React: how to protect routes with a Bearer Token

Security is a fundamental component in the development of modern web applications. When it comes to securing routes in a React application, using a Bearer Token is a common and effective practice. In this article, we will explore how to implement this security in your React routes.

What is a Bearer Token?

A Bearer Token is a type of access token that allows its holder to access protected resources. In the context of web applications, it is often used for user authentication. When a user successfully authenticates, they receive a Bearer Token that must be included in subsequent requests to gain access to resources protected by the backend.

Initial Configuration

First of all, make sure you have an authentication system implemented on your backend that provides a Bearer Token after user authentication. Once you have this part configured, you can start securing routes in your React frontend.

Using React Router

React Router is a very popular library for managing navigation in React applications. To protect routes, we can leverage the concept of private routes, which require authentication for access.

```
// src/components/PrivateRoute.js
```

```
import React from 'react';
import { Route, Redirect } from 'react-router-dom';

const PrivateRoute = ({ component: Component,
isAuthenticated, ...rest }) => (
   <Route
     {...rest}
     render={(props) =>
       isAuthenticated ? (
         <Component {...props} />
       ) : (
         <Redirect to="/login" />
       )
     }
   />
);

export default PrivateRoute;
```

You can now use the `PrivateRoute` component in places where you want to require authentication. For example:

```
// src/App.js

import React from 'react';
import { BrowserRouter as Router, Route, Switch }
from 'react-router-dom';
import PrivateRoute from './components/PrivateRoute';
import Home from './components/Home';
import Login from './components/Login';

const App = () => {
   const isAuthenticated = /* Logic to verify
authentication */;

   return (
     <Router>
       <Switch>
         <Route path="/login" component={Login} />
         <PrivateRoute
           path="/home"
           component={Home}
           isAuthenticated={isAuthenticated}
         />
         {/* Other routes */}
       </Switch>
     </Router>
   );
};

export default App;
```

When a user successfully authenticates, you will need to save the Bearer Token somewhere safe, such as in application state or cookies. Be sure to include the Bearer Token in your requests to the backend to gain access to protected resources.

```
// Example of how you might handle the Bearer Token
after authentication

const handleLogin = async (credentials) => {
   try {
     // Make the authentication request to the
backend
     const response = await api.post('/login',
credentials);

     // Extract the Bearer Token from the response
     const { token } = response.data;

     // Save the Bearer Token in the application
state or cookies
     setToken(token);

     // Perform other necessary actions, such as
redirecting to a secure page
     history.push('/home');
   } catch (error) {
     // Handle authentication errors
     console.error('Error authenticating', error);
   }
};
```

Conclusion

Securing routes with a Bearer Token in React is an important practice to ensure the security of your applications. By using React Router and implementing private routes, you can ensure that only authenticated users can access protected resources. Always remember to handle the Bearer Token carefully and include it correctly in your requests to the backend.