DIRETRIZES PARA EVOLUÇÃO DO GAIO COM BASE EM REQUISITOS DE SEGURANÇA WEB E ANÁLISE AUTOMATIZADA POR OWASP ZAP

Marcos Vinícius Freitas Andrade, Thiago Dias de Carvalho Quaresma Gama

Universidade Estadual de Goiás, Pires do Rio - GO, Brasil
(freitasmarcosviniicius@gmail.com)

Resumo: Este trabalho propõe diretrizes para a evolução do GAIO a partir da análise de relatório automatizado de segurança web gerado pelo OWASP ZAP. Os achados são interpretados como insumos para definição de novos recursos, requisitos não funcionais e critérios de priorização de versões futuras. A proposta organiza melhorias em eixos de segurança, gestão de sessão, endurecimento da aplicação, governança técnica e integração dos alertas ao *backlog* do sistema.

Palavras-chave: Segurança web; OWASP ZAP; Requisitos não funcionais; Evolução de software; Desenvolvimento seguro.

INTRODUÇÃO

Sistemas web voltados à gestão de projetos e tarefas deixaram de ser apenas ferramentas operacionais de registro e acompanhamento de atividades. Em contextos organizacionais, essas plataformas passam a concentrar informações sobre equipes, responsabilidades, prioridades, prazos, histórico de execução, indicadores de produtividade e decisões estratégicas. Por esse motivo, a evolução de uma aplicação desse tipo deve considerar não apenas a inclusão de novas funcionalidades visíveis ao usuário, mas também requisitos de segurança, confiabilidade, rastreabilidade, privacidade, integridade e governança técnica.

O GAIO - Gestão Ágil Integrada de Operações - insere-se nesse contexto como uma plataforma voltada à gestão de tarefas, organização visual por Kanban, colaboração entre equipes, controle de atividades, filtros de acompanhamento, autenticação, cadastro de usuários e recursos de gamificação, conforme proposta inicial da ferramenta (Sousa et al., 2025). Além dessas funcionalidades, o sistema incorpora uma proposta de apoio à decisão por meio do NRP (em inglês, *Next Release Problem*: problema da próxima versão), associado à priorização de versões futuras (Bagnall et al., 2001). Assim, o GAIO apresenta potencial para atuar não apenas como ferramenta de execução operacional, mas também como ambiente de apoio ao planejamento, monitoramento e evolução contínua de projetos de software (Beck et al., 2001).

Na Engenharia de Software, os requisitos podem ser compreendidos como descrições das funcionalidades, restrições e condições de qualidade que orientam o desenvolvimento de um sistema. Enquanto os requisitos funcionais indicam os serviços que a aplicação deve oferecer, os requisitos não funcionais estabelecem atributos relacionados à qualidade, desempenho, confiabilidade, segurança, disponibilidade e manutenibilidade (Sommerville, 2016). Assim, em uma plataforma como o GAIO, a segurança não deve ser vista como elemento periférico, mas como condição necessária para garantir confiança, integridade e continuidade no uso da aplicação.

A engenharia de requisitos também contribui para estruturar a forma como necessidades, restrições e critérios de qualidade são identificados, documentados e validados ao longo do ciclo de vida do software. Nesse sentido, requisitos de segurança precisam ser explicitados de forma verificável, permitindo que sejam associados a critérios de aceite, evidências de teste e decisões de priorização em versões futuras (ISO/IEC/IEEE, 2018).

A segurança por projeto reforça essa perspectiva ao defender que as decisões de segurança sejam incorporadas desde a concepção do produto. Isso significa que cada novo recurso deve considerar, antes de sua implementação, quais dados serão manipulados, quais perfis terão acesso, quais ações alteram o estado da aplicação, quais validações serão necessárias e quais controles devem ser adotados para reduzir riscos (OWASP Foundation, s.d.d).

Nesse sentido, a consolidação de uma plataforma de gestão como o GAIO exige que seus novos recursos sejam planejados com base em diretrizes de

segurança, evitando que controles essenciais sejam tratados apenas como correções posteriores. A segurança, portanto, não deve ser tratada como uma etapa isolada de correção de falhas, mas como parte do processo de definição de requisitos (Souppaya et al., 2022). Cada novo módulo, tela, fluxo de autenticação, recurso de gamificação, mecanismo de priorização, integração externa ou funcionalidade administrativa deve considerar riscos associados a sessão, autorização, exposição de dados, validação de entradas, proteção contra requisições indevidas e configuração segura da aplicação (OWASP Foundation, 2021).

O relatório automatizado de segurança gerado pelo OWASP ZAP (em inglês, *Zed Attack Proxy*: ferramenta mantida pela Open Worldwide Application Security Project) para o ambiente público do GAIO fornece evidências relevantes para orientar esse processo (ZAP Dev Team, s.d.). A análise não identificou alertas classificados como risco alto, mas apontou achados de risco médio, baixo e informativo. Entre eles, destacam-se ausência de tokens *Anti-CSRF*, ausência de *Content Security Policy* (CSP), ausência de cabeçalhos de proteção contra *clickjacking*, possível exposição de informações por *Spring Actuator*, ausência de atributo *SameSite* em *cookie*, ausência de HSTS (em inglês, *HTTP Strict Transport Security*), ausência do cabeçalho *X-Content-Type-Options*, inclusão de JavaScript de domínio externo, diretivas de cache a reexaminar e identificação de fluxos de autenticação e gerenciamento de sessão.

Diante disso, este trabalho parte do seguinte problema: como os achados de um relatório automatizado de segurança web podem ser convertidos em diretrizes para novos recursos e requisitos de segurança do GAIO?

O objetivo geral é propor diretrizes para evolução funcional e técnica do GAIO com base nos achados do relatório OWASP ZAP, de modo que novos recursos sejam planejados considerando requisitos de segurança, governança técnica e priorização evolutiva.

Como objetivos específicos, busca-se: analisar os achados do relatório de segurança aplicado ao GAIO; converter os alertas identificados em requisitos não funcionais; propor diretrizes para novos recursos da plataforma; organizar os requisitos de segurança por eixos de evolução; e indicar como tais requisitos podem ser incorporados ao *backlog* e priorizados em versões futuras do sistema.

MATERIAL E MÉTODOS

Este trabalho caracteriza-se como uma pesquisa aplicada, qualitativa e exploratória. É aplicada porque busca produzir diretrizes concretas para evolução do GAIO. É qualitativa porque interpreta os achados do relatório de segurança conforme seu significado técnico, funcional e arquitetural. É exploratória porque utiliza um relatório automatizado como ponto de partida para identificar oportunidades de melhoria e transformação dos achados em requisitos de software.

O objeto de análise é o GAIO, uma plataforma web de gestão de tarefas e projetos que combina *Kanban*, gestão de equipes, autenticação, cadastro, acompanhamento de atividades, filtros, gamificação e priorização de versões por NRP. A escolha desse objeto justifica-se pela necessidade de alinhar a evolução funcional do sistema a requisitos de segurança e governança, especialmente em uma aplicação que tende a manipular informações de usuários, equipes, tarefas, prioridades e histórico de decisões.

A principal fonte documental utilizada foi o relatório de varredura automatizada gerado pelo OWASP ZAP (ZAP Dev Team, s.d.). O relatório apresentou alertas distribuídos em níveis médio, baixo e informativo, sem identificação de risco alto. Para fins deste estudo, os achados foram analisados como evidências técnicas capazes de orientar requisitos de segurança e diretrizes para novos recursos.

A análise foi conduzida em cinco etapas. Na primeira, foram identificados os alertas presentes no relatório, considerando sua categoria, risco e relação com componentes do sistema. Na segunda, os achados foram agrupados em eixos técnicos: proteção contra requisições indevidas, segurança de sessão, cabeçalhos HTTP de proteção, exposição de informações internas, segurança de transporte, dependências externas e governança de testes. Na terceira, cada alerta foi convertido em requisito de segurança. Na quarta, os requisitos foram reinterpretados como diretrizes para novos recursos do GAIO. Na quinta, as diretrizes foram organizadas em uma proposta de priorização evolutiva.

A conversão dos achados em diretrizes considerou quatro critérios principais: risco indicado no relatório, impacto funcional sobre os módulos do GAIO, esforço estimado de correção e potencial de transformação do alerta em requisito ou novo recurso. O risco foi utilizado para distinguir alertas que exigem tratamento prioritário daqueles que representam endurecimento complementar. O impacto funcional permitiu relacionar cada achado a componentes como autenticação, cadastro, sessão, Kanban, tarefas, painéis e administração. O esforço estimado auxiliou na separação entre ajustes de configuração, alterações de código e propostas de evolução arquitetural.

Também se adotou uma distinção entre melhoria técnica e diretriz evolutiva. As melhorias técnicas

correspondem a ações diretamente vinculadas aos alertas, como configurar cabeçalhos HTTP, proteger *cookies*, restringir *endpoints* e validar entradas. As diretrizes evolutivas, por sua vez, representam orientações mais amplas para novas versões da plataforma, incluindo critérios de aceite seguros, integração dos alertas ao *backlog*, priorização por NRP e acompanhamento contínuo por indicadores. Essa distinção permitiu interpretar o relatório OWASP ZAP não apenas como lista de vulnerabilidades, mas como instrumento de planejamento da evolução do GAIO.

A interpretação dos resultados baseou-se em boas práticas de segurança de aplicações web, especialmente nas recomendações da OWASP, no princípio de segurança por projeto e na lógica de desenvolvimento seguro (Souppaya et al., 2022). Também foi considerada a aderência dos requisitos ao contexto funcional do GAIO, de modo que as diretrizes propostas não se limitem a ajustes técnicos isolados, mas orientem a evolução do sistema como plataforma de gestão ágil, colaborativa e segura.

RESULTADOS E DISCUSSÃO

A análise do relatório OWASP ZAP permite observar que o GAIO não apresentou alertas de risco alto no escopo avaliado. Esse resultado é positivo, mas não elimina a necessidade de evolução da segurança. Os alertas de risco médio, baixo e informativo revelam fragilidades e oportunidades de endurecimento que podem ser convertidas em requisitos não funcionais e em novos recursos internos de gestão de segurança.

A principal contribuição do relatório, portanto, não está apenas em apontar correções técnicas, mas em fornecer uma base objetiva para orientar a evolução do GAIO. Em vez de tratar cada alerta como uma ocorrência isolada, este trabalho propõe que os achados sejam usados como diretrizes para novos recursos, regras de desenvolvimento, critérios de aceite e requisitos de segurança aplicáveis a futuras versões da plataforma.

Diretriz 1 - Segurança por padrão em novos recursos

Todo novo recurso do GAIO deve ser concebido com segurança desde sua definição inicial. Isso significa que funcionalidades como criação de tarefas, alteração de status no *Kanban*, filtros, comentários, anexos, gestão de usuários, gamificação, painéis gerenciais e priorização por NRP devem incluir requisitos mínimos de autenticação, autorização, validação de entradas, proteção de sessão e registro de ações.

Essa diretriz deriva especialmente dos alertas relacionados à ausência de *Anti-CSRF*, gerenciamento de sessão e autenticação. Se o sistema possui fluxos de cadastro, login e manipulação de tarefas, qualquer novo recurso que altere estado da aplicação deve ser protegido contra requisições indevidas e deve validar se o usuário possui permissão para executar a operação.

Como requisito de segurança, recomenda-se que toda funcionalidade que crie, edite, exclua ou altere informações no GAIO possua proteção *Anti-CSRF*, validação no servidor, controle de autorização e registro de auditoria (OWASP Foundation, s.d.b).

Diretriz 2 - Backlog de segurança integrado ao GAIO

Os achados do relatório devem ser convertidos em itens de *backlog* dentro do próprio GAIO. Dessa forma, a plataforma pode ser utilizada para gerenciar sua própria evolução técnica. Cada alerta identificado por ferramentas de segurança pode gerar uma tarefa com tipo, prioridade, risco, responsável, prazo, evidência, recomendação e status de correção.

Essa diretriz transforma o relatório OWASP ZAP em insumo operacional para o processo de gestão. Em vez de manter os resultados em um documento externo, o GAIO pode criar um módulo ou funcionalidade de "Gestão de Vulnerabilidades", permitindo que alertas sejam registrados, acompanhados e priorizados dentro da própria ferramenta.

Esse novo recurso poderia conter os seguintes campos: identificador do alerta, tipo de vulnerabilidade, nível de risco, nível de confiança, endereço afetado - URL (em inglês, *Uniform Resource Locator*: localizador uniforme de recursos), evidência, requisito relacionado, responsável pela correção, status, data de identificação, data de correção, data de reteste e observação técnica.

Diretriz 3 - Priorização de requisitos de segurança por NRP

O GAIO já possui proposta de uso do NRP - *Next Release Problem* - para priorização de versões, abordagem relacionada ao problema de seleção de requisitos para uma próxima versão de software (Bagnall et al., 2001). Esse mecanismo pode ser ampliado para priorizar requisitos de segurança. Assim, o sistema não deve tratar apenas funcionalidades de negócio como candidatas a uma próxima versão, mas também requisitos não funcionais derivados de segurança, desempenho, usabilidade, acessibilidade e governança.

A priorização pode considerar critérios como risco técnico, impacto funcional, esforço estimado, dependência arquitetural, criticidade do módulo afetado, facilidade de correção e valor institucional da melhoria.

Com base no relatório analisado, requisitos como implementação de *Anti-CSRF*, configuração de CSP, proteção contra *clickjacking* e restrição do *Spring Actuator* devem receber prioridade alta, pois estão

classificados como risco médio e afetam a superfície de segurança da aplicação. Já requisitos como HSTS, *X-Content-Type-Options* e *SameSite* podem ser tratados como endurecimento de prioridade média, por representarem ajustes importantes de configuração e defesa em profundidade.

Diretriz 4 - Módulo de requisitos não funcionais

A evolução do GAIO pode incluir um módulo específico para gestão de requisitos não funcionais. Esse módulo permitiria registrar requisitos de segurança, desempenho, confiabilidade, disponibilidade, acessibilidade, privacidade e auditabilidade. Com isso, o sistema ampliaria seu escopo para além da gestão de tarefas funcionais, tornando-se também uma ferramenta de governança da qualidade do software.

No caso da segurança, cada requisito não funcional poderia ser vinculado a evidências de teste, recomendações técnicas, riscos mitigados e critérios de aceite. Por exemplo, o requisito "configurar *Content Security Policy*" poderia possuir critérios como: política aplicada em todas as páginas públicas e autenticadas; bloqueio de fontes não autorizadas; validação em modo relatório; ausência de quebra funcional; e novo teste automatizado sem recorrência do alerta (OWASP Foundation, s.d.a).

Esse módulo também poderia apoiar a integração entre segurança e gestão ágil, permitindo que requisitos técnicos sejam visualizados em quadros *Kanban*, priorizados por versão e acompanhados por indicadores de evolução.

Diretriz 5 - Endurecimento de cabeçalhos HTTP como requisito obrigatório

O relatório identificou ausência de cabeçalhos de segurança importantes, como *Content Security Policy*, *HTTP Strict Transport Security* (HSTS), *X-Content-Type-Options* e cabeçalhos de proteção contra *clickjacking*. Esses achados indicam a necessidade de uma diretriz técnica para que todos os ambientes do GAIO adotem cabeçalhos HTTP de proteção como requisito obrigatório.

Essa diretriz deve abranger páginas públicas, páginas autenticadas, rotas administrativas, APIs (em inglês, *Application Programming Interfaces*: interfaces de programação de aplicações) e recursos estáticos. O objetivo é reduzir a superfície de ataque associada à injeção de scripts, à interceptação de comunicação, à interpretação indevida de conteúdo e à incorporação maliciosa de páginas em frames externos.

Como requisito, toda nova funcionalidade web do GAIO deve ser publicada com política mínima de cabeçalhos de segurança, incluindo *Content Security Policy*, *X-Frame-Options* ou *frame-ancestors*, *X-Content-Type-Options*, *HTTP Strict Transport Security* e diretivas adequadas de *cache* (OWASP Foundation, s.d.c).

Diretriz 6 - Gestão segura de sessão e cookies

O relatório evidencia uso de JSESSIONID com atributos *Secure* e *HttpOnly*, o que representa uma configuração positiva. Entretanto, também aponta ausência do atributo *SameSite*, atributo relevante para o endurecimento da gestão de *cookies* e sessão em aplicações web (OWASP Foundation, s.d.c).

Todo novo recurso que dependa de sessão autenticada deve observar regras de expiração, renovação, invalidação no *logout*, proteção contra reutilização indevida e restrição de envio de *cookies* em requisições entre sites. O atributo *SameSite* deve ser configurado de acordo com o fluxo funcional da aplicação, preferencialmente como *Lax* ou *Strict*, quando compatível com a experiência do usuário.

Além disso, recomenda-se que o GAIO possua um painel administrativo ou rotina técnica para monitorar sessões ativas, tentativas de autenticação, falhas de login e encerramento de sessão por inatividade.

Diretriz 7 - Restrição de endpoints técnicos e monitoramento interno

O alerta relacionado ao Spring Actuator indica oportunidade de revisão da exposição de *endpoints* técnicos. Recursos de diagnóstico, saúde da aplicação, métricas e ambiente são úteis para monitoramento, mas não devem ser expostos publicamente sem controle adequado (Spring, s.d.).

A diretriz proposta é separar claramente recursos públicos, recursos autenticados, recursos administrativos e recursos internos de monitoramento. *Endpoints* técnicos devem ser restritos por autenticação, autorização, rede, perfil de ambiente ou *gateway* seguro. Além disso, somente *endpoints* estritamente necessários devem permanecer habilitados em produção.

Como novo recurso, o GAIO pode evoluir para possuir um painel interno de saúde da aplicação, acessível apenas a administradores autorizados, exibindo indicadores controlados, sem exposição indevida de informações sensíveis sobre ambiente, configuração ou infraestrutura.

Diretriz 8 - Validação robusta de entradas

O alerta de estouro de buffer indica necessidade de fortalecimento das regras de validação de entradas. Ainda que o achado precise ser validado manualmente, ele aponta uma diretriz importante: todo campo de entrada do GAIO deve possuir limites, formatos aceitos, validação no cliente e validação obrigatória no servidor.

Essa diretriz deve abranger campos de cadastro, login, nomes de tarefas, descrições, comentários, etiquetas, filtros, parâmetros de busca, dados de gamificação, configurações de equipe e parâmetros utilizados pelo NRP.

Como requisito, todo novo formulário do GAIO deve possuir especificação de tamanho máximo, tipo esperado, obrigatoriedade, caracteres permitidos, tratamento de erro, mensagens de validação e sanitização adequada.

Diretriz 9 - Controle de dependências externas

O relatório identificou inclusão de JavaScript de domínio externo. Esse tipo de dependência pode ser funcionalmente útil, mas exige controle técnico. Bibliotecas externas devem ser avaliadas quanto à origem, integridade, atualização, necessidade real e impacto em segurança.

A diretriz proposta é que novos recursos do GAIO somente utilizem bibliotecas externas após avaliação explícita. Quando possível, as dependências devem ser versionadas, empacotadas localmente ou protegidas por mecanismos de integridade de sub-recurso. Também deve haver inventário de dependências, indicando nome, versão, origem, finalidade e módulo que utiliza.

Essa diretriz pode originar um recurso interno de inventário técnico, permitindo rastrear bibliotecas utilizadas pelo sistema, identificar versões obsoletas e associar dependências a riscos conhecidos.

Diretriz 10 - Segurança incorporada à esteira de desenvolvimento

O relatório OWASP ZAP não deve ser visto como avaliação única. A evolução do GAIO deve incorporar testes automatizados de segurança à rotina de desenvolvimento, de modo que cada nova versão seja submetida a varreduras, análise de cabeçalhos, testes de autenticação, testes de sessão e validação de regressão.

Essa diretriz implica criar um fluxo contínuo de segurança, no qual os alertas são coletados, classificados, registrados no *backlog*, atribuídos a responsáveis, corrigidos, retestados e documentados. Com isso, o GAIO passa a operar sob uma lógica de melhoria contínua.

Como novo recurso, o sistema pode incorporar indicadores de segurança, tais como: quantidade de alertas por risco, tempo médio de correção, percentual de alertas corrigidos, reincidência por tipo de vulnerabilidade, módulos mais afetados e evolução histórica entre versões.

Proposta de requisitos para novas versões do GAIO

A partir das diretrizes apresentadas, o Quadro 1 reúne os requisitos de segurança propostos para orientar a evolução da plataforma em versões futuras.

Quadro 1. Requisitos de segurança propostos para evolução do GAIO com base no relatório OWASP ZAP.

Código	Requisito proposto	Origem no relatório	Prioridade
RS-01	Implementar proteção *Anti-CSRF* em todas as operações que alterem estado	Ausência de *tokens Anti-CSRF*	Alta
RS-02	Configurar política *Content Security Policy* para páginas públicas e autenticadas	*CSP Header Not Set*	Alta
RS-03	Bloquear incorporação indevida de páginas do GAIO em *frames* externos	*Missing Anti-clickjacking Header*	Alta
RS-04	Restringir *endpoints* do *Spring Actuator* em produção	*Spring Actuator Information Leak*	Alta
RS-05	Configurar *SameSite* nos *cookies* de sessão	*Cookie without SameSite Attribute*	Média
RS-06	Configurar HSTS para reforço do uso de HTTPS	*Strict-Transport-Security Header Not Set*	Média
RS-07	Configurar *X-Content-Type-Options* com valor *nosniff*	*X-Content-Type-Options Header Missing*	Média

RS-08	Revisar *cache* de páginas autenticadas e dados sensíveis	Re-examine *Cache-control Directives*	Média
RS-09	Definir política de uso de bibliotecas JavaScript externas	*Cross-Domain JavaScript Source File Inclusion*	Média
RS-10	Criar módulo de gestão de vulnerabilidades no GAIO	Consolidação dos alertas do relatório	Alta
RS-11	Integrar alertas de segurança ao *backlog* e ao *Kanban*	Uso do GAIO para gestão da própria evolução	Alta
RS-12	Priorizar requisitos de segurança por NRP	Evolução do mecanismo de priorização	Alta
RS-13	Criar indicadores de segurança por versão	Governança técnica e melhoria contínua	Média
RS-14	Definir critérios de aceite de segurança para novos recursos	Segurança por padrão	Alta
RS-15	Implantar rotina de reteste automatizado após correções	Validação contínua	Média

Os requisitos demonstram que o relatório pode orientar tanto melhorias técnicas imediatas quanto novos recursos de gestão. Assim, o GAIO pode evoluir em duas frentes complementares: a primeira consiste no endurecimento da aplicação; a segunda consiste na criação de funcionalidades internas para gerenciar requisitos de segurança, vulnerabilidades, testes e priorização técnica.

A análise dos requisitos propostos evidencia que os achados do relatório podem orientar a evolução do GAIO em duas dimensões complementares. A primeira dimensão é corretiva e está relacionada ao endurecimento técnico da aplicação, contemplando ajustes como proteção *Anti-CSRF*, configuração de *Content Security Policy*, uso de HSTS, definição de *SameSite* em *cookies*, restrição do *Spring Actuator* e adoção de cabeçalhos HTTP de segurança. Essa dimensão busca reduzir a superfície de ataque e corrigir fragilidades identificadas na varredura automatizada.

A segunda dimensão é evolutiva e está relacionada à criação de novos recursos de governança da segurança dentro do próprio GAIO. Nesse caso, os alertas deixam de ser tratados como itens externos ao processo de gestão e passam a ser incorporados ao fluxo da plataforma. Isso permite que vulnerabilidades sejam registradas como tarefas, acompanhadas em *Kanban*, vinculadas a responsáveis, priorizadas por NRP e avaliadas por indicadores.

Essa proposta amplia a contribuição do trabalho porque demonstra que uma ferramenta de gestão ágil pode incorporar requisitos de segurança como parte de sua lógica operacional. Em vez de separar gestão de projetos e segurança da aplicação, o modelo proposto aproxima essas dimensões. Assim, a segurança passa a ser planejada, acompanhada, priorizada e validada dentro do mesmo ambiente utilizado para organizar as atividades de desenvolvimento.

Proposta de novos recursos para o GAIO

Além dos requisitos técnicos, a análise permite propor novos recursos funcionais diretamente relacionados à segurança.

O primeiro recurso proposto é o módulo de gestão de vulnerabilidades. Esse módulo permitiria cadastrar achados oriundos de ferramentas como OWASP ZAP, classificá-los por risco, vincular evidências, atribuir responsáveis e acompanhar a correção por status.

O segundo recurso é a conversão automática de alertas em tarefas. Relatórios de segurança poderiam ser importados para o GAIO e transformados em cartões no *Kanban*, permitindo acompanhamento visual da correção.

O terceiro recurso é o painel de indicadores de segurança. Esse painel apresentaria métricas como alertas por risco, alertas por módulo, tempo médio de correção, vulnerabilidades reincidentes e evolução entre versões.

O quarto recurso é a priorização de segurança por NRP. Nesse caso, o algoritmo de priorização do GAIO passaria a considerar requisitos de segurança juntamente com funcionalidades de negócio, equilibrando valor, risco, esforço e criticidade.

O quinto recurso é a biblioteca de critérios de aceite seguros. Cada tipo de funcionalidade - login, cadastro, tarefas, anexos, filtros, administração,

gamificação e NRP - possuiria critérios mínimos de segurança a serem verificados antes da entrega.

O sexto recurso é o histórico de retestes. Após cada correção, o sistema poderia registrar nova evidência de varredura, data de reteste, resultado e situação final do alerta.

Esses recursos fortalecem o GAIO como plataforma de gestão ágil e, ao mesmo tempo, ampliam seu escopo para a governança da segurança no ciclo de vida do software.

CONCLUSÃO

Este trabalho apresentou diretrizes para evolução do GAIO a partir da análise de um relatório automatizado de segurança web gerado pelo OWASP ZAP. A proposta não se limita à correção de vulnerabilidades pontuais, mas utiliza os achados como base para definição de novos requisitos de segurança, novos recursos funcionais e critérios de priorização evolutiva.

A análise demonstrou que alertas como ausência de *Anti-CSRF*, ausência de *Content Security Policy*, falta de proteção contra *clickjacking*, exposição relacionada ao *Spring Actuator*, ausência de *SameSite*, ausência de HSTS e ausência de *X-Content-Type-Options* podem ser convertidos em requisitos não funcionais e diretrizes de desenvolvimento seguro.

Conclui-se que o GAIO pode evoluir para incorporar segurança como parte de sua própria lógica de gestão. Para isso, recomenda-se que a plataforma passe a tratar vulnerabilidades, requisitos de segurança e evidências de testes como itens gerenciáveis em seu *backlog*, integrando-os ao Kanban, aos indicadores, ao histórico de evolução e ao processo de priorização por NRP.

Como trabalhos futuros, recomenda-se implementar os requisitos propostos, realizar nova varredura após as correções, comparar os resultados antes e depois da intervenção, criar um módulo de gestão de vulnerabilidades no GAIO e integrar testes automatizados de segurança à rotina de desenvolvimento da plataforma.

AGRADECIMENTOS

Os autores agradecem à instituição de ensino pelo apoio, aos colaboradores envolvidos no desenvolvimento e avaliação do GAIO e às comunidades que disponibilizam ferramentas e boas práticas para segurança em aplicações web.

REFERÊNCIAS

BAGNALL, A. J.; RAYWARD-SMITH, V. J.; WHITTLEY, I. M. The next release problem. Information and Software Technology, v. 43, p. 883-890, 2001.

BECK, K. et al. Manifesto for Agile Software Development. 2001.

ISO/IEC/IEEE. ISO/IEC/IEEE 29148:2018 - Requirements engineering. ISO, Geneva, 2018.

OWASP FOUNDATION. Content Security Policy Cheat Sheet. OWASP Cheat Sheet Series, s.d.a.

OWASP FOUNDATION. Cross-Site Request Forgery Prevention Cheat Sheet. OWASP Cheat Sheet Series, s.d.b.

OWASP FOUNDATION. HTTP Security Response Headers Cheat Sheet. OWASP Cheat Sheet Series, s.d.c.

OWASP FOUNDATION. OWASP Top 10: A05:2021 - Security Misconfiguration. OWASP Foundation, 2021.

OWASP FOUNDATION. Secure Product Design Cheat Sheet. OWASP Cheat Sheet Series, s.d.d.

SOMMERVILLE, I. Software Engineering. Pearson, Boston, 2016.

SOUPPAYA, M.; SCARFONE, K.; DODSON, D. Secure Software Development Framework (SSDF) Version 1.1. NIST, Gaithersburg, 2022.

SOUSA, D. C.; GAMA, T. D. C. Q.; MARTINS, R. M. G. GAIO 0.1: Ferramenta de Gestão Ágil Integrada de Operações com Visualização Kanban e Gamificação. VI CoBICET - Congresso Brasileiro Interdisciplinar em Ciência e Tecnologia, 2025.

SPRING. Spring Boot Reference Documentation: Production-ready Features. Spring Documentation, s.d.

ZAP DEV TEAM. ZAP Documentation. Zed Attack Proxy Documentation, s.d.