embarcadero

DelphiAcademy

Dicas rápidas, truques e técnicas

RAD Server

Autorização e Autenticação

Fernando Rizzato
Lead Software Consultant, *Latin America*

AGENDA

- Processos de Autenticação
- Autenticação no nível do aplicativo
- Autenticação no nível do usuário
- Autenticação versus Autorização
- Fluxo de Autenticação e Autorização
- Configurações de Autorização
- Demos

PROCESSOS DE AUTENTICAÇÃO

- A autenticação é o processo pelo qual um servidor EMS exige de suas aplicações cliente fornecer alguma identificação antes de permitir que elas acessem seus *endpoints*
- Existem dois níveis de autenticação disponíveis, dependendo das necessidades do seu serviço
  - Autenticação no nível do aplicativo
  - Autenticação no nível do usuário

AUTENTICAÇÃO NO NÍVEL DO APLICATIVO

- A autenticação em nível de aplicativo requer que qualquer cliente forneça informações antes que elas possam solicitar com êxito um *endpoint*
- O EMS suporta dois tipos de autenticação em nível de aplicativo
  - *AppSecret*
  - *MasterSecret*
- Eles podem ser definidos na seção [Server.Keys] do arquivo de configuração do EMS

AUTENTICAÇÃO NO NÍVEL DO APLICATIVO

- Por padrão, o **AppSecret** está vazio na seção [Server.Keys]
- Quando **AppSecret** é definido, o cliente deve fornecer o valor do **AppSecret** em todas as solicitações de recursos ao EMS
- Os clientes passam o valor de **AppSecret** no cabeçalho da solicitação através do parâmetro **X-Embarcadero-App-Secret**
- Quando restrições adicionais a recursos são definidas na seção [Server.Authorization], tanto o **AppSecret** quanto as restrições [Server.Authorization] devem ser atendidas de maneira simultânea

AUTENTICAÇÃO NO NÍVEL DO APLICATIVO

- Quando o **MasterSecret** é definido e um cliente passa o valor de **MasterSecret** em um cabeçalho de solicitação, os direitos ao recurso solicitado são concedidos incondicionalmente
- Especificamente, quaisquer restrições definidas na seção [Server.Authorization] do arquivo de configuração do EMS serão ignoradas quando um **MasterSecret** válido é fornecido
- Os clientes passam o valor de **MasterSecret** no cabeçalho da solicitação através do parâmetro **X-Embarcadero-Master-Secret**

AUTENTICAÇÃO NO NÍVEL DO APLICATIVO

- A seção [Server.Keys] do arquivo de configuração do EMS também contém uma chave chamada **ApplicationID**
- O **ApplicationID** não se aplica especificamente à autenticação, mas pode ser empregado em instalações onde há vários servidores EMS para evitar uma incompatibilidade entre o cliente e o servidor
- Os clientes passam o valor do **ApplicationID** no cabeçalho da solicitação através do parâmetro **X-Embarcadero-Application-Id**

AUTENTICAÇÃO NO NÍVEL DO USUÁRIO

- A autenticação no nível de usuário requer que um usuário faça login formalmente no aplicativo, ocasião em que é gerado um token de sessão
- Esse token de sessão, cujo nome é **X-Embarcadero-Session-Token**, deve ser fornecido no cabeçalho de cada solicitação subsequente para que o servidor EMS possa identificar o usuário
- Este **token** de sessão é a peça central da autorização

AUTENTICAÇÃO VERSUS AUTORIZAÇÃO

- **Autenticação** é o processo de identificação de um usuário
- Para autenticar um usuário, um cliente deve primeiro chamar o o Login ou Signup do recurso User da API administrativa do EMS
- O login é uma solicitação POST que transmite um objeto JSON no corpo da mensagem HTTP. Esse objeto deve ter duas propriedades JSON, "username" e "password”
- Se os dados fornecidos corresponderem aos de um usuário do EMS existente, um token de sessão será retornado ao cliente na resposta

AUTENTICAÇÃO VERSUS AUTORIZAÇÃO

- **Autorização** é a determinação de que determinado usuário tem o direito de invocar um *endpoint* específico
- Isso pode ser feito no nível de usuário ou de grupo
- Uma autorização no nível de usuário ou de grupo é empregada para permitir ou negar acesso a um endpoint ou recurso específico
- Você controla a autorização e a autenticação a partir da seção [Server.Authorization] do arquivo de configuração do EMS

FLUXO DE AUTENTICAÇÃO E AUTORIZAÇÃO

1. O cliente faz a solicitação de login

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

2. O 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. O cliente chama um endpoint

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

4. O servidor responde

   HTTP/1.1 200 OK
   ”country test"

CONFIGURAÇÕES DE AUTORIZAÇÃO

```
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"]}
```

CONFIGURAÇÕES DE AUTORIZAÇÃO

```
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

OBRIGADO!

Perguntas?

Você pode me encontrar em:
@FernandoRizzato
fernando.rizzato@embarcadero.com

Siga-nos em
fb.com/DelphiBrasil
fb.com/EmbarcaderoBR