TIC em trilhas

Módulo 07

Segurança para APIs

TIC em trilhas

Aula 1 - Conceitos de segurança

Importância, conhecendo as principais falhas e como aplicar em APIs

Imagem extraída através do acervo gratuito do Freepik

1.1 A importância da segurança em APIs

A segurança em APIs, especialmente em um ambiente Node.js, desempenha um papel crítico na proteção dos dados sensíveis dos usuários.

Na era da tecnologia, a informação chega a ser uma "moeda" de troca poderosa, é necessário desenvolver aplicações que tenham um nível de segurança alta, para garantir que os dados não caiam em mãos erradas.

Nesta parte do treinamento, iremos abordar técnicas que irão auxiliar muito a criar projetos em Node.js mais seguros.

Imagem extraída através do acervo gratuito do Freepik

1.2 Principais falhas de segurança

- **Falta de Autenticação e Autorização Adequadas**: Uma das falhas mais críticas é a ausência ou implementação incorreta de autenticação e autorização.

- **Injeção de Código:** Vulnerabilidades de injeção de código, como SQL Injection e NoSQL Injection, acontecem quando dados não sanitizados são diretamente inseridos em comandos SQL ou NoSQL.

- **Exposição de Dados Sensíveis**: APIs mal configuradas podem expor dados sensíveis, como informações de autenticação, chaves de API, tokens e detalhes de usuário, por meio de respostas não seguras.

- **Falta de Controle de Acesso**: Falhas na implementação de controles de acesso adequados podem permitir que usuários não autorizados acessem recursos ou funcionalidades que deveriam estar restritos.

- **Cross-Origin Resource Sharing (CORS) Mal Configurado**: Uma configuração inadequada do CORS pode permitir que um site malicioso faça solicitações para a API de um usuário autenticado, comprometendo a segurança.

- **Excesso de Privilégios:** Conceder mais privilégios do que o necessário a determinadas contas de usuário pode levar a vulnerabilidades. Por exemplo, se um usuário comum tiver acesso administrativo sem necessidade, isso pode ser explorado por um invasor.

- **Falhas de Validação de Entrada**: A falta de validação adequada de entrada pode permitir que um invasor envie dados maliciosos, como caracteres especiais ou payloads de ataque, que podem ser processados pela API de forma insegura.

- **Exposição de Endpoints Sensíveis**: Se endpoints sensíveis, como aqueles para exclusão de dados, não forem adequadamente protegidos e autenticados, eles podem ser alvo de ataques maliciosos.

1.3 Tornando uma aplicação segura

Veja abaixo algumas dicas interessantes para deixarmos nossas aplicações em Node.js mais seguras:

- **Autenticação:** Verifique a identidade do usuário que está acessando a API. Isso pode ser feito com JWT (JSON Web Tokens), OAuth, ou mesmo por meio de autenticação baseada em sessão.

- **Autorização:** Depois de autenticar o usuário, é importante controlar quais recursos e dados ele pode acessar. Isso é feito por meio de sistemas de autorização, como RBAC (Role-Based Access Control) ou ABAC (Attribute-Based Access Control).

- **Validar Inputs:** Valide e sanitize todos os dados que chegam à sua API para evitar ataques de injeção de código, como SQL Injection ou XSS (Cross-Site Scripting).

- **Mensagens detalhadas:** Ao lidar com erros, retorne mensagens genéricas ao cliente. Mensagens de erro detalhadas podem ser úteis para os invasores, fornecendo informações sobre o sistema.

- **Logging Seguro:** Registre logs de forma segura e não inclua informações sensíveis, como senhas ou tokens de autenticação.

- **Proteção contra XSS:** Para proteger contra ataques XSS, utilize cabeçalhos HTTP apropriados, como CSP (Content Security Policy).

- **Testes de Segurança:** Realize testes em suas aplicações com frequência, assim você garante que as implementações de segurança estão em perfeito funcionamento.

- **Dependências de terceiros:** Tome cuidado ao baixar dependências de terceiros para desenvolver sistemas, pois podem conter algum *malware*.

1.4 Explore mais

Gostou das dicas de segurança? Saiba que é fundamental nossos projetos terem esse cuidado, precisamos garantir a integridade dos dados de todos os clientes, nos links abaixo há conteúdos bem interessantes para auxiliar nos estudos sobre segurança:

- Node.js - Security Best Practices
- Express - Melhores Práticas de Segurança: Produção
- Cubos Academy - Protegendo as suas aplicações NodeJS com Middlewares no Express.js

Bons estudos

Imagem extraída através do acervo gratuito do Freepik

TIC em trilhas

Aula 2 - JWT

O que é, arquitetura e como podemos implementar em nossos projetos

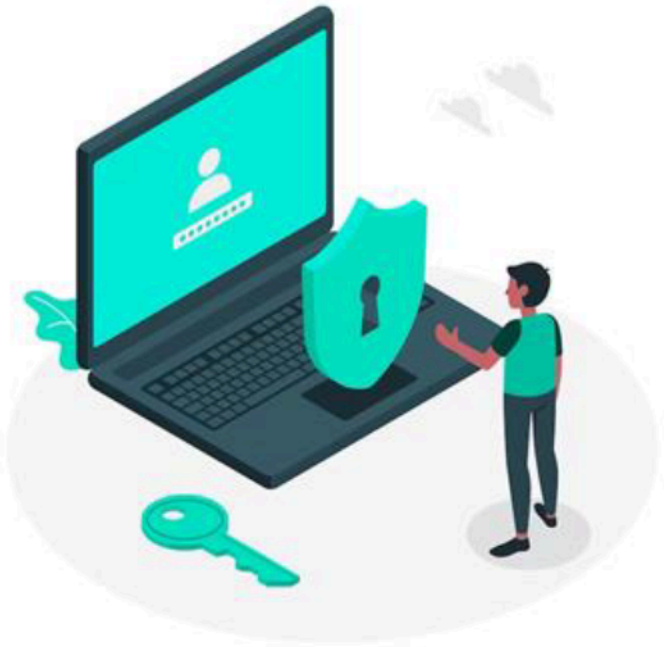

Imagem extraída através do acervo gratuito do Freepik

2.1 O que é

O JSON Web Token (JWT) foi criado pelo grupo IETF (Internet Engineering Task Force) como um padrão aberto e publicado como RFC 7519 em maio de 2015. A RFC (Request for Comments) é um tipo de documento que descreve os padrões, protocolos e procedimentos usados na Internet.

O grupo de trabalho JSON Web Token (JWT) da IETF foi responsável pelo desenvolvimento do JWT. Este grupo é dedicado a padronizar e promover o uso de tokens baseados em JSON para autenticação e autorização em sistemas distribuídos na Internet.

Dessa forma, o JWT se tornou uma especificação amplamente adotada e utilizada para transmitir informações de forma segura entre partes, fornecendo um formato eficiente e autossuficiente para tokens de autenticação e autorização.

TIC em trilhas

2.2 Estrutura

O JWT é divido em três partes, sendo elas:

```
{
    "alg": "HS256",
    "typ": "JWT"
}
```

Header

```
{
    "sub": "1234567890",
     "name": "John Doe",
     "iat": 1516239022
}
```

Payload

```
HMACSHA256(
        base64UrlEncode(hea
        der) + "." +
        base64UrlEncode(payl
        oad),
        suaSenha
)
```

Signature

Vale lembrar que o JWT é um token, então não armazene informações sensíveis como senhas e cpfs, pois um token não foi criado para criptografar dados em um nível considerado complexo, mas sim, garantir que determinado cliente tenha permissão para realizar determinadas funcionalidades que o sistema dispõe.

Um token não garante uma criptografia segura dos dados, tanto que é possível descobrir com facilidade o valor que está no **payload**, veja o exemplo abaixo criado no site do JWT, os caracteres em tom rosado são o **payload**:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
Jub21lIjoiUmFsZiIsInNlbmhhIjoiMTIzIiwia
WF0IjoxNTE2MjM5MDIyfQ.umbBW79--
rcdGtM4X-LyYeuLHJ6iQtr_OlamWv67EEk
```

PAYLOAD: DATA

```
  "nome": "Ralf",
  "senha": "123",
  "iat": 1516239022
}
```

Imagens do autor

2.4 Explore mais

Quer aprender ainda mais sobre o JWT? Abaixo você encontra materiais bem bacanas para complementar seus estudos dessa fantástica tecnologia.

- Geekforgeeks - JWT Authentication with Node.js
- Simplilearn - Understanding JWT Authentication with Node.js
- Toptotal - How to use JWT and Node.js for better App Security

Bons estudos

Imagem extraída através do acervo gratuito do Freepik

TIC em trilhas

Referências bibliográficas

Treinaweb: O que é JWT. Disponível em: https://www.treinaweb.com.br/blog/o-que-e-jwt. Acesso em março de 2024.

JWT: Introduction to JSON web token. Disponível em: https://jwt.io/introduction. Acesso em março de 2024.

Node.js: Security best practices. Disponível em: https://nodejs.org/en/learn/getting-started/security-best-practices. Acesso em março de 2024.

Ninelabs: Segurança em APIs Nodejs. Disponível em: https://ninelabs.blog/seguranca-em-apis-node-js-melhores-praticas-para-proteger-suas-aplicacoes/. Acesso em março de 2024.

Express: Melhores Práticas de Segurança para o Express em Produção. Disponível em: https://expressjs.com/pt-br/advanced/best-practice-security.html. Acesso em março de 2024.