
Newton
Quem se prepara, não para.

Arquitetura de Aplicações Web

5º período

Professora: Michelle Hanne

Sumário

- JSON JWT com Assinatura assimétrica do JWT

Passo a Passo

Newton
Quem se prepara, não para.

1) Instalar o Postman - se não tiver
   – https://www.postman.com/downloads/
2) Baixar o conteúdo da pasta Exemplo_JWT_Criptografia

Passo a Passo

Newton
Quem se prepara, não para.

3) Instalar os pacotes e atualizar as dependências

npm i express cookie-parser body-parser jsonwebtoken

Fonte: https://www.luiztools.com.br/post/autenticacao-json-web-token-jwt-em-nodejs/

Criptografia Assimétrica

Newton
Quem se prepara, não para.

Criptografia de chave pública, também conhecida como **criptografia assimétrica**, é qualquer sistema criptográfico que usa pares de chaves: *chaves públicas*, que podem ser amplamente disseminadas, e *chaves privadas* que são conhecidas apenas pelo proprietário. Isto realiza duas funções: autenticação, onde a chave pública verifica que um portador da chave privada parelhada enviou a mensagem, e encriptação, onde apenas o portador da chave privada parelhada pode decriptar a mensagem encriptada com a chave pública

Fonte: https://pt.wikipedia.org/wiki/Criptografia_de_chave_p%C3%BAblica

Criptografia Assimétrica

Newton
Quem se prepara, não para.

- Usaremos a Criptografia assimétrica, onde o **emissor/servidor** irá **gerar o token assinado com a chave privada (de posse somente dele)** e os consumidores/clientes podem verificá-lo usando a chave pública (de posse de todos **microservices**).

- Assim, **todos microservices confiam na emissão de tokens a partir um servidor central**, enquanto podem validar sua **assinatura** para garantir que não foi forjado, sem saber o **secret original**.

Fonte: https://www.luiztools.com.br/post/autenticacao-json-web-token-jwt-em-nodejs/

Criptografia Assimétrica

Newton
Quem se prepara, não para.

- Primeiro, vamos criar um par de chaves (uma pública e outra privada) usando o algoritmo RSA, um dos mais famosos e seguros do mundo de tipo assimétrico.
- https://www.csfieldguide.org.nz/en/interactives/rsa-key-generator/

Fonte: https://www.luiztools.com.br/post/autenticacao-json-web-token-jwt-em-nodejs/

Passo a Passo

Newton
Quem se prepara, não para.

4) Criar os arquivos **public.key** e **private.key** na raiz do projeto :

-----BEGIN RSA PUBLIC KEY-----

MIGJAoGBAJpt3D6TDgVnxsVl8wv3KwfHDLvlYl6IA7UXEsvlpvZzBzU9fJpkjrzR

4cXg8wMY6z5/aQ5ygJ/V/3JfKiq571EUycq4/gn7sUi/ufON/H4bEMAJLmirJu4q

kTWWutDSOD9PO1hxrXwpOS0JFOWYcVjW3Lh9rDf6edh3QfYvHAAXAgMBAAE=

-----END RSA PUBLIC KEY-----

Passo a Passo

Newton
Quem se prepara, não para.

```
-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQCabdw+kw4FZ8bFZfML9ysHxwy7yGJeiAO1FxLL5ab2cwc1PXya
ZI680eHF4PMDGOs+f2kOcoCf1f9yXyoque9RFMnKuP4J+7FIv7nzjfx+GxDACS5o
qybuKpE1lrrQ0jg/TztYca18KTktCRTlmHFY1ty4faw3+nnYd0H2LxwAFwIDAQAB
AoGAK84J8X4JNiNP0OKwZK6B+DzQMdwPez0dwBqBbHECQVozqzh7xdfMXWczocvD
YxeIczv08vIr5irvwOOZtHD1nxtupEjxPTN2kOZzc5S6cWSkGEa5KViXI6AfIizY
DtE2m9x4wt226XOvHQes759AqGbrWPamh13850vqDznqBnECQQDLxg6wYSRuK1ux
FFT0TZ7cBF2boqcQrrd5CkNZv+TBKT+h8AVUFYGL+VidHWCIwQHQOdpfz82HgThV
2pCBlm6bAkEAwgI5Pm+WsPj2b1dKYBfU7kaRQ9sFMbG+yYFoAbwarV2WggSGK90C
ib31+/SOEJgGf/X2aI+hHLwEH4MryRnuNQJBAKf59FHhQi6u/z7SC9X3xmSIFMIf
KqjN3eChXTO2w9OXNSVAvqO5tri0KyAY/2K798q2ZhVlL3/sPYxIR6cLYtECQFHI
tC7wxXqHCQO4rX7CrR1hHB2Zt5/SSRYS+jA5BpnsqOTYWWMmQMqmDIsQ9c5i9+J7
YkuDuKwvSiGqnwugZb0CQDJ/CXHM1n8aatGiRd6XRjj13m6jRt2yCFnDfUdftKV7
XUOYpn+PKmzp68hCYooPol8sNT7efJU6PJ8CMTXBeQ8=
-----END RSA PRIVATE KEY-----
```

Rota Login

Newton
Quem se prepara, não para.

```
//rota de login
app.post('/login', (req, res, next) => {
  if(req.body.user === 'arquiteturaWeb' && req.body.password === '123'){
      //auth ok
      const id = 1; //esse id viria do banco de dados
      var privateKey  = fs.readFileSync('./private.key', 'utf8');
      var token = jwt.sign({ id }, privateKey, {
          expiresIn: 300, // 5min
          algorithm:  "RS256" //SHA-256 hash signature
      });

      console.log("Fez login e gerou token!");
      return res.status(200).send({ auth: true, token: token });
  }

  return res.status(401).send('Login inválido!');
})
```

- O privateKey substitui o secret padrão que estávamos usando. A leitura do arquivo da chave é feita usando módulo fs e nas opções de assinatura (terceiro argumento da função) dizemos o algoritmo de hashing que o RSA vai usar no seu algoritmo interno (RS256 representado porSHA-256).

Fonte: https://www.luiztools.com.br/post/autenticacao-json-web-token-jwt-em-nodejs/

Função verifyJWT

Newton
Quem se prepara, não para.

```
//função que verifica se o JWT é ok
function verifyJWT(req, res, next){
  var token = req.headers['x-access-token'];
  if (!token)
      return res.status(401).send({ auth: false, message: 'Token não
informado.' });

  var publicKey  = fs.readFileSync('./public.key', 'utf8');
  jwt.verify(token, publicKey, {algorithm: ["RS256"]}, function(err, decoded)
{
      if (err)
          return res.status(500).send({ auth: false, message: 'Token
inválido.' });

      req.userId = decoded.id;
      console.log("User Id: " + decoded.id)
      next();
  });
```

Carregamos a public key a partir do respectivo arquivo, passamos ela pra função verify, bem como um objeto informando o algoritmo de hashing que usamos junto do RSA (RS256 refere-se a SHA-256).

Fonte: https://www.luiztools.com.br/post/autenticacao-json-web-token-jwt-em-nodejs/