UNIVERSIDADE ESTÁCIO DE SÁ
Polo Nova Ourinhos - Ourinhos/SP
Tecnologia em Desenvolvimento Full Stack
RPG0035 - Software sem Segurança não Serve

Aluno: Rogério Benedito Geraldo Matrícula: 202110027921.

Github: https://github.com/Rogeriobg/MPN5M5-RPG0035.git

RELATÓRIO DA MISSÃO PRÁTICA

1 INTRODUÇÃO

Através dessa atividade o aluno analisará uma falha de segurança, em uma
aplicação web, e aplicará as medidas corretivas necessárias para garantir o seu correto e seguro funcionamento.

2 CONTEXTUALIZAÇÃO

O time de segurança da Software House, onde você atua como Especialista
em Desenvolvimento de Software, identificou uma falha de segurança, explorada por ataques que geraram o vazamento de dados, além de outros
problemas, em uma das aplicações legadas, desenvolvida há alguns anos atrás. Tal falha consiste na concessão de acesso não autorizado de recursos a usuários. O cenário completo é descrito a seguir:
empresa detentora do software.
Logo, tal falha é passível de ser explorada via ataques de força bruta para descoberta do padrão usado na geração da “session-id” e consequente geração de valores aleatórios que serão usados para a realização de requisições – como solicitações de dados e também criação e atualização na aplicação, até a obtenção do acesso indevido.
Além do problema já relatado, o time de segurança descobriu que, atualmente, não é realizado nenhum tratamento no processamento dos parâmetros trafegados na aplicação. Logo, também é possível explorar outras falhas, como as de “Injection” de códigos maliciosos.
Frente ao exposto, seu trabalho consistirá em refatorar a aplicação,

conforme procedimentos descritos a seguir.

3 PROCEDIMENTOS

Abra o código-fonte fornecido acima na IDE ou editor;
Refatore o método de criptografia utilizado atualmente, substituindo a geração do “session-id” por um outro mecanismo de segurança, como tokens JWT;
Refatore a arquitetura da API, para que o token (atualmente representado pelo “session-id”) não seja trafegado via URI, mas através do header da requisição;
A cada requisição recebida pela API, valide o token de segurança, incluindo a identidade do usuário, data/hora de expiração do mesmo, etc.;
Inclua, em todos os endpoints, controle de acesso a recursos baseado no perfil do usuário. Garante que, à exceção do endpoint de login, todos os demais sejam acessados apenas por usuários com perfil ‘admin’;
Para testar a implementação do item anterior, crie um novo endpoint que permita a recuperação dos dados do usuário logado. Tal método não deverá conter o controle de acesso limitado ao perfil ‘admin’;
Refatore o método que realiza a busca de contratos no banco, tratando os parâmetros recebidos contra vulnerabilidades do tipo “Injection”. Para isso você poderá utilizar bibliotecas de terceiros, expressões regulares ou outro mecanismo que garanta o sucesso do processo em questão;
Salve o código e coloque a API para ser executada;
Utilizando um cliente (Insomnia, Postman ou outro de sua preferência), realize testes na API, garantindo que todos os pontos acima foram tratados.

5 CÓDIGO UTILIZADO

```
const express = require('express');
const bodyParser = require('body-parser');
const cors = require('cors');
const jwt = require('jsonwebtoken');
const bcrypt = require('bcrypt');
const { body, validationResult } = require('express-validator');

const app = express();
const port = process.env.PORT || 3000;
const SECRET_KEY = 'chave_super_secreta';

app.use(cors());
app.use(bodyParser.json());
```

```
const users = [
  {
    username: "user",
    password: bcrypt.hashSync("123456", 10),
    id: 123,
    email: "user@dominio.com",
    perfil: "user"
  },
  {
    username: "admin",
    password: bcrypt.hashSync("123456789", 10),
    id: 124,
    email: "admin@dominio.com",
    perfil: "admin"
  },
  {
    username: "colab",
    password: bcrypt.hashSync("123", 10),
    id: 125,
    email: "colab@dominio.com",
    perfil: "user"
  }
];


function sanitize(input) {
  if (typeof input !== 'string') return '';
  return input.replace(/[^a-zA-Z0-9-_@.]/g, '');
}


function doLogin(credentials) {
  return users.find(u => u.username === credentials.username);
}


function generateToken(user) {
  const payload = {
    id: user.id,
    perfil: user.perfil,
    email: user.email
  };
  return jwt.sign(payload, SECRET_KEY, { expiresIn: '1h' });
}


function authenticateToken(req, res, next) {
  const authHeader = req.headers['authorization'];
  const token = authHeader?.split(' ')[1];
```

```
  if (!token) return res.status(401).json({ message: 'Token não fornecido'
});

  jwt.verify(token, SECRET_KEY, (err, user) => {
    if (err) return res.status(403).json({ message: 'Token inválido' });
    req.user = user;
    next();
  });
}


function onlyAdmin(req, res, next) {
  if (req.user.perfil !== 'admin') {
    return res.status(403).json({ message: 'Acesso restrito a
administradores' });
  }
  next();
}


class Repository {
  execute(query) {
    console.log("Query executada (fake):", query);

    return [
      {
        contrato_id: 1,
        empresa: 'AcmeCorp',
        data_inicio: '2024-01-01',
        descricao: 'Contrato de prestação de serviços'
      }
    ];
  }
}


function getContracts(empresa, inicio) {
  const sanitizedEmpresa = sanitize(empresa);
  const sanitizedInicio = sanitize(inicio);
  const query = `SELECT * FROM contracts WHERE empresa =
'${sanitizedEmpresa}' AND data_inicio = '${sanitizedInicio}'`;
  const repository = new Repository();
  return repository.execute(query);
}


app.post('/api/auth/login',
  body('username').notEmpty().withMessage('Usuário obrigatório'),
  body('password').notEmpty().withMessage('Senha obrigatória'),
  (req, res) => {
    const errors = validationResult(req);
```

```
    if (!errors.isEmpty()) return res.status(400).json({ errors:
errors.array() });

    const credentials = req.body;
    const user = doLogin(credentials);
    if (!user) return res.status(401).json({ message: 'Credenciais inválidas'
});

    const passwordValid = bcrypt.compareSync(credentials.password,
user.password);
    if (!passwordValid) return res.status(401).json({ message: 'Credenciais
inválidas' });

    const token = generateToken(user);
    res.json({ token });
  }
);


app.get('/api/auth/profile', authenticateToken, (req, res) => {
  res.status(200).json({ user: req.user });
});


app.get('/api/users', authenticateToken, onlyAdmin, (req, res) => {
  res.status(200).json({ data: users });
});


app.get('/api/contracts/:empresa/:inicio', authenticateToken, onlyAdmin,
(req, res) => {
  const { empresa, inicio } = req.params;
  const result = getContracts(empresa, inicio);
  if (result.length > 0)
    res.status(200).json({ data: result });
  else
    res.status(404).json({ message: 'Dados não encontrados' });
});


app.listen(port, () => {
  console.log(`Servidor rodando na porta ${port}`);
});
```