Missão Prática | Nível 5 | Mundo 5

João Gabriel Cesconetto - 202208324053

Software sem segurança não serve

Objetivos da prática:

1) **Descrever o controle básico de acesso a uma API REST;**
2) **Descrever o tratamento de dados sensíveis e log de erros com foco em segurança;**
3) **Descrever a prevenção de ataques de acesso não autorizado com base em tokens desprotegidos/desatualizados;**
4) **Descrever o tratamento de SQL Injection em códigos-fonte; descrever o tratamento de CRLF Injection em códigos-fonte;**
5) **Descrever a prevenção de ataques do tipo CSRF em sistemas web.**

Missão Prática

1) Refatoração do método de criptografia para utilização de tokens JWT:

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

  if (!token) {
    return res.status(401).json({ message: 'Acesso não autorizado' })
  }

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

2) Tráfego do token pelo header da requisição, ao invés da URI:

```
app.post('/api/auth/login', (req, res) => {
  const credentials = req.body
  const userData = doLogin(credentials)

  if (userData) {
    const user = { name: userData.username }
    const accessToken = jwt.sign(user, SECRET_KEY, { expiresIn: 60 * 60 })
    return res.json({ accessToken })
  } else {
    return res.status(400, json({ message: 'Informe o usuário e senha' }))
  }
})
```

3) Validação do token a cada requisição recebida, além de controle de acesso a recursos baseado em perfil:

```
app.get('/api/users', authenticateToken, (req, res) => {
  const userPerfil = req.user

  if (userPerfil !== 'admin') {
    return res.status(403).json({ message: 'Acesso não autorizado' })
  }

  res.status(200).json({ data: users })
})

app.get('/api/current-user', authenticateToken, (req, res) => {
  const body = req.body
  const user = users.find(item => item.id === body.id)

  if (user) res.status(200).json({ user })
  else res.status(401).json({ message: 'Nenhum usuário encontrado' })
})

app.get('api/contracts/:empresa/:inicio', authenticateToken, (req, res) => {
  const userPerfil = req.user

  if (userPerfil !== 'admin') {
    return res.status(403).json({ message: 'Acesso não autorizado' })
  }

  const empresa = req.params.empresa
  const dataInicio = req.params.inicio
  const result = getContracts(empresa, dataInicio)

  if (result) res.status(200).json({ data: result })
  else res.status(404).json({ data: 'Dados não encontrados' })
})
```

4) Tratamento de vulnerabilidades, como SQL Injection, em método de busca de contratos através do escape de caracteres indevidos:

```
function escapeSQL(param) {
  return param.replace(/'/g, "\\'").replace(/"/g, '\\"')
}

function getContracts(empresa, inicio) {
  const query = `SELECT * FROM contracts WHERE empresa = ${escapeSQL(empresa)} AND data_inicio = ${escapeSQL(inicio)} `
  const result = repository.execute(query)
  return result
}
```

```
app.get('api/contracts/:empresa/:inicio', authenticateToken, (req, res) => {
  const userPerfil = req.user

  if (userPerfil !== 'admin') {
    return res.status(403).json({ message: 'Acesso não autorizado' })
  }

  const empresa = req.params.empresa
  const dataInicio = req.params.inicio
  const result = getContracts(empresa, dataInicio)

  if (result) res.status(200).json({ data: result })
  else res.status(404).json({ data: 'Dados não encontrados' })
})
```

Resultados exibidos no Postman: