A7-Missing Function Level Access Control

Exploitability: EASY | Prevalence: COMMON | Detectability: AVERAGE | Technical Impact: MODERATE

Description

Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed.

Attack Mechanics

If requests are not verified for access rights on server, attackers can forge requests in order to access functionality without proper authorization.

In the insecure demo application, this vulnerability exists in benefits module, which allows changing benefit start date for employees. The link to the benefits module is visible only to the admin user (user: admin, password: Admin_123). However, an attacker can access this module simply by logging in as any non-admin user and accessing benefits url (/benefits) directly.

How Do I Prevent It?

Most web applications don’t display links and buttons to unauthorized functions, but this “presentation layer access control” doesn't actually provide protection. You must also implement checks in the controller or business logic.

Source Code Examples

In vulnerable application, there is no authorization check for benefits related routes in `routes/index.js`

```
// Benefits Page
app.get("/benefits", isLoggedIn, benefitsHandler.displayBenefits);
app.post("/benefits", isLoggedIn, benefitsHandler.updateBenefits);
```

This can be fixed by adding a middleware to verify user's role:

```
// Benefits Page
app.get("/benefits", isLoggedIn, isAdmin, benefitsHandler.displayBenefits);
app.post("/benefits", isLoggedIn, isAdmin, benefitsHandler.updateBenefits);
```

To implement `isAdmin` middleware, check if isAdmin flag is set for the logged in user in database.
For example, here is middleware function that can be added to `routes\session.js` :

```
this.isAdminUserMiddleware = function(req, res, next) {
    if (req.session.userId) {
        userDAO.getUserById(req.session.userId, function(err, user) {
             if(user && user.isAdmin) {
                 next();
             } else {
                 return res.redirect("/login");
             }
        });
    } else {
        console.log("redirecting to login");
        return res.redirect("/login");
    }
};
```

It can be then made available in `routes/index.js` router as:

```
var SessionHandler = require("./session");
//Middleware to check if user has admin rights
var isAdmin = sessionHandler.isAdminUserMiddleware;
```