RFC000032 - Autenticação Aplicações On-Premises

Resumo

Definir uma solução que permita a autenticação das aplicações TOTVS On-Premises junto ao TOTVS Identity bem como uma solução de autenticação de contingência para quando este último estiver indisponível.

Contexto e necessidade

O projeto de modernização dos ERPs, também conhecido como projeto Júpiter, tem como uma de suas premissas principais a integração das aplicações On-Premises às aplicações TOTVS Apps.

Para que isso seja possível os processos de autenticação destas aplicações devem estar integrados, compartilhando, dentro do possível, as mesmas credenciais de acesso, evitando que o usuário tem que realizar múltiplos procedimentos de autenticação, um para cada aplicação que ele deseje acessar.

Contúdo, e compreendendo que nem sempre o serviço de autenticação (TOTVS Identity) pode estar disponível, como por exemplo na ausência da infraestrutura de comunicação com a Internet, precisamos oferecer um processo de autenticação de contingência para que as aplicações On-Premises possam ser utilizadas neste cenário.

Norteadores da proposta

Visando atender as necessidades acima mencionadas, devemos fornecer:

- Uma definição de qual protocolo/solução deve ser utilizada no processo de autenticação junto ao TOTVS Identity.
- Uma definição de qual protocolo/solução deve ser utilizada no processo de autenticação de contingência.

Opções consideradas

No que diz respeito aos protocolos e soluções de autenticação, foram analisados:

- O fluxo ROPC do protoloco de autenticação OAuth2 utilizando ROPC (fluxo de password), que já foi depreciado devido a várias falhas de segurança, como explicadas em mais detalhes neste artigo (Don't use the OAuth password grant type).
- O protocolo de autenticação OIDC (OpenID Connect), que tem como premissa o protocolo OAuth2, porém utilizando de authorization code e client credentials, e já é um protocolo suportado por algumas aplicações da TOTVS.
- O protocolo de autenticação TOTP (Time-based One-Time Password), comumente utilizado como MFA (Múltiplo Fator de Autenticação).

Proposta

Utilizar o protocolo OIDC como fluxo principal de autenticação, e o TOTP como fluxo alternativo de autenticação quanto (e somente quando) o TOTVS Identity não puder ser acessado (contingência).

Papéis envolvidos no fluxo de autenticação

- RESOURCE OWNER = USUÁRIO
- CLIENT = FRONTEND DA APLICAÇÃO
- RESOURCE SERVER = BACKEND DA APLICAÇÃO
- AUTHORIZATION SERVER = ACCOUNTS
- IDP = IDENTITY

Fluxo de Autenticação OIDC

1. O USUÁRIO, no NAVEGADOR, acessa o endereço do FRONTEND
2. O FRONTEND tenta acessar uma API SEGURA do BACKEND
3. O BACKEND verifica que a requisição não tem um TOKEN DE APLICAÇÃO e retorna um 401 (NÃO AUTORIZADO)
4. O FRONTEND verifica que houve um retorno 401, que não existe um REFRESH TOKEN e o TOTVS Identity está disponível
5. O FRONTEND gera um STATE e chama a API de AUTHORIZE do ACCOUNTS passando:

- grant_type = authorization_code
- state = < STATE gerado pelo FRONTEND >
- response_type = code
- scope = < id da APLICAÇÃO >
- client_id = < id da credencial daquela instância da APLICAÇÃO >
- redirect_uri = < url da APLICAÇÃO que fará a troca do token >

6. O API de AUTHORIZE do ACCOUNTS valida se a REDIRECT_URI é válida para o CLIENT_ID informado
7. O API de AUTHORIZE do ACCOUNTS gera um CODE e retorna um redirect para a tela de LOGIN do IDENTITY
8. O IDENTITY apresenta a tela de LOGIN para o USUÁRIO
9. O USUÁRIO informa suas CREDENCIAIS:

- e-mail
- password

10. O IDENTITY valida as CREDENCIAIS e informa o ACCOUNTS que o CODE acima esta autorizado
11. O ACCOUNTS retorna um redirect para a REDIRECT_URI passando:

- code = < CODE >
- state = < UUID gerado pelo FRONTEND no início do fluxo >

12. O FRONTEND valida o STATE recebido e chama a API de TOKEN do BACKEND passando o CODE recebido

```
13. A API de TOKEN do BACKEND chama a API de TOKEN do ACCOUNT passando:
  - grant_type = authorization_code
  - code = < CODE autorizado >
  - header authorization = < basic (base64) da credencial daquela instância
da APLICAÇÃO >
14. A API de TOKEN do ACCOUNTS retorna:
```

```
  - access_token = < TOKEN opaco de acesso >
  - token_type = Bearer
  - refresh_token = < TOKEN opaco para refresh >
  - expires_in = < TIMEOUT em segundos = 1200 >
  - id_token = < JWT com informações do USUÁRIO >
15. A API de TOKEN do BACKEND gera e retorna para o FRONTEND:
  - token da aplicação
  - refresh token
```

16. O FRONTEND recebe o TOKEN DE APLICAÇÃO e passa a injetar (HEADER ou COOKIE) em todas as requisições para o BACKEND
17. O FRONTEND tenta acessar novamente a API do BACKEND, agora com sucesso

Fluxo de Refresh OIDC

1. O FRONTEND tenta acessar uma API SEGURA do BACKEND
2. O BACKEND verifica que a requisição tem um TOKEN DE APLICAÇÃO mas o mesmo está EXPIRADO e retorna um 401 (NÃO AUTORIZADO)
3. O FRONTEND verifica que houve um retorno 401, que existe um REFRESH TOKEN (que é um COOKIE)
4. O FRONTEND chama a API de TOKEN da APLICAÇÃO informando:

- grant_type = refresh_token

```
5. A API de TOKEN do BACKEND chama a API de TOKEN do ACCOUNTS passando:
  - grant_type = refresh_token
  - refresh_token = < REFRESH_TOKEN >
  - header authorization = < basic da credencial daquela instância da
APLICAÇÃO >
6. A API de TOKEN do BACKEND retorna:
  - token da aplicação
  - refresh token
```

Fluxo de Autenticação TOTP

1. O USUÁRIO, no NAVEGADOR, acessa o endereço do FRONTEND
2. O FRONTEND tenta acessar uma API SEGURA do BACKEND
3. O BACKEND verifica que a requisição não tem um TOKEN DE APLICAÇÃO e retorna um 401 (NÃO AUTORIZADO)
4. O FRONTEND verifica que houve um retorno 401, que não existe um REFRESH TOKEN e que o TOTVS IDENTITY não está disponível
5. O FRONTEND chama a tela de LOGIN TOTP da APLICAÇÃO

```
6. O BACKEND apresenta da tela de LOGIN TOTP
```

7. O USUÁRIO informa suas CREDENCIAIS:

- e-mail
- token temporário gerado por um gerador de tokens

```
8. A API de LOGIN TOTP do BACKEND valida as CREDENCIAIS chamando o
VALIDADOR TOTP
9. A API de LOGIN TOTP do BACKEND retorna um redirect para API de TOKEN do
BACKEND
10. A API de TOKEN do BACKEND gera e retorna para o FRONTEND:
  - token da aplicação
  - refresh token
```

11. O FRONTEND recebe o TOKEN DE APLICAÇÃO e passa a injetar (HEADER ou COOKIE) em todas as requisições para o BACKEND
12. O FRONTEND tenta acessar novamente a API do BACKEND, agora com sucesso

** Os endpoints acima de TOTP do BACKEND só devem estar acessíveis em tempo de CONTINGÊNCIA, para evitar que esse fluxo sejam utilizados indevidamente.

Fluxo de Refresh TOTP

1. O FRONTEND tenta acessar uma API SEGURA do BACKEND
2. O BACKEND verifica que a requisição tem um TOKEN DE APLICAÇÃO mas o mesmo está EXPIRADO e retorna um 401 (NÃO AUTORIZADO)
3. O FRONTEND verifica que houve um retorno 401, que existe um REFRESH TOKEN (que é um COOKIE)
4. O FRONTEND chama a API de TOKEN do BACKEND informando:

- grant_type = refresh_token

```
5. A API de TOKEN do BACKEND retorna:
  - token da aplicação
  - refresh token
```

Neste cenário a aplicação fica reponsável por gerar (e validar) também o REFRESH TOKEN.

Consequências

Consequências da proposta acima:

API de TOKEN

Implementação de uma API (endpoint HTTP) para a geração de tokens.
Esta API deve suportar tanto o fluxo OIDC quanto o fluxo TOTP, quando não há interação com o ACCOUNTS, seja para a geração, seja para o refresh do mesmo.

Configuração TOTP

Implementação de uma funcionalidade que permita ao usuário configurar o TOTP em gerador de tokens.
O usuário pode acessar essa funcionalidade a qualquer tempo desde que já esteja logado na aplicação.
O usuário pode utilizar qualquer gerador de tokens disponíveis nas plataformas Android ou IOS, ou até mesmo como extensões que são encontradas em alguns navegadores.
O administrador pode acessar essa funcionalidade em nome de outro usuário, caso o mesmo ainda não tenha realizado essa configuração.
O administrador deve obrigatoriamente configurar o seu TOTP, visto que ele mesmo não teria acesso em tempo de contingência.
A aplicação pode optar por utilizar alguma biblioteca de mercado para a geração do QRCODE ou então a ferramenta TOTVS TOTP Server, detalhada mais abaixo.

Login TOTP

Implementação da tela de LOGIN TOTP da APLICAÇÃO, baseado num template HTML funcional que será disponibilizada num repositório posteriormente.

API de LOGIN TOTP

Implementação de uma API (endpoint HTTP) para a validação das credenciais (email e token TOTP) informadas pelo usuário.
A aplicação pode optar por utilizar alguma biblioteca de mercado para essa validação ou então a ferramenta TOTVS TOTP Server, detalhada mais abaixo.

Single Sign-On (SSO)

O Single Sign-On entre as aplicações OnPremises e as aplicações TOTVS APPs, onde o usuário não precisa reapresentar suas credenciais de acesso (Login), é possível desde que o processo de autenticação OIDC seja realizado numa mesma sessão do navegador, para que os COOKIES de autenticação (do TOTVS Identity) podem ser compartilhados entre as execuções dos fluxos de autenticação.

Chamadas às APIs dos TOTVS APPs

As integrações máquina para máquina, ou seja, as chamadas às APIs dos TOTVS APPs a partir do BACKEND da APLICAÇÃO, devem utilizar tokens gerados via Client Credentials.

Client Credentials

Toda e qualquer instalação de uma aplicação deve possuir uma Client Credential específica, necessária para as chamadas ao ACCOUNTS ou para os TOTVS APPs.
Essas Client Credentials são geradas pela plataforma de provisionamento, mas devem ser configuradas nas instâncias das aplicações.

Sobre o TOTP Server

O TOTP Server é uma ferramenta, na forma de um executável, que pode ser utilizada para fazer a configuração e a validação dos tokens TOTP acima mencionados.
Ela pode ser utilizada como um comando no sistema operacional ou como um serviço HTTP, neste último caso deve ser instalado no mesmo servidor do BACKEND da aplicação, para que não ocorra o tráfego das chaves TOTP pela rede.

TOTPServer server [port:3000]

TOTPServer generate
< url|qrcode >
< applicationname >
< secretkey >

TOTPServer validate
< tokentovalidate >
< secretkey >

Generate

POST `/generate`

Endpoint that generates a TOTP QR Code

Parameters Cancel

Name	Description
type * required string *(query)*	qrcode | url qrcode
product * required string *(query)*	Product's name. Protheus
secret * required string *(query)*	User's secret. totvs@123

Execute | Clear

Responses Response content type: application/json

Curl

```
curl -X 'POST' \
  'http://localhost:3000/generate?type=qrcode&product=Protheus&secret=totvs%40123' \
  -H 'accept: application/json' \
  -d ''
```

Request URL

```
http://localhost:3000/generate?type=qrcode&product=Protheus&secret=totvs%40123
```

Server response

Code	Details
200	Response body

Response headers

```
content-length: 2335
content-type: image/png
date: Wed,13 Mar 2024 14:31:46 GMT
vary: Accept-Encoding
x-powered-by: Express
```

Responses

Code	Description
200	TOTP QR Code (PNG format)
400	Invalid query parameters
500	Error generating QR Code

Validate

POST `/validate`

Endpoint that validates a TOTP token

Parameters Cancel

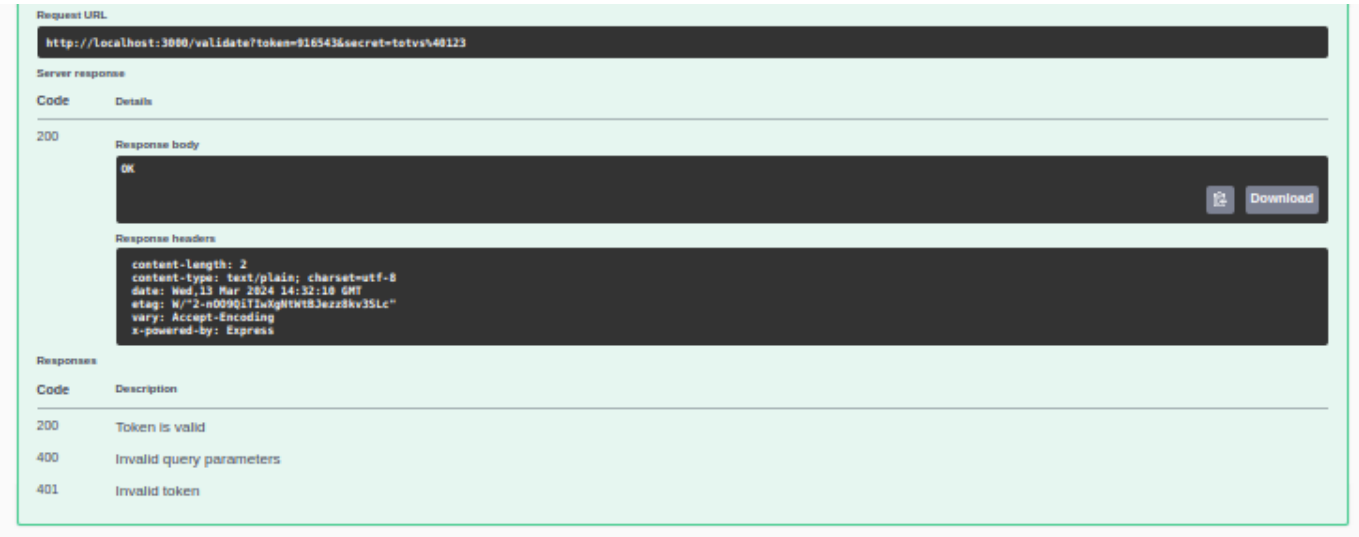
Name	Description
token * required string *(query)*	Token to be validated. 916543
secret * required string *(query)*	User's secret. totvs@123

Execute | Clear

Responses Response content type: application/json

Curl

```
curl -X 'POST' \
  'http://localhost:3000/validate?token=916543&secret=totvs%40123' \
  -H 'accept: application/json' \
  -d ''
```



Referências

- OAuth 2.0
- OpenID Foundation
- OAuth 2.0 RFC 6749
- TOTP RFC 6238

Metadado

Metadado	
Autores	**roger.steuernagel@totvs.com.br**
Revisores	**tamara.fonseca@fluig.com**, **eduardo.teixeira@totvs.com.br**, **gustavo.martins@fluig.com**, **conrado.gomes@totvs.com.br**, **felipe.conti@totvs.com.br**
Origem	**N/A**
Categoria	**Conceitual**
Solução	**N/A**
Nível de Recomendação	**Requerido**
Status	**Ativa**
Status de Publicação	**Público**