Unidade 4

Melhores práticas em produção

Visão Geral

- O termo "produção" refere-se ao estágio no ciclo de vida do software onde um aplicativo ou API está geralmente disponível para os seus usuários finais ou consumidores

Não use versões descontinuadas ou vulneráveis do Express

- Os Express 2.x e 3.x não são mais mantidos!

Use TLS

- Se o seu aplicativo negocia com ou transmite dados sensíveis, use a Segurança da Camada de Transporte (TLS) para proteger a conexão e os dados

Use Helmet

- O Helmet é na realidade apenas uma coleção de nove funções de middlewares menores que configuram cabeçalhos HTTP relacionados à segurança:
    - **CSP** (Content-Security-Policy)
    - A **hidePoweredBy** remove o cabeçalho X-Powered-By.
    - **htps** ou Strict-Transport-Security
    - A **ieNoOpen** configura o X-Download-Options para o IE8+.
    - A **noCache** configura os cabeçalhos Cache-Control e Pragma para desativar o armazenamento em cache no lado do cliente.
    - A **noSniff** configura o X-Content-Type-Options para evitar que os navegadores procurem por MIME uma resposta a partir do content-type declarado.
    - A **frameguard** configura o cabeçalho X-Frame-Options para fornecer proteção clickjacking.
    - A **xssFilter** configura o X-XSS-Protection para ativar o filtro de Cross-site scripting (XSS) nos navegadores da web mais recentes.

Use cookies de maneira segura

- Para assegurar que os cookies não deixem o seu aplicativo aberto a ataques, não use o cookie de sessão padrão e configure as opções de segurança de cookies adequadamente.

- Não use o nome do cookie da sessão padrão

```
var session = require('express-session');
app.set('trust proxy', 1) // trust first proxy
app.use( session({
   secret : 's3Cur3',
   name : 'sessionId',
  })
);
```

Configure as opções de segurança de cookie

- **secure** - Assegura que o navegador só envie o cookie por HTTPS.

- **httpOnly** - Assegura que o cookie seja enviado apenas por HTTP(S), não por cliente JavaScript, ajudando assim a se proteger contra ataques de cross-site scripting.

- **domain** - indica o domínio do cookie; use-o para comparação contra o domínio do servidor em que a URL está sendo solicitada. Se elas corresponderem, verifique o atributo de caminho em seguida.

- **path** - indica o caminho do cookie; use-o para comparação contra o caminho da solicitação. Se este e o domínio corresponderem, então envie o cookie na solicitação.

- **expires** - use para configurar uma data de expiração para cookies persistentes.

Configure as opções de segurança de cookie

```
var session = require('cookie-session');
var express = require('express');
var app = express();

var expiryDate = new Date( Date.now() + 60 * 60 * 1000 ); // 1 hour
app.use(session({
  name: 'session',
  keys: ['key1', 'key2'],
  cookie: { secure: true,
            httpOnly: true,
            domain: 'example.com',
            path: 'foo/bar',
            expires: expiryDate
          }
  })
);
```

Assegure que suas dependências sejam seguras

Assegure que suas dependências sejam seguras

- **nsp** e **requireSafe**

```
$ npm i nsp -g
```

Use este comando para enviar o arquivo `npm-shrinkwrap.json` para validação para o nodesecurity.io:

```
$ nsp audit-shrinkwrap
```

Use este comando para enviar o arquivo `package.json` para validação para o nodesecurity.io:

```
$ nsp audit-package
```

Aqui está como usar o requireSafe para auditar seus módulos Node:

```
$ npm install -g requiresafe
$ cd your-app
$ requiresafe check
```