RAD Server

Autorización y Autenticación

Fernando Rizzato
Lead Software Consultant, *Latin America*

AGENDA

- Procesos de autenticación
- Autenticación a nivel de aplicación
- Autenticación a nivel de usuario
- Autenticación versus Autorización
- Flujo de autenticación y autorización
- Configuración de autorización
- Demos

PROCESOS DE AUTENTICACIÓN

- La autenticación es el proceso por el cual un servidor EMS requiere de sus aplicaciones cliente proporcionar algún identificador antes de permitir que ellas accedan a sus *endpoints*
- Hay dos niveles de autenticación disponibles, dependiendo de las necesidades de su servicio
  - Autenticación a nivel de aplicación
  - Autenticación a nivel de usuario

AUTENTICACIÓN A NIVEL DE APLICACIÓN

- La autenticación a nivel de aplicación requiere que cualquier cliente proporcione información antes de que puedan solicitar correctamente un *endpoint*
- EMS admite dos tipos de autenticación a nivel de aplicación
  - *AppSecret*
  - *MasterSecret*
- Se pueden definir en la sección [Server.Keys] del archivo de configuración de EMS

AUTENTICACIÓN A NIVEL DE APLICACIÓN

- Por defecto, **AppSecret** está vacía en la sección [Server.Keys]
- Cuando se establece **AppSecret**, el cliente debe proporcionar el valor de **AppSecret** en todas las solicitudes de recursos a EMS
- Los clientes pasan el valor de **AppSecret** en el encabezado de la solicitud a través del parámetro **X-Embarcadero-App-Secret**
- Cuando las restricciones adicionales a recursos se definen en la sección [Server.Authorization], tanto el **AppSecret** como las restricciones [Server.Authorization] se deben atender de forma simultánea

AUTENTICACIÓN A NIVEL DE APLICACIÓN

- Cuando se establece el **MasterSecret** y un cliente pasa el valor de **MasterSecret** en un encabezado de solicitud, los derechos al recurso solicitado se conceden incondicionalmente
- En concreto, cualquier restricción definida en la sección [Server.Authorization] del archivo de configuración de EMS se ignora cuando se proporciona un **MasterSecret** válido
- Los clientes pasan el valor de **MasterSecret** en el encabezado de la solicitud a través del parámetro **X-Embarcadero-Master-Secret**

AUTENTICACIÓN A NIVEL DE USUARIO

- La sección [Server.Keys] del archivo de configuración de EMS también contiene una clave llamada **ApplicationID**
- **ApplicationID** no se aplica específicamente a la autenticación pero se puede utilizar en instalaciones donde hay varios servidores EMS para evitar una incompatibilidad entre el cliente y el servidor
- Los clientes pasan el valor del **ApplicationID** en el encabezado de la solicitud a través del parámetro **X-Embarcadero-Application-Id**

AUTENTICACIÓN A NIVEL DE USUARIO

- La autenticación en el nivel de usuario requiere que un usuario inicie sesión formalmente en la aplicación, cuando se genera un token de sesión
- Este token de sesión, cuyo nombre es **X-Embarcadero-Session-Token**, se debe proporcionar en el encabezado de cada solicitud posterior para que el servidor EMS pueda identificar al usuario
- Este **token** de sesión es la pieza central de la autorización

AUTENTICACIÓN VERSUS AUTORIZACIÓN

- **Autenticación** es el proceso de identificación de un usuario
- Para autenticar a un usuario, un cliente primero debe llamar *Login* o *Signup* del recurso User de la API administrativa de EMS
- El inicio de sesión es una solicitud POST que transmite un objeto JSON en el cuerpo del mensaje HTTP. Este objeto debe tener dos propiedades JSON, "username" y "contraseña"
- Si los datos proporcionados corresponden a los de un usuario de EMS existente, un token de sesión se devuelve al cliente en la respuesta

AUTENTICACIÓN VERSUS AUTORIZACIÓN

- **Autorización** es la determinación de que un determinado usuario tiene derecho a invocar un punto final específico
- Esto se puede hacer a nivel de usuario o de grupo
- Una autorización en el nivel de usuario o de grupo se emplea para permitir o denegar el acceso a un *endpoint* o un recurso específico
- Usted controla la autorización y la autenticación de la sección [Server.Authorization] del archivo de configuración de EMS

FLUJO DE AUTENTICACIÓN Y AUTORIZACIÓN

1. El cliente realiza la solicitud de login

   POST http://da-build:8080/users/login HTTP/1.1
   {"username":"User1","password":"User1pass"}

2. El servidor responde

   HTTP/1.1 201 Created {"username":"User1","_id":"04C3B621-A056-49CF-8C56-D18E8363F58E","_meta":{"creator":"04C3B621- A056-49CF-8C56-D18E8363F58E","created":"2018-05-04T09:05:54.000Z"},"session Token":"d7bdc5523d04ecab7a35c1df53a7077d"}

3. El cliente llama un *endpoint*

   GET http://da-build:8080/country HTTP/1.1
   X-Embarcadero-Session-Token: d7bdc5523d04ecab7a35c1df53a7077d

4. El servidor responde

   HTTP/1.1 200 OK
   ”country test"

CONFIGURACIÓN DE AUTORIZACIÓN

```
Users={"groups" : ["everyone"]}
Users.LoginUser={"public": true}
Users.AddUser={"groups", ["admin"]}
Users.DeleteUser={"groups", ["admin"]}
Groups={"groups" : ["everyone"]}
Groups.AddGroup={"groups", ["admin"]}
Groups.DeleteGroup={"groups", ["admin"]}
Groups.UpdateGroup={"groups", ["admin"]}
Customers={"groups" : ["everyone", "admin"]}
Accounts={"groups" : ["*"]}
AccessControl={"groups", ["admin"]}
```

CONFIGURACIÓN DE AUTORIZACIÓN

```
procedure TTestResource1.CheckAdministrator(const AContext:
TEndpointContext);
begin
  //Allow MasterSecret unconditionally
  if not (TEndpointContext.TAuthenticate.MasterSecret in
AContext.Authenticated) then
  begin
    if AContext.User = nil then
      EEMSHTTPError.RaiseUnauthorized('', 'User required');
    if not AContext.User.Groups.Contains('Administrators') then
      EEMSHTTPError.RaiseForbidden('', 'Administrator required');
  end;
end;

procedure TTestResource1.Get(const AContext: TEndpointContext;
  const ARequest: TEndpointRequest; const AResponse: TEndpointResponse);
begin
  CheckAdministrator(AContext);
  // implement response here
  // …
end;
```

DEMOS

GRACIAS!

Preguntas?

Me puedes encontrar en:
@FernandoRizzato
fernando.rizzato@embarcadero.com

Síguenos en
fb.com/EMBTLatAm