Segurança da Informação em uma instituição de ensino superior: Estudo de Caso de Falhas em Aplicações Web e Mobile

Iaan Mesquita de Souza[1], **Sérgio Teixeira de Carvalho**[1]

[1]Instituto de Informática – Universidade Federal de Goiás (UFG)

iaanmesquita@discente.ufg.br sergiocarvalho@ufg.br

Abstract. *This article presents a security case study of three anonymized institutional systems at a higher education instituion - System Alpha, System Beta, and System Omega. Using static and dynamic analysis, controlled traffic inspection, and safe parameter manipulation, we identified: (i) improper exposure of information in Alpha due to access control and configuration weaknesses; (ii) SQL injection in Beta's login, enabling unauthorized access to sensitive records; and (iii) in Omega, a predictable association between identifiers and images and enumeration via pagination. The evidence was anonymized and responsibly disclosed; maintainers applied fixes that were subsequently validated. The findings were mapped to the OWASP Top 10, and the paper systematizes the results, provides a critical discussion of the vulnerabilities, and highlights lessons learned.*

Resumo. *Este artigo apresenta um estudo de caso de segurança em três sistemas institucionais anonimizados de uma instituição de ensino superior - Sistema Alpha, Sistema Beta e Sistema Ômega. Com análise estática e dinâmica, inspeção controlada de tráfego e manipulação segura de parâmetros, identificamos: (i) exposição indevida de informações em Alpha, decorrente de falhas de controle de acesso e de configuração; (ii) injeção de SQL no login de Beta, permitindo acesso não autorizado a registros sensíveis; e (iii) em Ômega, associação previsível entre identificadores e imagens e enumeração por paginação. As evidências foram anonimizadas e divulgadas de forma responsável e os mantenedores aplicaram correções, posteriormente validadas. As falhas foram mapeadas ao OWASP Top 10, e o trabalho sistematiza os achados, discute criticamente as vulnerabilidades e as lições aprendidas.*

1. Introdução

A transformação digital nas universidades brasileiras intensificou a dependência de sistemas web e aplicações móveis para a gestão acadêmica e administrativa. Com a crescente digitalização e dependência de sistemas conectados a internet para operações críticas, a segurança da informação tornou-se um requisito fundamental para organizações de todos os portes e segmentos [IBM 2024], e as instituições de ensino superior não são exceção. Esse movimento amplia a superfície de ataque e expõe instituições, estudantes e servidores a riscos significativos quando princípios básicos de engenharia segura não são observados. Nesse contexto, iniciativas de avaliação proativa de segurança, alinhadas a boas

práticas internacionalmente reconhecidas, como o *Open Web Application Security Project* (OWASP) [OWASP Foundation 2021a] *Top 10*, tornam-se essenciais para antecipar ameaças, reduzir vulnerabilidades e fortalecer a confiança dos usuários e da sociedade.

Este artigo apresenta um estudo de caso conduzido em três sistemas de uma instituição de ensino superior (denominados, *Sistema* ***Alpha***, *Sistema* ***Beta*** e *Sistema* ***Ômega***), com foco em identificar, analisar, documentar e notificar de forma responsável vulnerabilidades que pudessem afetar a confidencialidade, a integridade e a disponibilidade de dados e serviços. A investigação foi realizada sem ações destrutivas e com escopo estritamente técnico, combinando análise estática e dinâmica, inspeção controlada de tráfego, manipulação de parâmetros e testes de enumeração, sempre observando limites éticos e legais.

Justificativa da escolha dos sistemas: Os três sistemas analisados foram selecionados especificamente por possuírem acesso público pela Internet, o que permitiu a realização dos testes de segurança de forma ética e controlada, sem a necessidade de credenciais privilegiadas. Esta escolha, porém, apresenta limitações: sistemas de acesso público podem não representar toda a complexidade da infraestrutura institucional, e vulnerabilidades em sistemas internos ou com autenticação mais robusta podem não ter sido detectadas. Ainda assim, a análise de sistemas públicos é relevante por estes serem frequentemente o primeiro alvo de atacantes e por representarem a interface mais exposta da instituição.

Todo o material coletado foi anonimizado, incluindo nomes de sistemas, URLs, identificadores e quaisquer dados pessoais. O propósito deste artigo é científico e educativo, sem a intenção de expor indevidamente a instituição ou seus desenvolvedores. Deliberadamente, o trabalho não apresenta ”receitas de ataque”, nem passos operacionais detalhados, mitigando o risco de uso indevido.

Como síntese das evidências e para orientar a leitura, a Tabela 1 resume as vulnerabilidades observadas em cada sistema, a gravidade atribuída, o mapeamento correspondente ao OWASP *Top 10* e os principais impactos. Em termos gerais, observou-se:

Sistema Alpha: Um caso típico de *Insecure Direct Object Reference* (IDOR). O *backend* validava o *token* de sessão, porém não aplicava verificação de autorização por objeto (*ownership*), permitindo consultar, alterar e deletar dados de outros usuários.

Sistema Beta: Uma injeção de Script Query Language (SQL) no fluxo de autenticação que possibilitava acesso não autorizado a registros sensíveis e confidenciais, e por consequência, a problemas de autorização.

Sistema Ômega: A associação determinística entre identificadores e arquivos de imagem, a possibilidade de enumerar usuários por paginação de uma *Application Programming Interface* (API), possibilidade de sobrecarregar o servidor com requisições custosas e fragilidades no uso de identificadores com potencial para consumo indevido de créditos de terceiros.

Os impactos potenciais envolveram violação de privacidade, fraude transacional, alteração de dados, indisponibilidade e ampliação não autorizada de privilégios, com severidade predominante alta/crítica. O procedimento de divulgação responsável foi seguido rigorosamente: as falhas foram documentadas e comunicadas imediatamente à

Tabela 1. Resumo das vulnerabilidades observadas e mapeamento ao OWASP (*Top 10*, API e *Mobile*).

Sistema	Vulnerabilidades encontradas	OWASP ID	Gravidade
Alpha	IDOR; enumeração de usuários; leitura/alteração/exclusão de dados	A01:2021; A05:2021; API1:2023; API3:2023; API8:2023	Alta
Beta	*SQL injection*	A03:2021; A01:2021	Crítica
Ômega	IDs determinísticos em imagens; paginação enumerável	A01:2021; A04:2021; A05:2021; API1:2023; API3:2023; API4:2023; API8:2023; M3; M8; M9	Alta

coordenação do curso para conectar com às equipes responsáveis, que procederam com as correções. A redação deste artigo foi concluída após a verificação das correções aplicadas às vulnerabilidades reportadas, assegurando que não seja mais possível reproduzir qualquer exploração aqui descrita.

Em resumo. as contribuições deste trabalho são:

- Sistematização de três estudos de caso anonimizados (*Alpha*, *Beta* e *Ômega*), representativos de classes recorrentes de vulnerabilidades.
- Mapeamento explícito ao OWASP *Top 10*, OWASP API *Top 10* e OWASP *Mobile Top 10*, facilitando a priorização de riscos e o diálogo entre equipes de desenvolvimento, segurança e gestão.
- Discussão sobre mitigações com referência a trabalhos relacionados que abordam controles de segurança em maior profundidade [Libâneo and Carvalho 2016, OWASP Foundation 2021a, OWASP Foundation 2021b].
- Ênfase em ética, privacidade e divulgação responsável como pilares para fortalecer a segurança institucional e, por extensão, o ecossistema nacional de Segurança da Informação.

O restante do artigo está organizado da seguinte forma: a Seção 2 apresenta a fundamentação teórica e trabalhos relacionados. A Seção 3 detalha os três estudos de caso (*Sistema **Alpha***, *Sistema **Beta*** e *Sistema **Ômega***) com uma análise crítica e implicações práticas das vulnerabilidades encontradas. A Seção 4 conclui com as considerações finais.

2. Fundamentação Teórica e Trabalhos Relacionados

A segurança da informação pode ser definida como o conjunto de práticas e tecnologias que protegem dados digitais contra acesso não autorizado, uso indevido ou roubo ao longo de todo o seu ciclo de vida [IBM 2024]. Esta proteção abrange não apenas aspectos técnicos, mas também processos organizacionais e conscientização humana.

Assim sendo, nesta seção são apresentados os conceitos fundamentais que embasam este trabalho e trabalhos relacionados da literatura. Adotamos como eixo principal o *OWASP Top 10* para aplicações web [OWASP Foundation 2021a], o *OWASP API Top 10* para APIs [OWASP Foundation 2023] e o *OWASP Mobile Top 10* [OWASP Foundation 2016] para aplicações móveis .

Ademais, a análise de vulnerabilidades em sistemas de instituições de ensino superior tem se mostrado uma preocupação crescente no contexto brasileiro, especialmente diante da crescente digitalização dos processos acadêmicos e administrativos.

[Libâneo and Carvalho 2016] apresentam um mapeamento abrangente das principais vulnerabilidades em aplicações web e seus controles de segurança, baseando-se no relatório OWASP *Top 10*. O trabalho detalha controles específicos para cada classe de vulnerabilidade, incluindo validação de entradas, gerenciamento de sessão, codificação de saída, proteção de dados e controle de acesso. Este material é particularmente relevante para o presente estudo, pois oferece diretrizes práticas de mitigação que complementam nossa análise empírica. Como os autores destacam, técnicas como *query parameterization* (consultas parametrizadas) são essenciais para prevenir injeção de SQL, enquanto o gerenciamento adequado de sessão e autenticação de múltiplos fatores protegem contra quebra de autenticação.

[Borges and da Silva 2025] realizaram uma análise de vulnerabilidades em aplicações web de uma universidade privada e identificaram 517 falhas. Uma parcela significativa permitia a execução de código arbitrário e a movimentação lateral pela rede, evidenciando a fragilidade dos controles de segurança da informação em organizações que administram ampla gama de dispositivos e dados sensíveis de milhares de indivíduos. O estudo reforça a urgência de adoção de medidas e arquiteturas de cibersegurança robustas para a proteção de dados pessoais e acadêmicos.

De forma complementar, [Fernandes 2019] identificou vulnerabilidades críticas no Sistema Integrado de Gestão de Atividades Acadêmicas (SIGAA), sistema amplamente utilizados por instituições de ensino, majoritariamente instituições públicas. As vulnerabilidades documentadas pelo autor foram *SQL Injection*, *Cross-Site Scripting* (XSS) e *Insecure Direct Object Reference* (IDOR). O estudo documentou que 42 instituições de ensino superior utilizam o sistema, evidenciando o alcance potencial dessas falhas. O autor seguiu práticas de divulgação responsável, notificando previamente os desenvolvedores sobre as vulnerabilidades identificadas.

[de Carvalho and de Castro Júnior 2014] investigaram a combinação de técnicas de *Google Hacking* com exploração de *SQL Injection*, demonstrando como atacantes utilizam operadores avançados de busca para identificar sistemas vulneráveis e, posteriormente, explorar falhas de injeção de SQL. O estudo de caso apresentado pelos autores evidencia a facilidade com que sistemas web mal configurados podem ser descobertos e comprometidos, reforçando a necessidade de testes proativos de segurança. As recomendações incluem uso de consultas parametrizadas, configuração inteligente de privilégios do banco de dados e testes regulares com as mesmas técnicas usadas por atacantes.

Estes estudos convergem ao destacar a importância crítica de avaliações proativas de segurança para proteger dados pessoais e acadêmicos em ambientes universitários, ressaltando a urgência de políticas de segurança da informação alinhadas às melhores práticas internacionais como as definidas pela OWASP.

2.1. Taxonomias e Modelos Relevantes

O *OWASP Top 10* é um documento padrão de conscientização para desenvolvedores que sintetiza classes recorrentes de risco em aplicações a partir de dezenas de fraquezas ma-

peadas. Ele representa um amplo consenso sobre os riscos de segurança mais críticos para aplicações web [OWASP Foundation 2021a]. Em conjunto com os *OWASP API Top 10* [OWASP Foundation 2023] e *OWASP Mobile Top 10* [OWASP Foundation 2016] esses catálogos organizam mais de 30 categorias de risco, oferecendo um vocabulário comum e prioridades práticas para mitigação.

À luz desses referenciais, os achados foram mapeados às categorias do OWASP, conforme a Tabela 1, privilegiando uma leitura orientada a risco e a mitigação. Esse enquadramento fornece a base conceitual para as recomendações técnicas e de processo apresentadas nas seções seguintes, sem prescrever procedimentos operacionais detalhados, em alinhamento à divulgação responsável.

3. Estudos de Caso

Este estudo integra uma avaliação de segurança com base nas diretrizes OWASP *Top 10*, OWASP API *Top 10* e OWASP *Mobile Top 10*.

3.1. Sistema *Alpha*

3.1.1. Contexto e escopo

O escopo abrangeu a análise pontual de duas rotas da API do Sistema *Alpha*, plataforma de gestão acadêmica utilizada por discentes e docentes e acessível via navegador web. A abordagem foi minimamente invasiva, limitada à demonstração técnica da falha (prova de conceito), sem exfiltração massiva de dados, seguida de comunicação imediata e responsável aos mantenedores, em conformidade com princípios de divulgacão responsável. Após o reporte e a confirmação da vulnerabilidade, a equipe técnica implementou as correções e emitiu parecer atestando sua remediação. A validação final consistiu na execução de requisições simples às rotas afetadas, confirmando a eliminação do comportamento indevido.

3.1.2. Vulnerabilidades identificadas

A primeira etapa consistiu na inspeção das requisições realizadas pelo cliente (navegador web) ao servidor. Foram identificadas duas rotas Representational State Transfer (REST) parametrizadas por identificador no caminho da URL:

```
/dados-pessoais/123
/dados-pagamentos/123
```

em que ”123” correspondia ao identificador do próprio usuário autenticado. O recebimento do identificador diretamente no *path* (parâmetro de rota), quando combinado a validações incompletas no *backend*, é um padrão associado a falhas de controle de acesso em nível de objeto.

Teste 1 - Ausência de *token*. Visando verificar a existência de validação básica de autenticação, foram emitidas requisições sem o cabeçalho de autorização. O sistema rejeitou adequadamente tais requisições (por exemplo, com `401/403`), indicando que de fato havia checagem de existência/validade do *token* antes do processamento do recurso, conforme esperado.

Teste 2 - *Token* válido e identificador (id) de terceiro. Para avaliar o controle de autorização por objeto, repetiu-se a requisição agora com um *token* válido, porém

alterando-se apenas o id na URL para um valor pertencente a outro usuário. Nessas condições, o *backend* retornou integralmente os dados do recurso associado ao id informado, confirmando a ausência de verificação de posse/pertinência entre o usuário autenticado *(claims do token)* e o objeto solicitado (id na rota). A comprovação técnica foi conduzida com a ferramenta `curl`, presente em sistemas operacionais.

Conforme evidenciado na Figura 1, a API retornou dados pessoais completos ao se alterar apenas o identificador na URL, mesmo com *token* pertencente a outro usuário.

Figura 1. Resposta da API a uma requisição aos endpoints `/dados-pessoais/<id>` com token válido, porém com `id` de terceiro. Campos sensíveis foram ofuscados.

Comportamento análogo foi observado na rota `/dados-pagamentos/<id>`, que retornava metadados de pagamentos previamente utilizados (por exemplo, últimos dígitos, bandeira e validade). Embora a aplicação não armazenasse dados sensíveis do cartão (tokenização realizada pelo adquirente), o acesso a metadados de terceiros configura exposição indevida de informações pessoais e financeiras.

Adicionalmente, verificou-se que os identificadores utilizados nas rotas eram sequenciais e de baixa entropia (por exemplo, 1, 2, 3, 4, ...). Tal característica, quando combinada à ausência de autorização por objeto evidenciada na Figura 1, facilita a enumeração sistemática de contas e a automação de requisições para diferentes identificadores. Em um cenário sem limitação de taxa, sem detecção de anomalias e com respostas determinísticas, ferramentas triviais (por exemplo, `curl` encadeado em *scripts*) poderiam iterar o parâmetro de id e, potencialmente, exfiltrar dados de grande parte da base de usuários. Por razões éticas, não foi realizada coleta massiva. A inferência decorre do padrão de identificadores observado, do comportamento de autorização descrito e na prova de conceito, enviada prontamente aos responsáveis pelo sistema.

3.2. Sistema *Beta*

3.2.1. Contexto e escopo

O escopo incluiu a análise do fluxo de autenticação do Sistema *Beta*, com foco na superfície de ataque exposta por campos de entrada no formulário. O objetivo foi identificar vulnerabilidades de alto impacto, validar de forma minimamente invasiva e comunicar imediatamente aos responsáveis, seguindo práticas de divulgação responsável. Não houve exfiltração de dados. Os testes limitaram-se à comprovação técnica da vulnerabilidade e

à comunicação imediata. As falhas reportadas já se encontram corrigidas pelos mantenedores.

Adicionalmente, após a identificação desta falha específica (classificada como crítica), o autor decidiu não prosseguir com testes adicionais, interrompendo a prova de conceito no ponto mínimo necessário e comunicando imediatamente a equipe responsável, em conformidade com princípios éticos e de divulgação responsável. Em termos estritamente teóricos, vulnerabilidades dessa natureza podem servir como ponto de partida para tentativas de elevação de privilégios, variando desde o acesso a áreas administrativas do sistema até, em cenários extremos, a execução remota de comandos e eventual controle total do servidor. Ressalta-se, contudo, que tais hipóteses não foram testadas nem exploradas, justamente por se tratar de uma falha de alto impacto, tendo prevalecido a decisão prudente de priorizar a mitigação rápida e a proteção dos dados.

3.2.2. Vulnerabilidades identificadas

A principal vulnerabilidade observada foi uma injeção de SQL no ponto de *login*, causada pela concatenação direta de entrada do usuário na construção de consultas, sem uso de parâmetros vinculados *(prepared statements)* e sem saneamento robusto. Na sondagem inicial, a simples inserção de um apóstrofo (`'`) no campo de *login* gerou o alerta de sintaxe SQL

```
Warning: sqlanywhere_query() [function.sqlanywhere-query]:
SQLAnywhere: [-131] Syntax error near ...
```

o que é um forte indício de que o valor foi interpolado dentro de um literal de *string* na cláusula `WHERE`. Em cenários dessa natureza, *payloads* clássicos como `' OR 1=1;--` podem alterar a lógica da consulta, resultando em autenticação indevida.

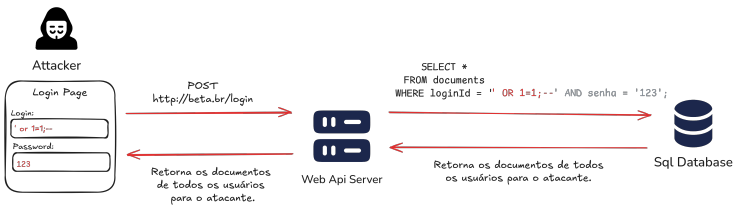

Figura 2. Fluxo do *bypass* por SQL Injection no *login*: em vermelho, o payload injetado; em cinza, a parte comentada que é ignorada, levando o *backend* a retornar todos os documentos.

Conforme ilustrado na Figura 2, o atacante preenche o campo de *login* com o *payload* `' OR 1=1;--` e um valor qualquer no campo *password*. A aplicação, ao concatenar diretamente esses valores na `WHERE`, acaba gerando uma consulta semelhante a:

```
 SELECT * FROM documents WHERE loginId = ''
OR 1=1;-- ' AND senha = '123';
```

Na figura, em vermelho está destacado o *payload* inserido (`' OR 1=1;--`); em cinza, a parte da consulta que passa a ser ignorada por causa do comentário. O resultado prático é que a condição fica sempre verdadeira e o banco retorna todos os documentos de todos os usuários, que acabam sendo enviados ao atacante pelo *backend*.

Por que o *payload* funciona: O *payload* funciona porque o primeiro apóstrofo (`'`) fecha prematuramente o literal de string esperado para `loginId`, alterando a estrutura sintática da consulta. Em seguida, o operador `OR` introduz a tautologia `1=1`, tornando a cláusula `WHERE` verdadeira para todas as linhas. Quando aceito pelo Sistema Gerenciador de Banco de Dados (SGBD), o `;` encerra a instrução atual, e o marcador de comentário `--` faz com que todo o restante da linha seja ignorado pelo *parser* SQL, incluindo a cláusula legítima `AND senha = '123'`. Como resultado, a `WHERE` efetiva reduz-se a uma condição verdadeira (`... WHERE TRUE`), removendo o filtro pretendido e possibilitando o acesso indevido.

3.3. Sistema *Ômega*

3.3.1. Contexto e escopo

O escopo contemplou a avaliação exploratória de uma aplicação móvel integrada ao ecossistema institucional, com foco no comportamento de serviços de *backend* e na eventual exposição de dados pessoais. A atividade foi conduzida de modo minimamente invasivo, com inspeção de tráfego por meio de ferramenta genérica de análise de rede. Como a camada cliente empregava verificação de certificado *(SSL pinning)*, foi necessário contornar esse mecanismo exclusivamente para fins de observação e comprovação técnica do fluxo de requisições/respostas entre aplicativo e servidor. Não houve coleta massiva de dados, tampouco retenção indevida de conteúdos sensíveis. A validação restringiu-se ao mínimo necessário para caracterização da vulnerabilidade e comunicação imediata e responsável aos mantenedores, em alinhamento a princípios de divulgação responsável. Após o reporte, as correções foram implementadas pela equipe técnica, que atestou a remediação. A validação final consistiu em requisições simples para confirmar a eliminação do comportamento indevido.

3.3.2. Vulnerabilidades identificadas

A investigação seguiu um encadeamento metodológico centrado na observação das rotas efetivamente utilizadas pelo aplicativo. Inicialmente, realizou-se o contorno das verificações de *SSL pinning* no aplicativo, possibilitando a inspeção do tráfego protegido entre o aplicativo e os serviços de *backend* por meio de ferramenta genérica de análise de rede. Com a visibilidade do tráfego, identificaram-se as rotas chamadas pelo aplicativo para obtenção de recursos estáticos e dados pessoais, incluindo o caminho lógico associado a imagens de perfil. A partir de então, a investigação iniciou-se a partir da observação de que a URL utilizada pela aplicação para exibir a foto de perfil aparentava conter um identificador coincidente com o CPF do usuário.

O padrão observado era compatível com caminhos do tipo:

`https://ômega.br/sys623/fotos/18859946050.jpg`

Sugeriu-se, então, como hipótese de trabalho, que o nome do arquivo (11 dígitos + `.jpg`) coincidisse com um identificador civil (CPF) do titular. À época, esta correlação era uma hipótese, não uma certeza.

Ao solicitar o diretório lógico do recurso (`/sys623/fotos`) sem parâmetros adicionais, o serviço retornou um documento *Extensible Markup Language* (XML) com

exatos 10 registros de pessoas, contendo `id`, `nome_completo` e o nome do arquivo de `foto`, com padrão numérico de 11 dígitos seguido de `jpg`. O documento era semelhante à:

```
<?xml version="1.0" encoding="UTF-8"?>
<pessoas>
    <pessoa>
        <id>1234567</id>
        <nome_completo>Ana Silva Santos</nome_completo>
        <foto>12345678901.png</foto>
    </pessoa>
    ...
</pessoas>
```

A partir desse ponto, constatou-se que o *endpoint* de listagem respondia sem exigir prova de autenticação ou, alternativamente, sem vincular a autorização ao titular dos dados. O atributo `foto` apresentava um nome de arquivo numericamente compatível com um CPF em formato canônico (11 dígitos, sem separadores), o que implica uma vinculação direta entre um CPF e um dado biométrico identificável (imagem facial). Ainda que o CPF não seja classificado como ”sigiloso” por si, sua divulgação combinada com a fotografia e o nome completo eleva substancialmente o potencial de danos ao titular.

Adicionalmente, como o documento retornava exatamente 10 elementos, formulou-se a hipótese de paginação implícita e testaram-se parâmetros comumente empregados por APIs para navegação de resultados (`?offset=` e `?max=`), com o objetivo de obter lotes adicionais e percorrer o conjunto. A hipótese confirmou-se: ambos os parâmetros foram aceitos e permitiram a navegação determinística entre páginas.

Observou-se, ainda, que o `max` não parecia sofrer limitação *server-side* adequada: ao fornecê-lo com um valor suficientemente elevado, o serviço retornou, em uma única resposta, a totalidade dos registros disponíveis (isto é, todos os usuários do aplicativo). Embora a latência fosse elevada (ordem de minutos), o comportamento caracterizou de forma inequívoca a vulnerabilidade de enumeração em escala. Para corroborar a associação entre identificadores e imagens, realizou-se acesso pontual a um subconjunto mínimo de URLs de fotografia retornadas no XML, confirmando a disponibilidade das imagens no formato utilizado pelo aplicativo. A coleta foi estritamente limitada e interrompida assim que a hipótese foi confirmada, priorizando a comunicação imediata aos responsáveis e a mitigação célere do problema.

Sobretudo, é importante pontuar que, a possibilidade de observar o tráfego do aplicativo por ferramenta de análise de rede foi relevante para a detecção, mas não constitui a causa-raiz. A falha é predominantemente *server-side*, decorrente de controles de autorização insuficientes e de desenho inadequado da interface de listagem. Mesmo que o aplicativo móvel adote mecanismos de reforço (p.ex., verificação de certificado), a correção correta deve residir no *backend*.

4. Considerações finais

Este trabalho realizou um estudo de caso, de maneira ética e anonimizada de três sistemas de uma instituição de ensino superior, com mapeamento dos achados às taxonomias OWASP e correções verificadas via divulgação responsável. Os resultados evidenciaram a recorrência de classes de risco bem documentadas na literatura. As notificações responsáveis resultaram na pronta correção das vulnerabilidades, confirmada posteriormente por validações pontuais.

O estudo, de natureza exploratória e conduzido sob restrições éticas, teve escopo limitado a rotas e funcionalidades específicas, sem coleta massiva ou testes destrutivos, o que inviabiliza estimativas quantitativas e pode deixar variáveis não observadas. As inferências decorrem de evidências empíricas, e a generalização para outros contextos exige cautela diante de diferenças arquiteturais e de governança. Ainda assim, a convergência com a recomendações de padrões como OWASP, sustenta a validade externa das recomendações apresentadas pelas diretrizes definidas.

Em síntese, a contribuição deste trabalho está na sistematização de evidências empíricas sobre vulnerabilidades representativas em ambientes acadêmicos brasileiros e no seu enquadramento às taxonomias OWASP.

5. References

Referências

Borges, J. G. and da Silva, C. A. (2025). Análise de segurança em aplicações web acadêmicas.

de Carvalho, A. W. and de Castro Júnior, A. P. (2014). Google hacking para ataques sql injection. In *Anais da II Escola Regional de Informática de Goiás*, pages 65–77, Goiânia, GO, Brasil. Escola Regional de Informática de Goiás, SBC.

Fernandes, V. (2019). Hackeando instituições de ensino no brasil. `https://vitor-fernandes.github.io/Hacking-Universities/`. Acessado em: 09 nov. 2025.

IBM (2024). O que é segurança da informação? `https://www.ibm.com/br-pt/think/topics/information-security`. Acesso em: 09 nov. 2025.

Libâneo, M. C. and Carvalho, S. T. (2016). Defendendo aplicações web: Mapeando as principais vulnerabilidades e seus controles de segurança. In *Anais da IV Escola Regional de Informática de Goiás*, pages 167–179, Goiânia, GO, Brasil. Escola Regional de Informática de Goiás, SBC.

OWASP Foundation (2016). Owasp mobile top ten 2016. `https://owasp.org/www-project-mobile-top-10/`. Acessado em: 09 nov. 2025.

OWASP Foundation (2021a). Owasp top 10 - 2021. `https://owasp.org/Top10/`. Acessado em: 09 nov. 2025.

OWASP Foundation (2021b). Sql injection prevention cheat sheet. `https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html`. Acessado em: 09 nov. 2025.

OWASP Foundation (2023). Owasp api security top 10 - 2023. `https://owasp.org/API-Security/editions/2023/en/0x00-introduction/`. Acessado em: 09 nov. 2025.