Week 2: Implementing Security Measures

I installed npm validator libaray to validate the user.

1. **Implementing anti-CSRF packages such as the OWASP CSRFGuard.**

Login

```
<form action="/admin/login" method="POST" class="card-body">
```

Registration

```
<form action="/register" method="POST" class="card-body">
```

Fix:

Login

```
<form action="/admin/login" method="POST" class="card-body">
  <input type="hidden" name="_csrf" value="<%= csrfToken %>">
  <!-- Your other form inputs -->
</form>
```

Register

```
<form action="/register" method="POST" class="card-body">
  <input type="hidden" name="_csrf" value="<%= csrfToken %>">
  <!-- Your other form inputs -->
</form>
```

App.js

```
const csrf = require("csurf");
app.use(cookieParser());
app.use(express.urlencoded({ extended: false }));
```

```
// Setup CSRF protection with cookie
const csrfProtection = csrf({ cookie: true });
app.use(csrfProtection);

// Make the CSRF token available in all views
app.use((req, res, next) => {
  res.locals.csrfToken = req.csrfToken();
  next();
});
```

2. Content Security Policy (CSP) Header Not Set

Login.ejs

```
<script>
   const password = document.querySelector("#pwd");
   const eyeIcon = document.querySelector("#eye");
   const togglePassword = document.querySelector("#togglePassword");

   togglePassword.addEventListener("click", () => {
      if (eyeIcon.classList.contains("bi-eye")) {
         password.setAttribute("type", "text");
         eyeIcon.classList.replace("bi-eye", "bi-eye-slash");
      } else {
         password.setAttribute("type", "password");
         eyeIcon.classList.replace("bi-eye-slash", "bi-eye");
      }
   });
</script>
```

Fix:

npm install helmet

App.js

```
const helmet = require("helmet");

app.use(helmet({
  contentSecurityPolicy: {
    directives: {
      defaultSrc: ["'self'"],
      scriptSrc: ["'self'"],
      styleSrc: ["'self'", "'unsafe-inline'"], // allow inline styles if needed
      imgSrc: ["'self'", "data:"],
      connectSrc: ["'self'"],
      fontSrc: ["'self'"],
      objectSrc: ["'none'"],
      upgradeInsecureRequests: [],
    }
  }
}));
```

Register

```
<script>
    document.addEventListener('DOMContentLoaded', function () {
        const form = document.querySelector('form');

        form.addEventListener('submit', function (event) {
            // Prevent form submission
            event.preventDefault();

            // Check if the form is valid
```

```
    if (!form.checkValidity()) {
      // Form is invalid - do something like show a message
      return;
    }

    // Custom validation for password length
    const password = document.getElementById('pwd');
    if (password.value.length < 8) {
      password.setCustomValidity('Password must be at least 8 characters long.');
      password.reportValidity();
      return;
    } else {
      password.setCustomValidity(''); // Clear custom validity message
    }

    // Check if passwords match
    const pwdConf = document.getElementById('pwdConf');
    if (password.value !== pwdConf.value) {
      pwdConf.setCustomValidity('Passwords do not match.');
      pwdConf.reportValidity();
      return;
    } else {
      pwdConf.setCustomValidity(''); // Clear custom validity message
    }

    // Form is valid, you can proceed with form submission or further processing
    form.submit();
  });

  // Add input event listeners to reset custom validity messages when the user corrects them
  document.querySelectorAll('input[type="password"]').forEach(function (input) {
    input.addEventListener('input', function () {
      input.setCustomValidity('');
    });
  });
});

const password = document.querySelector("#pwd");
const passwordConf = document.querySelector("#pwdConf");
const eyeIcon = document.querySelector("#eye");
const eyeIcon2 = document.querySelector("#eye2");
```

```
  const togglePassword = document.querySelector("#togglePassword");
  const togglePassword2 = document.querySelector("#togglePassword2");

  togglePassword.addEventListener("click", () => {
    if (eyeIcon.classList.contains("bi-eye")) {
      password.setAttribute("type", "text");
      eyeIcon.classList.replace("bi-eye", "bi-eye-slash");
    } else {
      password.setAttribute("type", "password");
      eyeIcon.classList.replace("bi-eye-slash", "bi-eye");
    }
  });

  togglePassword2.addEventListener("click", () => {
    if (eyeIcon2.classList.contains("bi-eye")) {
      passwordConf.setAttribute("type", "text");
      eyeIcon2.classList.replace("bi-eye", "bi-eye-slash");
    } else {
      passwordConf.setAttribute("type", "password");
      eyeIcon2.classList.replace("bi-eye-slash", "bi-eye");
    }
  });

</script>
```

Fix:

Handling input validity

```
input.addEventListener('input', function () {
  input.setCustomValidity('');
});
```

Handling password validity

```
password.setCustomValidity('Password must be at least 8 characters long.');
```

3. Missing Anti-clickjacking Header

App.js

```
<script>
   const password = document.querySelector("#pwd");
   const eyeIcon = document.querySelector("#eye");
   const togglePassword = document.querySelector("#togglePassword");

   togglePassword.addEventListener("click", () => {
      if (eyeIcon.classList.contains("bi-eye")) {
         password.setAttribute("type", "text");
         eyeIcon.classList.replace("bi-eye", "bi-eye-slash");
      } else {
         password.setAttribute("type", "password");
         eyeIcon.classList.replace("bi-eye-slash", "bi-eye");
      }
   });
</script>
```

Fix:

App.js

```
app.use(function(req, res, next) {
  res.setHeader('X-Frame-Options', 'sameorigin');
  next();
});
```

User-layout.ejs

```
<style>
     #vulnerable_website {
       position:relative;
       opacity:0.3;     //👈 changed code
       width:800px;
       height:900px;
       top:75px;
       left: -95px;
       z-index:2;
       padding-left:80px;
       }
```

```
  #attacker_website {
    position:absolute;
    z-index:1;
    }
  #attacker_website button {
    margin-left:100px;
  }
</style>
```