Universidad de las Fuerzas Armadas ESPE

Departamento de Ciencias de la Computación

Ingeniería de Software

Integrantes: Mathias Guevara, Christopher Iza, Shared Tinoco.

Materia: Desarrollo de Software Seguro

Docente: Diego Saavedra

Fecha de entrega: 27 de julio de 2024

Semestre: Séptimo Semestre

NRC: 14943

Medidas Implementadas

Para realizar la auditoría del sistema de Blog Educativo se tuvo como referencia el listado de prácticas de codificación segura propuestas por OWASP en las cuales encontramos puntos clave que permitirán corregir vulnerabilidades comunes en sistemas informáticos.

Validación de entradas:

■ Verificar si todas las entradas del usuario son validadas correctamente para prevenir ataques de inyección.

```
const handleChange = (e) => {
  const { name, value } = e.target;
  const sanitizedValue = DOMPurify.sanitize(value);

  setInputs((prev) => ({ ...prev, [name]: sanitizedValue }));
};
const handleSubmit = async (e) => {
  e.preventDefault();
  const { username, password } = inputs;
  console.log(password);

  try {
    console.log(inputs);
    if (
      !/^[a-zA-Z0-9]*$/.test(username) &&
      !/^[a-zA-Z0-9]*$/.test(password)
    ) {
      console.log("aló");
      setError("Special characters are not allowed.");
      setTimeout(() => setError(null), 3000);

      return;
    }
    await login(inputs);
```

Para validar las entradas se implementaron medidas de sanitización de ingresos y verificación de caracteres, como se puede ver en la imagen las entradas son sanitizadas cada que existe un cambio en los campos de entrada y antes de hacer el llamado al backend.

Codificación de salidas:

■ Comprobar si las salidas de datos están correctamente codificadas para prevenir ataques de XSS.

```
</div>
<h1>{post.title}</h1>
<p
  dangerouslySetInnerHTML={{
    __html: DOMPurify.sanitize(post.desc),
  }}
></p>
```

Para evitar que se ejecuten comandos de javascript a través del renderizado del html en donde al igual que en las entradas se implementó la librería de DOMpurify.

Autenticación y gestión de contraseñas:

■ Evaluar la implementación de los mecanismos de autenticación y la gestión segura de contraseñas.

```
const secret = speakeasy.generateSecret({ name: "Blog Grupo 8" });        You, 4 hours

qrcode.toDataURL(secret.otpauth_url, (err, data_url) => {
  if (err) return res.status(500).json(err);

  const q =
    "INSERT INTO users(`username`,`email`,`password`, `2fa_secret`) VALUES (?)";
  const values = [req.body.username, req.body.email, hash, secret.base32];
```

```
if (!isPasswordCorrect)
  return res.status(400).json("Usuario o Contraseña incorrecta!");

const { token } = req.body;
const isTokenValid = speakeasy.totp.verify({
  secret: data[0]["2fa_secret"],
  encoding: "base32",
  token: token,
});
```

En este sistema se implemento la doble autenticación de Google en la cuál a través de un código QR generado cuando se realiza el registro de un nuevo usuario.

```
const isPasswordCorrect = bcrypt.compareSync(
  req.body.password,
  data[0].password
);

if (!isPasswordCorrect)
  return res.status(400).json("Usuario o Contraseña incorrecta!");
```

En cuanto a gestión de contraseñas se realiza una comparación de las contraseñas hasheadas en la base de datos.

Administración de sesiones:

■ Analizar cómo se manejan las sesiones de los usuarios y si existen medidas de seguridad para protegerlas.

```
    const jwtToken = jwt.sign({ id: data[0].id }, process.env.JWT_SECRET, {
      expiresIn: "8h",
    });
    const { password, ...other } = data[0];

    res
      .cookie("access_token", jwtToken, {
        httpOnly: true,
        sameSite: "strict",
        secure: true,
      })
      .status(200)
      .json(other);
  });
```

La administración de sesiones se lo realizó a través JWT y el uso de cookies para guardar la misma, la configuración de JWT se configuro para que el token generado al iniciar sesión tenga fecha de caducidad.

Control de Acceso:

■ Revisar los controles de acceso y asegurarse de que están correctamente implementados para proteger los recursos.

```
const { token } = req.body;
const isTokenValid = speakeasy.totp.verify({
  secret: data[0]["2fa_secret"],      You, 4 hours ago • Configuración ssl,
  encoding: "base32",
  token: token,
});

if (!isTokenValid) return res.status(400).json("Token 2FA incorrecto!");
```

El control de acceso se lo hace validando la generación del segundo factor de autenticación.

Prácticas Criptográficas:

■ Verificar el uso de técnicas criptográficas para proteger datos sensibles.

```
const salt = bcrypt.genSaltSync(10);
const hash = bcrypt.hashSync(req.body.password, salt);
```

```
const isPasswordCorrect = bcrypt.compareSync(
  req.body.password,
  data[0].password
);
```

Se implementó la librería bcrypt para convertir la contraseña en hash utilizando salts aleatorios los que aumentan la robustes contra ataques de fuerza bruta

Manejo de errores y Logs:

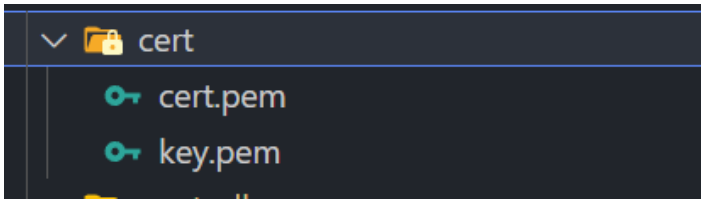
■ Evaluar cómo se gestionan los errores y los registros, asegurándose de que no se expone información sensible.

```
if (err) return res.status(500).json(err);
if (data.length) return res.status(409).json("Usuario ya existe!");

const salt = bcrypt.genSaltSync(10);
const hash = bcrypt.hashSync(req.body.password, salt);

const secret = speakeasy.generateSecret({ name: "Blog Grupo 8" });

qrcode.toDataURL(secret.otpauth_url, (err, data_url) => {
  if (err) return res.status(500).json(err);

  const q =
    "INSERT INTO users(`username`,`email`,`password`, `2fa_secret`) VALUES (?)";
  const values = [req.body.username, req.body.email, hash, secret.base32];

  db.query(q, [values], (err, data) => {
    if (err) return res.status(500).json(err);

    res.status(200).json({
      message: "Se creo el usuario",
      qr_code: data_url,
    });
  });
});
```

Se implementaron respuestas en formato de estados http en conjunto con mensajes descriptivos para realizar un seguimiento de los errores y los resultados de las peticiones implementadas.

Seguridad en las comunicaciones:

■ Comprobar si las comunicaciones entre componentes están aseguradas, por ejemplo, utilizando HTTPS.



Se crearon certificados para que se pudiera realizar la adecuada implementación de los protocolos https.

```
// Configuración de HTTPS
const options = {
  key: fs.readFileSync("./cert/key.pem", "utf8"),
  cert: fs.readFileSync("./cert/cert.pem", "utf8"),
};

const httpsServer = https.createServer(options, app);

httpsServer.listen(8800, () => {
  console.log("Conectado puerto 8800!");
});
```

Utilizando las librerías fs y https de node se utilizaron los certificados para crear un servidor con el protocolo seguro.

Configuración de los sistemas:

■ Evaluar la configuración del sistema para asegurar que no se dejan configuraciones inseguras por defecto.

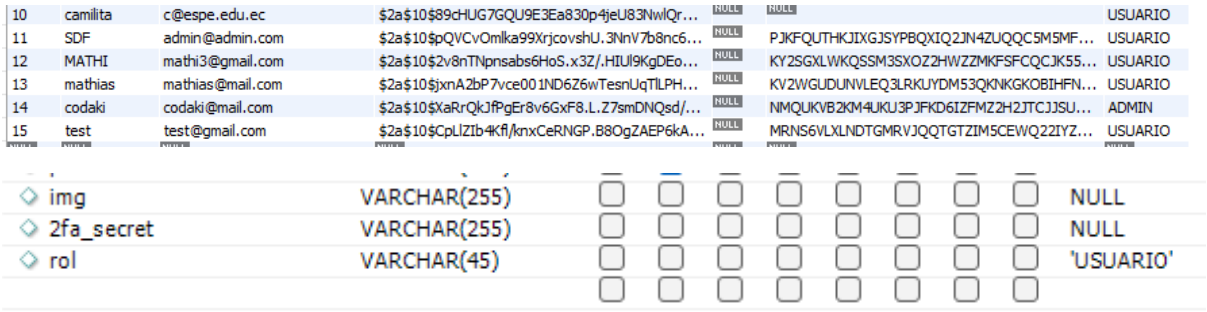
```
app.use(helmet());
app.use(
  helmet.contentSecurityPolicy({
    directives: {
      defaultSrc: ["'self'"],
      scriptSrc: ["'self'", "'unsafe-inline'", "example.com"],
      styleSrc: ["'self'", "'unsafe-inline'"],
      imgSrc: ["'self'", "data:", "example.com"],
      connectSrc: ["'self'"],
      fontSrc: ["'self'"],
      objectSrc: ["'none'"],
      upgradeInsecureRequests: [],
    },
  })
);
app.use(helmet.frameguard({ action: "deny" }));
app.use(helmet.xssFilter());
app.use(helmet.noSniff());
app.use(
  helmet.hsts({ maxAge: 63072000, includeSubDomains: true, preload: true })
);

// Eliminar cabeceras innecesarias para ocultar la tecnología del servidor
app.use((req, res, next) => {
  res.removeHeader("X-Powered-By");
  next();
});

const limiter = rateLimit({
  windowMs: 15 * 60 * 1000,       You, 37 minutes ago • Uncommitted changes
  max: 100,
  message:
    "Demasiadas peticiones desde esta IP, por favor intenta de nuevo después de 15 minutos.",
});
app.use("/api/", limiter);
```

Dentro de las configuraciones del servidor se implemetaron varias medidas para protegerse contra diversos ataques. Entre estas medidas tenemos limitadores de peticiones por IP o

protecciones de ataques a cabeceras, así como protección contra herramientas de sniffing en conjunto con la limitación de origen de peticiones.

Seguridad de Base de Datos:

■ Revisar las prácticas de seguridad implementadas en la base de datos, incluyendo el uso de cuentas de servicio y privilegios mínimos.



10	camilita	c@espe.edu.ec	$2a$10$89cHUG7GQU9E3Ea830p4jeU83NwlQr...	NULL	NULL	USUARIO
11	SDF	admin@admin.com	$2a$10$pQVCvOmlka99XrjcovshU.3NnV7b8nc6...	NULL	PJKFQUTHKJIXGJSYPBQXIQ2JN4ZUQQC5M5MF...	USUARIO
12	MATHI	mathi3@gmail.com	$2a$10$2v8nTNpnsabs6HoS.x3Z/.HIUl9KgDEo...	NULL	KY2SGXLWKQSSM3SXOZ2HWZZMKFSFCQCJK55...	USUARIO
13	mathias	mathias@mail.com	$2a$10$jxnA2bP7vce001ND6Z6wTesnUqTlLPH...	NULL	KV2WGUDUNVLEQ3LRKUYDM53QKNKGKOBIHFN...	USUARIO
14	codaki	codaki@mail.com	$2a$10$XaRrQkJfPgEr8v6GxF8.L.Z7smDNQsd/...	NULL	NMQUKVB2KM4UKU3PJFKD6IZFMZ2H2JTCJJSU...	ADMIN
15	test	test@gmail.com	$2a$10$CpLlZIb4Kfl/knxCeRNGP.B8OgZAEP6kA...	NULL	MRNS6VLXLNDTGMRVJQQTGTZIM5CEWQ22IYZ...	USUARIO

img	VARCHAR(255)								NULL
2fa_secret	VARCHAR(255)								NULL
rol	VARCHAR(45)								'USUARIO'

En la base de datos todos los usuarios tienen los privilegios mínimos los cuales solo les permiten crear posts o a su vez, modificar y eliminar posts que ellos hayan creado. El rol de administrador es solo asignado de manera manual dentro de la base de datos.

Manejo de Archivos:

■ Verificar cómo se gestionan los archivos, asegurándose de que se previenen ataques como la inclusión de archivos.

```
<input
  style={{ display: "none" }}
  type="file"
  id="file"
  name=""
  accept=".png"
  onChange={(e) => {
    const selectedFile = e.target.files[0];
    if (selectedFile && selectedFile.type !== "image/png") {
      alert("Solo se aceptan archivos en formato .png .");
      e.target.value = null;
      setFile(null);
    } else {
      setFile(selectedFile);
    }
  }}
/>
<label className="file" htmlFor="file">
  Subir Archivo
</label>
{errors.file && <span className="error">{errors.file}</span>}
```

Para el manejo de archivos se delimitó y comprobó que los archivos permitidos dentro del sistema solo sean de formato .png, al intentar subir un archivo diferente se establecerá una notificación especificando el error.