Artigo
Heloisa Paiva · jan 17, 2023 8min de leitura

Token Authentication - O básico que você precisa para começar a programar

Porque eu decidi escrever este texto

Recentemente eu recebi o desafio de criar um método de autenticação segura para autorizar acesso a alguns dados, mas infelizmente eu não tinha nenhuma experiência com essas configurações de segurança e senti que me faltava, alguns conceitos básicos para compreender melhor a documentação oficial.

Depois de estudar e conseguir entregar as classes que me pediram para desenvolver, eu gostaria de compartilhar um pouco do meu novo conhecimento, que me ajudou seguir os tópicos da documentação.

Começando com o básico: a santíssima trindade dos servidores

First, it's important to understand what exactly we're dealing with. In general, we have some data that may be sensitive, or for any kinds of reason needs to be protected. There are some people (users) that will be able to see them, some will be able to change them, some won't have any kind of access. To take care of the users, access and data, we will have three servers: the client, the resources and the authorization.

Primeiro, é importante entender com o que exatamente estamos lidando. Em geral, temos alguns dados que podem ser sensíveis, ou por qualquer razão precisam ser protegidos. Há algumas pessoas (usuários) que poderão vê-los, outras poderão alterá-los e outras não terão nenhum tipo de acesso. Para cuidar dos usuários, acesso e dados, teremos três servidores: O cliente, o de recursos e o de autorização.

Este é o fluxo:

O servidor cliente envia uma requisição para o servidor de autorização, que fará a validação: checa se pode autorizar o acesso (essa validação será discutida com mais detalhes posteriormente). Uma vez que o servidor decide que é ok liberar o acesso, envia de volta ao cliente um token. O cliente agora pode enviar este token para o servidor de recursos, que vai reconhecê-lo e devolver ao cliente a informação necessária.

Para ter certeza de que tudo está claro antes de irmos ao próximo passo, vamos calçar os sapatos de cada servidor:

- **Eu sou o servidor de autorização:** quero receber informação do cliente e se eu reconhecê-lo, vou fornecer-lhe a chave (o token) para acessar os dados. Eu sou o recepcionista.
- **Eu sou o servidor de recursos:** eu quero receber um token. Se esse token é valido, seja quem for que o mandou terá permissão de acessar os dados que eu guardo. Eu sou o banco.
- **Eu sou o servidor cliente:** eu quero acessar os dados do servidor de recursos, mas não sei a chave. Então, vou me identificar para o servidor de autorização para que ele me dê a chave. Então eu mostro minha chave para o servidor de recursos, que vai me levar aos dados que preciso. Eu sou o freguês.

Seguindo adiante: o token e a validação

Mas o que é a validação e esse token? Bem, é simples. O token é uma *string* que o servidor de recursos irá interpretar como o tipo de acesso que esse cliente pode ter aos dados. Então, se tenho acesso de visualização, meu token dirá ao servidor de recursos "Ela pode realizar um SELECT!" e algumas outras coisas, como quem sou eu, por quanto tempo posso usar esse token, etc. Ou, se tenho qualquer tipo de acesso, o token vai tagarelar tudo sobre mim ao servidor de recursos e lhe dirá "Ela pode fazer o que quiser nessa tabela!".

Na prática, o servidor de autorização usará algum tipo de criptografia para transformar "Oi, eu sou a Heloísa e minha senha para esse servidor é JuroDeDedinhoSerHonesta.123" em "iuiDBIldb3287$@*++==" ou algo similar, que significa "Servidor de recursos, dê a seja lá quem enviou esse token, o acesso a SELECT" ou qualquer outro tipo de acesso que corresponda. O token poderia facilmente ser uma *string* dizendo literalmente "ACESSO AO SELECT", mas isso não teria nenhuma segurança garantida.

Contudo, nem tudo são flores, então vamos complicar um pouco. É claro que o servidor de autorização não proverá esse token a qualquer um que pedir, porque você tem que ser muuuito especial para acessar esses dados, não é mesmo? Para resolver isso, esse servidor tem uma tabela simples com nomes de usuários e as respectivas senhas e acessos permitidos. O cliente deve se identificar com um nome de usuário e senha, para que a validação seja feita. A validação, então, é literalmente checar a tabela pelos usuários e senhas. A parte que realmente vamos complicar um pouco é como o cliente envia essa informação.

Há alguns métodos:

- O cliente envia uma requisição "Oi Aut, quero me identificar", e o servidor de autorização faz um *prompt* ao usuário para que digite o nome e a senha. Após fazer a validação, retorna um código de autorização, que o cliente enviará novamente ao servidor de autorização para conseguir o token necessário para o servidor de recurso;
- O mesmo fluxo acontece, mas ao invés de receber um código de autorização, o cliente receberá diretamente o token;
- O cliente envia o nome de usuário e a senha numa requisição ao servidor de autorização, que fará uma validação e enviará de volta o token;
- Não há usuário, o cliente envia um ID de Cliente e um Segredo de Cliente (Client ID e Client Secret) para obter o token.

Cada método tem seus prós e contras, dependendo do nível de segurança que os dados necessitam. Discutiremos isso posteriormente.

Uma pequena comparação para que as coisas fiquem claras

Isso me deu como uma imagem do Hagrid e do Harry Potter entrando no banco Gringotts para pegar a pedra filosofal. Com a exceção de que o Dumbledore é o servidor cliente, que envia o Harry, o usuário, e Hagrid, a requisição do cliente, dizendo "Olá, preciso de uns dados do cofre". O cofre é o servidor de recursos e os duendes são o servidor de autorização, validando se Hagrid pode pegar os dados para o Dumbledore ou não. E, é claro, a chave para o cofre é o token. Se você não assistiu Harry Potter provavelmente está ainda mais confuso que no início, mas agora você pode colocar o filme para assistir pensando "isso é para estudo, não estou procrastinando". Essa é uma ótima comparação na verdade, porque o Harry só sabe o que está obtendo do cofre uma vez que o abre, o que é todo o objetivo da autenticação. Além disso, o Harry não faz nada além de dizer "Oi, sou Harry Potter e não tenho ideia do que está acontecendo, mas de alguma forma consegui acesso a um cofre riquíssimo que por acaso é meu!". Observe como, para o cofre do Dumbledore, Harry tinha acesso de SELECT, pois poderia ver o que estava lá dentro, mas não poderia fazer nada com isso. Para o seu próprio cofre, tinha acesso ALL, porque poderia retirar ou depositar qualquer coisa, permitira cesso a quem quisesse, etc...

Talvez para o terceiro método, Harry teria ido com Hagrid, mas iria fazer compras na Gemialidades Weasley enquanto o Hagrid pegava o pacote, e para o último método, Dumbledore teria enviado o Hagrid sozinho ao banco.

Chega de Harry Potter, vamos voltar ao trabalho.

Colocando as mãos na massa: a prática

Agora que temos todos os conceitos mais claros, precisamos colocar a coisa para funcionar. Para começar, você vai querer se perguntar "qual servidor estou desenvolvendo?"

Se está desenvolvendo mais de um, pode ser uma boa ideia fazer um de cada vez. Responder à pergunta te levará a focar em o que exatamente essa classe deve fazer:

- **Servidor de recursos:** receber uma requisição com um token e checar se esse token é válido. Se é, realiza a ação enviada pela requisição (por exemplo SELECT * FROM bd.tabela) e responde com a informação obtida.

- **Servidor de autorização:** depende do tipo de validação escolhida. Para a primeira, se recebe uma requisição vazia, faz um *prompt* ao usuário para obter seu nome e senha. Após esse passo, realiza a validação e se estiver ok, responde com um código de autorização. Se recebe uma requisição com um código de autorização válido, responde com o token. Para o segundo, recebe uma requisição vazia, faz um *prompt* ao usuário para obter seu nome e senha, realiza a validação e se tudo corre bem responde com o token. Para o terceiro, recebe uma requisição com um nome de usuário e senha, faz a validação e se está ok responde com o token. Para a última, recebe uma requisição com um Client ID e Client Secret, faz a validação e se estiver ok responde com o token.

- **Servidor cliente:** depende do tipo de validação que o servidor de autorização usa. Para o primeiro, envia uma requisição ao servidor de autorização e se o usuário tiver acesso, recebe um código na resposta, envia esse código em outra requisição para o servidor de autorização e recebe um token. Finalmente, envia esse token ao servidor de recursos e recebe os dados. Para o segundo, envia uma requisição ao servidor de autorização e se o usuário tem acesso, recebe um token na resposta, envia o token numa requisição ao servidor de recursos e recebe os dados. Para o terceiro, recebe um nome de usuário e senha (pode ser de outro servidor, por um *prompt* ao usuário, podem ser nomes e senhas fixos, etc.), os envia numa requisição para o servidor de autorização e, se o usuário tem acesso, recebe o token e blablablá ao servidor de recursos e de volta com os dados novamente. Para o último, o servidor tem um Client ID e um Client Secret que serão enviados numa requisição ao servidor de autorização, receberá o token na resposta e - blablablá mais uma vez.

Discutindo os métodos de validação

A maior parte da segurança se deve ao método de validação. Se você recebe ordens específicas de desenvolver com um ou outro método, não precisa se preocupar com isso, mas é interessante saber por quê seu chefe fez

essa escolha.

Então, se você é quem deve escolher ou está apenas interessado, aqui estão algumas boas discussões sobre autenticação por token:

Do StackOverflow: Proper way to send username and password from client to server

Do StackExchange:Why shouldn't my client just send the user's username and password with every request?

PS.: não irei discutir sobre esses assuntos aqui porque é fora do escopo deste artigo e, além disso, acredito que vale a pena ler os links na íntegra.

Fim!

Muito obrigada por ler e espero que este artigo tenha sido de alguma ajuda.

Sinta-se à vontade para comentar dúvidas ou entrar em contato comigo para que eu ajude em casos específicos ou somente para discutir. Ficarei feliz de conversar!

#Autenticação #Controle de acesso #Innovatium #Documentação #InterSystems IRIS #Portal de Aprendizagem

---

URL de origem:https://pt.community.intersystems.com/post/token-authentication-o-b%C3%A1sico-que-voc%C3%AA-precisa-para-come%C3%A7ar-programar