**Missão Prática – Mundo 05 – Nível 05**

**Gilvan Pereira de Oliveira – 2023.01.53256-61197**

**Polo Centro – São Lourenço Da Mata – PE**

**RPG0034 – SOFTWARE SEM SEGURANÇA NÃO SERVE! – 9001 – 2025.1**

<https://github.com/GilvanPOliveira/FullStack/tree/main/Mundo05/softwareSeguranca>

**Micro atividade 01: Análise e Correção de Vulnerabilidades em Código**

O código utiliza o framework Express para criar um servidor HTTP que processa requisições:

Texto

O conteúdo gerado por IA pode estar incorreto.

* Ele define duas rotas, raiz (/) e (/acessoRestrito), que requer autenticação para acessar;
* Tratamento de Segurança com Autenticação, incluindo um middleware de autenticação que valida o acesso à rota /acessoRestrito. Ele verifica se o cabeçalho “autorizar” contém o valor "firewall";
* Retornar Mensagem Genérica e Status 401 em Caso de Acesso Não Autorizado, se o token de autenticação estiver ausente ou for inválido, o middleware retorna uma mensagem genérica ("Não autorizado") e define o status HTTP como 401 Unauthorized;
* Enviar Resposta da Requisição em Caso de Autenticação Bem-Sucedida, se o token de autenticação for válido, o middleware chama next(), permitindo que a requisição prossiga para a rota /acessoRestrito. A rota então retorna uma resposta JSON com a mensagem "Acesso permitido!".

**Micro atividade 2: Descrever o tratamento de dados sensíveis e log de erros com foco em segurança**

Texto

O conteúdo gerado por IA pode estar incorreto.

* Verificação de usuário existente:
  + Checagem para evitar duplicidade de nomes de usuário no cadastro.
* Verificação da quantidade mínima de caracteres:
  + Foi definido um tamanho mínimo (8 caracteres) e o código verifica se a senha atende a esse requisito.
* Permissão de quaisquer caracteres na senha:
  + Removida a restrição, no código original, que limitava a senha apenas a caracteres numéricos.
* Limitação de tentativas inválidas:
  + Antes de validar as credenciais, é verificado se o número de tentativas inválidas já excedeu o limite permitido. Caso seja, o código retornará uma mensagem informando que o usuário atingiu o número máximo de tentativas.
* Mensagem de erro genérica:
  + Ao validar as credenciais, se houver qualquer erro, o código retorna a mesma mensagem: “Usuário ou senha incorretos”, sem informar detalhes específicos.

**Micro atividade 3: Descrever a prevenção de ataques de acesso não autorizado com base em tokens desprotegidos/desatualizados**

Backend (app.js):  
Texto

O conteúdo gerado por IA pode estar incorreto.

Frontend (index.html):

Texto

O conteúdo gerado por IA pode estar incorreto.  
  
Styles:

Texto

O conteúdo gerado por IA pode estar incorreto.

Frontend (script.js):Tela de computador com texto preto sobre fundo branco

O conteúdo gerado por IA pode estar incorreto.

Backend (app.js):

1. Geração do Token com Claim “exp”:

* O endpoint /autenticar valida as credenciais (neste caso, usuário e senha "admin");
* Em caso de sucesso, calcula o timestamp de expiração (exp) e inclui-o no payload do token JWT (determinado por 3600s, ou seja, 1 hora);
* O token e o timestamp de expiração são retornados no JSON de resposta.

1. Validação do Token e Expiração:

* endpoint /acessoRestrito extrai o token do cabeçalho "autorizar", remove o prefixo "firewall " e utiliza verify (token, chaveSecreta) para validar o token;
* Caso a validação falhe, o bloco catch retorna uma mensagem genérica, atendendo ao requisito de não revelar detalhes sobre a causa da falha.

1. Organização dos Arquivos Estáticos:

* Todos os arquivos estáticos (como CSS e JS) estão na pasta componentes;
* O index.html é servido via res.sendFile, e residi na raiz do projeto.

1. Validação de Credenciais:

* A função validateUser compara os valores com "admin". Essa verificação simples é suficiente para o que foi solicitado nesta micro atividade.

Frontend (index.html e script.js):

1. Recebimento e Armazenamento do Token:
   * Ao efetuar o login, a requisição POST para /autenticar armazena tanto o jwt\_token quanto o exp no localStorage.
2. Validação do Token Antes da Requisição:
   * Antes de chamar o endpoint /acessoRestrito, o script verifica se o token existe e se o timestamp atual (em segundos, 3600) é menor que o valor armazenado;
   * Assim, se o token estiver expirado, a requisição não é realizada e uma mensagem de erro é exibida.
3. Exibição de Mensagens e Logoff:
   * A função showMessage é utilizada para exibir feedback na página;
   * O botão "Logoff" remove os itens "token" e "tokenExp" do localStorage, deslogando o usuário, já que ao logar a credencial permanece ativa por 1 hora.

Requisitos solicitados:

* **Backend:**
  + Gera um token JWT com o claim “exp”;
  + Valida o token e sua expiração, retornando um erro genérico se o token for inválido ou expirado;
  + As credenciais são validadas e o token é retornado ao frontend.
* **Frontend:**
  + Armazena o token e o timestamp de expiração no localStorage;
  + Verifica a validade do token antes de realizar uma requisição ao backend;
  + Possui botões para login, acesso restrito e logoff, com exibição adequada de mensagens.

**Micro atividade 4: Descrever o tratamento de SQL Injection em códigos-fonte**

A principal falha encontrada, é que o parâmetro id é concatenado diretamente na query, permitindo que um invasor injete comandos maliciosos. Para evitar esse problema, é recomendado usar declarações preparadas ou queries parametrizadas, que tratam os parâmetros de forma segura.

Segue um exemplo utilizando o módulo mysql2, que suporta prepared statements:

Texto

O conteúdo gerado por IA pode estar incorreto.

* Prepared Statements (declarações preparadas):
  + A query SQL foi modificada para usar um (?) no lugar da concatenação direta do valor de id;
  + O valor de id é passado como um parâmetro seguro para a função connection.execute.
  + Parameterized Queries (queries parametrizadas):
    - O método execute do mysql2 garante que o valor de id seja tratado como um parâmetro seguro, evitando que eleseja interpretado como parte da query SQL.
    - Segurança:
      * Mesmo que um atacante tente enviar um valor malicioso como ' OR '1'='1, ele será tratado como um valor literal e não como parte da query SQL.

**Micro atividade 5: Descrever o tratamento de CRLF Injection em códigos-fonte**

Texto

O conteúdo gerado por IA pode estar incorreto.

1. Recebimento da URL: A rota /redirecionar captura o parâmetro url enviado via query string. Por exemplo, na URL http://dominio.com/redirect?url=/pagina, o valor /pagina é extraído como a URL de destino;
2. Sanitização: Antes de qualquer processamento, o código remove todos os caracteres de quebra de linha (CR – \r e LF – \n) utilizando redirectUrl.replace(/[\r\n]/g, ''). Essa etapa é crucial para evitar a injeção de cabeçalhos HTTP adicionais por meio de caracteres maliciosos;
3. Validação do Domínio:
   * O construtor new URL(redirectUrl, base) é empregado para criar um objeto URL. Se o valor fornecido for uma URL relativa, ele será resolvido com base no domínio permitido (que foi denominada: dominio.com);
   * Em seguida, é verificado se o hostname da URL resultante é exatamente igual ao domínio autorizado (definido em allowedDomain). Se não for, o redirecionamento é impedido, evitando que o usuário seja direcionado para um domínio externo não autorizado;
4. Redirecionamento Seguro: Se a URL estiver devidamente sanitizada e validada (ou seja, pertence ao domínio permitido ou é uma URL relativa que se resolve para ele), o código executa o redirecionamento chamando res.redirect(redirectUrl). Dessa forma, o redirecionamento ocorre somente para destinos seguros.