UNIVERSIDADE POLARIS

Instituto de Segurança da Informação

Análise da aplicação - OWASP

Colaboradores: Davih Gonçalves Duque, Fabiana Santos Soares, Filipe Acacio Costa, Leonardo Guedes Gomes Junior, Vítor César Reis Francisco, Lucas de Oliveira Fonseca

Belo Horizonte — Minas Gerais

2025

1 INTRODUÇÃO

O sistema backend analisado é responsável pelo gerenciamento de informações dos usuários da Universidade Polaris, operando como um CRUD básico que inclui funcionalidades de cadastro, leitura, atualização e exclusão de dados. Entre suas principais rotinas, destacam-se o processo de login/sign-in e a área “Meu Perfil”, na qual o usuário pode visualizar e editar suas próprias informações ou solicitar sua exclusão.

Apesar de sua estrutura simples, o sistema lida com dados pessoais e credenciais sensíveis, o que demanda práticas adequadas de segurança para evitar acessos não autorizados, manipulação indevida de informações ou falhas que possam comprometer a integridade, a confidencialidade e a disponibilidade do serviço. Assim, torna-se necessário identificar potenciais vulnerabilidades e propor medidas que reforcem a proteção da aplicação e de seus usuários.

2 ANÁLISE OWASP

A01:2021 - Broken Access Control (Falhas de Controle de Acesso)

Falhas de controle de acesso surgem quando o sistema não restringe adequadamente as ações que cada usuário pode executar. Em um backend de CRUD como o da Universidade Polaris, isso ocorre quando endpoints permitem que um usuário acesse ou manipule dados que não lhe pertencem, como editar perfis de outras pessoas, acessar registros administrativos ou consultar informações restritas. O risco aumenta especialmente quando o backend presume que o frontend fará a filtragem, permitindo que um atacante manipule requisições e explore rotas vulneráveis.

Se o sistema não valida a propriedade dos recursos — por exemplo, aceitando qualquer ID enviado na URL ou no corpo da requisição — um usuário autenticado pode assumir o controle de dados de terceiros. Esse problema é crítico porque não depende de falhas complexas de invasão: basta um atacante modificar parâmetros, repetir requisições ou explorar endpoints desprotegidos para comprometer a confidencialidade e a integridade das informações.

Medidas de mitigação

- Validar no backend a propriedade dos recursos acessados (userID da sessão vs. userID da requisição).
- Restringir rotas sensíveis a perfis específicos, com regras explícitas de autorização.
- Aplicar o princípio do menor privilégio em todos os níveis do sistema.
- Bloquear o acesso direto a IDs numéricos previsíveis (preferir UUIDs).
- Registrar e auditar tentativas de acesso indevido.
- Evitar qualquer lógica de autorização no frontend sem validação equivalente no backend.

A02 - Cryptographic Failures (Falhas Criptográficas)

Falhas criptográficas ocorrem quando dados sensíveis não são protegidos de maneira adequada, seja por ausência de criptografia, uso de algoritmos obsoletos ou manejo incorreto de chaves e senhas. No backend da Universidade Polaris, que armazena informações pessoais e realiza autenticação de usuários, essas falhas podem expor credenciais, permitir acesso indevido ou comprometer a integridade dos dados armazenados e transmitidos.

O risco aumenta quando senhas são salvas em texto puro, quando conexões não utilizam HTTPS ou quando tokens são gerados sem assinatura ou com chaves previsíveis. Situações como essas permitem que atacantes interceptem dados, modifiquem informações ou até forjem sessões válidas. Em um ambiente acadêmico, isso pode resultar em invasão de perfis, alterações de informações pessoais ou exposição de dados institucionais.

Medidas de mitigação

- Armazenar senhas usando **hash seguro** (bcrypt, Argon2id, PBKDF2) com salt.
- Assinar e criptografar tokens, garantindo entropia adequada nas chaves.
- Rotacionar periodicamente chaves criptográficas sensíveis.
- Evitar algoritmos obsoletos (MD5, SHA-1, DES, RC4).
- Criptografar dados sensíveis em repouso quando aplicável.

A05 - Security Misconfiguration (Configurações de Segurança Incorretas)

Configurações de segurança inadequadas acontecem quando servidores, serviços ou frameworks são implantados com padrões frágeis, opções desnecessárias habilitadas ou componentes não revisados. Em um backend como o da Universidade Polaris, isso inclui APIs expostas além do necessário, permissões de diretórios incorretas, mensagens de erro que revelam detalhes internos, serviços de depuração habilitados em produção ou configurações inseguras de CORS e cabeçalhos HTTP.

Esses problemas não dependem de código vulnerável — muitas vezes surgem apenas porque o ambiente foi configurado sem critérios. Uma porta aberta, um banco sem restrição de origem, um stack trace exposto ou regras de firewall permissivas já são suficientes para comprometer a aplicação. Assim, a segurança depende não apenas do código, mas também da forma como o sistema e sua infraestrutura são configurados ao longo do ciclo de vida.

Medidas de Mitigação

- Desabilitar serviços, módulos e portas que não são necessários em produção.
- Configurar CORS de forma restritiva, aceitando apenas origens legítimas.
- Ocultar mensagens de erro detalhadas em ambientes públicos.
- Utilizar cabeçalhos de segurança adequados (X-Content-Type-Options, X-Frame-Options, Content-Security-Policy etc.).

A07:2021 – Identification and Authentication Failures (Identificação e Autenticação Quebradas)

Falhas de identificação e autenticação acontecem quando o sistema não consegue confirmar adequadamente quem é o usuário que está tentando acessar a aplicação ou quando os mecanismos usados para essa verificação são frágeis. No backend da Universidade Polaris, isso afeta diretamente processos como login, recuperação de senha e manutenção de sessões. Senhas fracas, ausência de MFA, tokens mal gerados, sessões que não expiram ou fluxos que permitem brute force podem tornar a autenticação facilmente burlável.

Essas falhas se tornam especialmente críticas porque representam a porta de entrada para quase todos os outros ataques. Se a autenticação permitir que usuários mal-intencionados assumam identidades legítimas, isso pode resultar em invasão de perfis, alteração de dados pessoais, exclusão indevida de registros ou obtenção de informações privadas. Mesmo que o controle de acesso e as configurações de segurança estejam corretas, uma autenticação fraca abre caminho para comprometimento total do sistema.

Medidas de Mitigação

- Imposição de políticas de senha forte e armazenamento seguro usando hash moderno (bcrypt, Argon2id, PBKDF2).
- Bloqueio temporário ou rate limit após várias tentativas de login.
- Geração de tokens imprevisíveis e assinados, com expiração adequada.
- Invalidação imediata de tokens após logout ou troca de senha.
- Configuração correta do ciclo de vida de sessões, com expiração automática após inatividade.

3. Configuração da Infraestrutura e Arquitetura de Sistemas

A infraestrutura do sistema Universidade Polaris foi consolidada em um ambiente de produção baseado na Amazon Web Services (AWS), operando sob uma topologia híbrida que desacopla a unidade computacional da camada de persistência.

O núcleo do processamento reside em uma instância EC2 executando Ubuntu Linux, que centraliza tanto o servidor web quanto a lógica de negócios, enquanto o armazenamento de dados é gerenciado externamente por uma instância AWS RDS isolada logicamente, garantindo maior robustez e escalabilidade ao ambiente.

No que tange à camada de apresentação, a aplicação desenvolvida em React (Single Page Application) foi transpilada para arquivos estáticos e implantada no diretório /var/www/polaris.

Para servir este conteúdo, utiliza-se o Apache HTTP Server (v2.4) operando na porta 80, configurado com diretivas de mod_rewrite para gerenciar o roteamento interno da interface do usuário.

Simultaneamente, a lógica de aplicação (Backend) é executada em ambiente Node.js e gerenciada pelo monitor de processos PM2, assegurando alta disponibilidade através de reinicialização automática.

A integração entre essas camadas ocorre via configuração de Proxy Reverso no Apache, que intercepta as requisições destinadas à API e as encaminha internamente para a porta 5000 — mantida oculta de acesso público direto —, aplicando ainda políticas de CORS ajustadas especificamente para o IP da instância.

A persistência de dados é realizada no serviço gerenciado AWS RDS utilizando o motor MariaDB. A conectividade entre o servidor de aplicação na EC2 e o banco de dados trafega exclusivamente pelo protocolo TCP/IP na porta 3306. Em conformidade com os requisitos de segurança da infraestrutura (require_secure_transport=ON), a conexão utiliza criptografia SSL/TLS autenticada pelo AWS Certificate Authority Global Bundle, assegurando a confidencialidade e integridade das transações acadêmicas durante o tráfego de dados.

Políticas de segurança adotadas:

Validar no backend a propriedade dos recursos acessados (userID da sessão vs. userID da requisição).

Identificamos todas as requests para o crud principal por meio de userID do usuário logado

Restringir rotas sensíveis a perfis específicos, com regras explícitas de autorização.

Aplicar o princípio do menor privilégio em todos os níveis do sistema.
Temos um sistema hierárquico de "roles": Admin > Moderador > Professor > Aluno, em que cada "role", possui permissão acima dos demais abaixo.

Bloquear o acesso direto a IDs numéricos previsíveis (preferir UUIDs).

Utilizamos UUID'S dentro da database para criar os cadastros

Registrar e auditar tentativas de acesso indevido

Temos um sistema de log que acompanha tentativas de acesso indevido

Evitar qualquer lógica de autorização no frontend sem validação equivalente no backend.

Toda nossa lógica de autenticação é feita no backend

Armazenar senhas usando hash seguro (bcrypt, Argon2id, PBKDF2) com salt

Imposição de políticas de senha forte e armazenamento seguro usando hash moderno

Evitar algoritmos obsoletos (MD5, SHA-1, DES, RC4).

Usamos bcrypt para nossa criptografia das senhas de cadastro.

Desabilitar serviços, módulos e portas que não são necessários em produção.

Todas as rotas e serviços estão sendo utilizadas no sistema

Configurar CORS de forma restritiva, aceitando apenas origens legítimas.

A política de CORS está configurada para que apenas o backend possa fazer autenticação, backend esse cujo ip está em uma variável de ambiente. Poderia-se expandir essa variável de ambiente em uma lista para que fossem autorizados outros ips ou domínios da universidade.

Ocultar mensagens de erro detalhadas em ambientes públicos.

As mensagens de erro comunicam apenas o necessário para o frontend

Utilizar cabeçalhos de segurança adequados – CORTOU

Utilizou-se o padrão do node.js express

Bloqueio temporário ou rate limit após várias tentativas de login.

Aplicamos lógica generalizada de rate limit para o mesmo ip, e uma lógica mais restrita de rate-limit para tentativas de login da mesma origem

Não aplicado no projeto

Assinar e criptografar tokens, garantindo entropia adequada nas chaves.

Geração de tokens imprevisíveis e assinados, com expiração adequada.

Invalidação imediata de tokens após logout ou troca de senha.

Nosso token de autenticação que vem do frontend é uma key única, um JWT(JSON WEB TOKEN) estático. Futuramente deveria se implementar uma lógica de autenticação OAUTH2 que use o Amazon Cognito ou similar para que hajam tokens únicos e expiráveis

Rotacionar periodicamente chaves criptográficas sensíveis.

Criptografar dados sensíveis em repouso quando aplicável.

Não possuímos script que varra a database e faça um crypt e decrypt dos dados sensíveis dos clientes ou alternativas, nem uma modularidade que permita alternar a chave de criptografia consistentemente.

Configuração correta do ciclo de vida de sessões, com expiração automática após inatividade.

Não foi configurada expiração de sessão por tempo e/ou inatividade