pugchallenge,dc=org"
        group-role-attribute="cn"
        group-search-filter="(roleOccupant={0})"
        user-search-base=
            "ou=users,ou=sports,ou=tenants,dc=pugchallenge,dc=org"
        user-search-filter="(uid={0})"
    />
</authentication-manager>
```
Configuring Spring Security LDAP: Authentication Manager

```xml
<authentication-manager id="oeablApplicationAuth">
    <ldap-authentication-provider
        server-ref="PrimayLDAP"
        role-prefix="ROLE_"
        group-search-base="ou=groups,ou=sports,ou=tenants,dc=pugchallenge,dc=org"
        group-role-attribute="cn"
        group-search-filter="(roleOccupant={0})"
        user-search-base="ou=users,ou=sports,ou=tenants,dc=pugchallenge,dc=org"
        user-search-filter="(uid={0})"
    />
</authentication-manager>
```
Configuring Spring Security LDAP: Authentication Manager

```xml
<authentication-manager id="oeablApplicationAuth">
  <ldap-authentication-provider
    server-ref="PrimaryLDAP"
    role-prefix="ROLE_"
    group-search-base="ou=groups,ou=sports,ou=tenants,dc=pugchallenge,dc=org"
    group-role-attribute="cn"
    group-search-filter="(roleOccupant={0})"
    user-search-base="ou=users,ou=sports,ou=tenants,dc=pugchallenge,dc=org"
    user-search-filter="(uid={0})"
  />
</authentication-manager>
```
Configuring Spring Security LDAP: Authentication Manager

```xml
<authentication-manager id="oeablApplicationAuth">
  <ldap-authentication-provider
    server-ref="PrimayLDAP"
    role-prefix="ROLE_"
    group-search-base="ou=groups,ou=sports,ou=tenants,dc=pugchallenge,dc=org"
    group-role-attribute="cn"
    group-search-filter="(roleOccupant={0})"
    user-search-base="ou=users,ou=sports,ou=tenants,dc=pugchallenge,dc=org"
    user-search-filter="(uid={0})"
  />
</authentication-manager>
```
<authentication-manager id="oeablApplicationAuth">
    <ldap-authentication-provider
        server-ref="PrimayLDAP"
        role-prefix="ROLE_"
        group-search-base="ou=groups,ou=sports,ou=tenants,dc=pugchallenge,dc=org"
        group-role-attribute="cn"
        group-search-filter="(roleOccupant={0})"
        user-search-base="ou=users,ou=sports,ou=tenants,dc=pugchallenge,dc=org"
        user-search-filter="(uid={0})"
    />
</authentication-manager>
Configuring Spring Security LDAP: Authentication Manager

```xml
<authentication-manager id="oeablApplicationAuth">
  <ldap-authentication-provider server-ref="PrimaryLDAP">
    <user-search-base="ou=users,ou=sports,ou=tenants,dc=pugchallenge,dc=org">
      <user-search-filter="(uid={0})"/>
    </user-search-base>
    <group-search-filter="(roleOccupant={0})"/>
    <group-search-base="ou=groups,ou=sports,ou=tenants,dc=pugchallenge,dc=org">
      <group-role-attribute-cn>cn</group-role-attribute-cn>
    </group-search-base>
    <group-search-ref="PrimayLDAP"/>
  </ldap-authentication-provider>
</authentication-manager>
```

cn=Customer.Read
cn=Customer.Update
cn=PSCUser
URL-based Authorization
Intercept URLs

- A means to provide **authorization** based on a URL + method (verb)
  - We use role-based authorization

- An intercept-url element (filter) contains three attributes
  - Pattern: A URI pattern relative to the web application (**restsec** in this case). The pattern can be as general or specific as required.
  - Method: The HTTP method to which this element applies (optional)
  - Access: The access (restriction) to apply when the pattern and method match the requested resource.

- Intercepts are processed in the order in which they appear in the configuration
Example intercept URL

- Specified in the oeablSecurity-*.xml security configuration file
- Add an intercept-url element under `<http pattern="/rest/**"` element
  - pattern – URL pattern to which this role authorization applies
    - ** in path indicates it applies recursively
    - pattern path is relative to the restsec webapp
    - Look for other examples in XML file
  - method - HTTP method; delete is what we want to restrict
  - access - Authorize using hasAnyRole('ROLE_CUSTOMER_DELETE')

```xml
<intercept-url
  pattern="/rest/restws/customers/**"
  method="DELETE"
  access="hasAnyRole('ROLE_CUSTOMER.DELETE')" />
```
Cross-Origin Resource Sharing (CORS)
Cross Origin Resource Sharing (CORS)

- A simplified example may be easiest to understand
  2. This page lists all customers by GETting /rest/restws/customers (using XMLHttpRequest)
  3. If it GETs http://www.paul.com/restsec/rest/restws/customers this is a SAME ORIGIN request
  4. If it GETs /rest/restws/customers from anywhere else this is a CROSS ORIGIN request

- Browsers generally dislike this for security reasons

  **CORS is a mechanism that allows a web page loaded from one domain to make HTTP requests to access resources located in another domain**

- PAS provides CORS functionality using the Open Source “CORS Filter”
CORS vs. NON-CORS

- CORS-enabled web servers classify all requests as
  1. CORS request containing an “Origin” header
  2. CORS preflight request containing “Access-Control-Request-Method” header in an OPTION request
  3. A generic (non-CORS) request that does not contain any CORS headers

- How the web server reacts to each of these requests is configured in the “OECorsFilter” in the oeablSecurity-*.xml file
Example: OpenEdge CORS Filter

<!-- The security filter that implements the CORS standard for controlling cross site resource access by http clients. -->

<b:bean id="OECORSFilter"
   class="com.progress.appserv.services.security.OECORSFilter" >
  <b:property name="allowAll" value="true" />
  <b:property name="allowDomains"
     value="http://52.91.193.122:8810" />
  <b:property name="allowSubdomains" value="false" />
  <b:property name="allowMethods" value="" />
  <b:property name="messageHeaders" value="" />
  <b:property name="responseHeaders" value="" />
  <b:property name="supportCredentials" value="true" />
  <b:property name="maxAge" value="-1" />
</b:bean>
Deploying in Production
Considerations for a production environment

- Authentication defaults to ANONYMOUS in PROD and DEV modes
  - Our psychics were unable to determine your security mode when OE goes to FCS

  **PLEASE DON'T LEAVE IT LIKE THIS**

- Limited user and file permissions on UNIX installs
- Deployment tooling in PDSOE for DEV, not PROD
- May need changes to security XML files
- No r-code in PROD server
  - Use traditional deployment means
  - R-code in WAR via PDSOE in 11.6
- Need to explicitly deploy oe-/manager to PROD
Q&A

pjudge@progress.com or pk@wss.com
The oldest and most respected independent DBA consulting firm in the world

Four of the world’s top OpenEdge DBAs

Author of ProTop, the #1 FREE OpenEdge Database Monitoring Tool
  • http://dbappraise.com/protop.html