Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

REVISTA SOCIEDADE CIENTÍFICA, VOLUME 7, NÚMERO 1, ANO 2024

ARTIGO ORIGINAL

Segurança da informação na pesquisa científica

Marcelo Pontes Rodrigues[1]; Fernando Marques Borges[2]; Maria do Socorro Pires e Cruz[3]

Como Citar:
RODRIGUES, Marcelo Pontes; BORGES Fernando Marque;E CRUZ Maria do Socorro Pires. Segurança da informação na pesquisa científica.. Revista Sociedade Científica, vol.7, n.1, p.1952-1964, 2024. https://doi.org/10.61411/rsc202436617

DOI: 10.61411/rsc202436617

Área do conhecimento: Ciências da Saúde.

Palavras-chaves: Segurança; Software; UUID; API; REST.

Publicado: 15 de abril de 2024.

Resumo

Este artigo destaca a importância da segurança e eficiência de sistemas computacionais nas instituições de ensino e pesquisa devido ao aumento no volume de dados gerados. As interfaces de Programação de Aplicativos (APIs, do inglês *Application Programming Interfaces*) baseadas em REST (*Representation State Transfer*) são utilizadas para automatizar a integração de softwares, principalmente na pesquisa científica. A segurança destas APIs é crucial, pois configurações inadequadas podem torná-las vulneráveis a ataques cibernéticos, resultando em vazamento de dados. Neste sentido, o desenvolvimento orientado a testes de software e o uso de UUIDs (Identificadores Únicos Universais) devem ser práticas essenciais a serem consideradas no desenvolvimento de sistemas computacionais nesta área. O propósito do artigo é destacar vantagens do uso de UUIDs (*Universally Unique Identifier*) nas URLs (*Uniform Resource Locator*) das APIs como recurso adicional visando reduzir o risco de vazamento acidental ou intencional de informações, de modo a tornar as identificações exclusivas e, desta forma, mais seguras.

1. Introdução

A pesquisa científica experimentou um notável crescimento nas últimas décadas, impulsionada pelo emprego de *Workflows*, cuja definição adota uma abordagem específica conforme delineada pelo *Workflow Management Coalition* (WfMC) [20].

[1]Programa de Pós-Graduação em Tecnologias Aplicadas a Animais de Interesse Regional–UFPI, Teresina, Brasil. ✉
[2]Analista de cibesegurança, Brasília, Brasil. ✉
[3]Programa de Pós-Graduação em Tecnologias Aplicadas a Animais de Interesse Regional–UFPI, Teresina, Brasil. ✉

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

Este conceito denota a automatização, total ou parcial, de processos de negócios, nos quais documentos, informações ou tarefas são transferidos entre participantes, seguindo um conjunto de regras geralmente aplicadas aos fluxos de processos no contexto corporativo. Embora sua aplicação inicial tenha se manifestado predominantemente em ambientes empresariais, os *Workflows* são atualmente empregados de maneira difundida em contextos científicos, notadamente em laboratórios de pesquisa universitários e em grandes complexos industriais [17].

A maioria dos processos experimentais em produção apresenta um aumento significativo no volume de dados manipulados, devido à complexidade intrínseca dos problemas a serem analisados e à necessidade de colaboração entre grupos de pesquisa geograficamente distribuídos. A e-Science, voltada para a ciência em larga escala, em conjunto com os novos experimentos envolvendo a manipulação de dados em diversas tecnologias computacionais, passou a ser denominada experimentação científica *in silico*. Os *Workflows* Científicos (WfC) desempenham um papel crucial na e-Science, visando representar processos encadeados, também chamados de atividades, e a combinação de dados consumidos ou produzidos por essas atividades [17]. Eles oferecem uma abordagem alternativa para analisar o problema do sequenciamento estruturado de passos necessários no desenvolvimento de experimentos científicos, destacando a importância das garantias de reprodutibilidade e confiabilidade desses experimentos [1].

Atualmente, os WfC compartilham características semelhantes aos *workflows* de negócio, mas possuem particularidades relacionadas à especificação e implementação, como suas fases de composição, execução e análise[3, 11], as quais podem demandar meses para sua conclusão e resultar em descobertas planejadas ou inesperadas. Os resultados obtidos devem ser registrados para avaliação dos experimentos, compartilhamento de dados, garantia de reprodução e para fins de auditoria e segurança [9]. As ferramentas mais utilizadas na construção de WfC são denominadas Sistemas de Gerência de *Workflow* Científico (SGWfC) [11], proporcionando suporte à construção

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

dos fluxos (composição), à captura de metadados, bem como à execução, coleta, análise e tratamento da proveniência dos dados e processos [19].

A disponibilidade de acesso a ambientes de alta tecnologia a custos relativamente baixos facilita a integração e o armazenamento de dados distribuídos. Contudo, destaca-se a crescente importância da segurança desses dados, muitos dos quais são disponibilizados em tecnologias web. Nesse contexto, é imperativo considerar aspectos como: (i) a relevância da segurança e eficiência de sistemas computacionais e (ii) a importância do uso de Interfaces de Programação de Aplicativos (APIs, do inglês *Application Programming Interfaces*) baseadas em REST para o compartilhamento seguro de informações.

No cenário contemporâneo, as tecnologias web permeiam diversas esferas de nossas vidas diárias. À medida que a dependência de aplicações web se intensifica, as APIs desempenham um papel crucial, atuando como o elo que viabiliza a comunicação eficaz entre distintos programas. No entanto, o aumento da complexidade das APIs também acarreta um crescimento nos riscos de segurança associados a elas. Nesse contexto, a segurança das APIs emerge como uma área de grande relevância na comunidade científica.

As APIs desempenham um papel fundamental em diversos processos, proporcionando benefícios como o compartilhamento de dados entre sistemas distintos, a otimização da execução de tarefas e o aumento da produtividade em ambientes distribuídos.

A evolução tecnológica tem incorporado cada vez mais as APIs na arquitetura de softwares modernos e em diversas soluções computacionais. Entretanto, o aumento do uso dessa tecnologia também resulta no crescimento dos vetores de ataque [5,6], destacando a importância da adoção de padrões de segurança rigorosos no desenvolvimento e manutenção dessas interfaces [4, 6].

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

As metodologias de desenvolvimento de software consistem em conjuntos de técnicas e métodos empregados para criar soluções de sistemas, visando organizar as equipes de trabalho para o desenvolvimento eficiente das funções de um programa. A ausência de uma metodologia clara no desenvolvimento de produtos ou serviços torna o processo mais complexo, resultando em problemas, atrasos, erros e resultados indesejados. Existem dois grupos principais de metodologias de desenvolvimento de software: as ágeis e as tradicionais.

As metodologias tradicionais caracterizam-se pela definição total e rígida dos requisitos no início dos projetos de engenharia de software, com ciclos de desenvolvimento pouco flexíveis e resistentes às mudanças, ao contrário das metodologias ágeis, que têm obtido mais destaque devido à sua flexibilidade.

Uma das metodologias ágeis é o Desenvolvimento Orientado a Testes (TDD, do inglês *Test-Driven Development*), que substitui o fluxo tradicional de codificar e testar. Essa abordagem enfatiza o desenho da solução e seus testes, envolvendo a escrita interativa automatizada de maneira *Test-First*, onde o teste é projetado antes da implementação. Esse processo inverte o ciclo básico de desenvolvimento de código mais comum, proporcionando uma abordagem mais eficaz após reflexões sobre o ciclo tradicional 2].

Python destaca-se como uma das linguagens de programação mais apropriadas para a implementação de testes. O *framework* Pytest, fundamentado em Python, proporciona uma ampla gama de ferramentas para a criação de testes automatizados, sendo aplicável em projetos de diversas escalas[14].

No âmbito do desenvolvimento de APIs RESTful, o FastAPI surge como um dos *frameworks* relevantes, implementado em Python [7,16]. A integração desta tecnologia com o Pytest emerge como uma estratégia viável para o desenvolvimento aderente às melhores práticas de aplicações de WfC e ao compartilhamento de dados orientado a testes.

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

Diante desse contexto, este trabalho propõe-se a realizar uma introdução sucinta sobre a interseção entre segurança da informação, pesquisa científica e a adoção de novas tecnologias. O enfoque recai sobre a sinergia entre Python, o *framework* Pytest e o FastAPI, delineando uma abordagem que não apenas propicia eficácia no desenvolvimento de software, mas também se alinha às práticas recomendadas no contexto de *Workflows Científicos*.

2. Metodologia

Este estudo constitui uma parte integrante do projeto denominado "A Ciência de Dados em Estudos Epidemiológicos de Leishmaniose", em fase de desenvolvimento no âmbito do Programa de Pós-graduação em Tecnologias Aplicadas a Animais de Interesse Regional (PPGTAIR) da Universidade Federal do Piauí (UFPI).

Configurando-se como um estudo descritivo [13], sua abordagem visa a análise do volume de dados gerados em pesquisas científicas, a introdução dos conceitos relacionados à segurança e eficiência de sistemas computacionais, a descrição da relevância do emprego de APIs baseadas em REST, bem como a exploração dos temas relacionados ao desenvolvimento orientado a testes e à utilização de UUID no *framework* FASTAPI.

As informações contidas neste resumo expandido foram obtidas por meio de pesquisa-ação. Conforme Thiollent, essa abordagem pode ser considerada uma variação de estudo de caso, diferindo na medida em que o pesquisador não é apenas um observador, mas também um participante ativo na implementação de ações [13].

Dada a participação direta dos autores deste resumo expandido na concepção e execução das ações relacionadas à infraestrutura computacional, destinada a viabilizar a utilização de novas tecnologias, compartilhamento de dados e integração de sistemas existentes, prevê-se eficácia na consecução dos objetivos almejados, especificamente na prevenção, controle e análise de riscos associados à leishmaniose.

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

3. Desenvolvimento e discussão

As APIs desempenham um papel fundamental em diversos processos, proporcionando benefícios significativos, tais como o compartilhamento de dados entre sistemas distintos, otimização da execução de tarefas e aumento da produtividade em ambientes distribuídos.

A evolução tecnológica tem incorporado cada vez mais as APIs na arquitetura de softwares modernos e em variadas soluções computacionais. Contudo, o crescimento do uso dessa tecnologia também resulta no aumento de vetores de ataque [5,6], evidenciando a necessidade premente de adotar padrões de segurança rigorosos no desenvolvimento e na manutenção desse componente [4;6].

O OWASP API Security TOP 10 destaca diversas vulnerabilidades que devem ser consideradas como potenciais riscos. Essas vulnerabilidades abrangem desde falhas em mecanismos de autenticação e autorização até a exposição indevida de dados sensíveis. Tais fragilidades não apenas ameaçam a integridade dos sistemas que dependem desses serviços, mas também podem resultar em violações de dados e impactos negativos nas instituições [4,10].

A lista subsequente apresenta as dez principais vulnerabilidades de segurança de APIs, conforme definido pela OWASP em 2023.

API1:2023 - *Broken Object Level Authorization*: Os invasores podem explorar pontos de extremidade de APIs vulneráveis à quebra de autorização no nível de objeto, manipulando a ID de um objeto enviado na solicitação. As IDs de objeto podem ser qualquer coisa, desde inteiros sequenciais, UUIDs ou cadeias de caracteres genéricas. Independentemente do tipo de dados, eles são fáceis de identificar no destino da solicitação (parâmetros de caminho ou cadeia de caracteres de consulta), cabeçalhos de solicitação ou até mesmo como parte da carga útil da solicitação.

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

API2:2023 - *Broken Authentication*: O mecanismo de autenticação é um alvo fácil para os invasores, uma vez que está exposto a todos. Embora habilidades técnicas mais avançadas possam ser necessárias para explorar alguns problemas de autenticação, as ferramentas de exploração estão geralmente disponíveis.

API3:2023 - *Broken Object Property Level Authorization*: a exploração da exposição excessiva de dados é simples e geralmente é realizada farejando o tráfego para analisar as respostas da API, procurando exposição de dados confidenciais que não devem ser devolvidos ao usuário. A exploração, também, geralmente requer uma compreensão da lógica de negócios, das relações de objetos e da estrutura da API. A exploração da atribuição em massa é mais fácil nas APIs, já que, por *design*, elas expõem a implementação subjacente do aplicativo junto com os nomes das propriedades.

API4:2023 - *Unrestricted Resource Consumption*: a exploração requer solicitações de API simples. Várias solicitações simultâneas podem ser executadas a partir de um único computador local ou usando recursos de computação em nuvem. A maioria das ferramentas automatizadas disponíveis é projetada para causar "Negação de Serviço", ou DoS (*Denial of Service*).

API5:2023 - *Broken Function Level Authorization*: A exploração exige que o invasor envie chamadas de API legítimas para um ponto de extremidade de API ao qual ele não deve ter acesso como usuários anônimos ou usuários regulares e sem privilégios. Os pontos de extremidade expostos serão facilmente explorados.

API6:2023 - *Unrestricted Access to Sensitive Business Flows*: a exploração geralmente envolve entender o modelo de negócios apoiado pela API, encontrar fluxos de negócios confidenciais e automatizar o acesso a esses fluxos, causando danos aos negócios.

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

API8:2023 - *Security Misconfiguration*: os invasores geralmente tentam encontrar falhas não corrigidas, pontos de extremidade comuns, serviços executados com configurações padrão inseguras ou arquivos e diretórios desprotegidos para obter acesso não autorizado ou conhecimento do sistema. A maior parte disso é de conhecimento público e explorações podem estar disponíveis.

API9:2023 - *Improper Inventory Management*: os agentes de ameaças geralmente obtêm acesso não autorizado por meio de versões antigas da API ou pontos de extremidade deixados em execução sem patch e usando requisitos de segurança mais fracos. Em alguns casos, *exploits* (programa, ou pedaço de código, projetado para encontrar e tirar vantagem de uma falha de segurança ou vulnerabilidade de um aplicativo) estão disponíveis. Alternativamente, eles podem obter acesso a dados confidenciais por meio de uma 3ª parte com quem não há razão para compartilhar dados.

API10:2023 - *Unsafe Consumption of APIs*: explorar esse problema exige que invasores identifiquem e potencialmente comprometam outras APIs/serviços com os quais a API alvo está integrada. Normalmente, esta informação não está disponível publicamente ou a API/serviço integrado não é facilmente explorável.

Diante do potencial risco, é imperativo considerar a adoção de tecnologias específicas para mitigar as vulnerabilidades, destacando-se, por exemplo, o uso de Identificadores Únicos Universais (UUID, do inglês *Universally Unique Identifier*) e o desenvolvimento de sistemas orientados a testes. O UUID consiste em um identificador universalmente exclusivo empregado para identificar entidades no domínio da computação.

Compreendendo um número de 128 bits, é representado por 32 dígitos hexadecimais organizados em cinco grupos separados por hifens, seguindo a forma textual "8-4-4-4-12", totalizando 36 caracteres (por exemplo, "3d0ca315-aff9–4fc2-be61–3b76b9a2d798", com 32 caracteres alfanuméricos e 4 hifens) [8].

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

Em muitos casos, ao criar uma tabela em um banco de dados, adiciona-se uma coluna ID (ou similar) como chave primária, configurada como autoincremento. O ID autoincremento é um número inteiro que inicia em 1 e é incrementado em +1 sempre que um novo registro é inserido no banco de dados. No entanto, essa abordagem expõe o ID nas URLs das aplicações, sendo comum em APIs Rest, como exemplificado na Tabela 1.

Tabela 1 – Representação de uma estrutura de API REST

Verbo	URI	Ação
GET	/users	index
GET	/users/create	create
POST	/users	store
GET	/users/{id}	show
GET	/users/{id}/edit	edit
PUT/PATH	/users/{id}	update
DELETE	/users/{id}	destroy

Ao realizar a criação de um novo usuário em uma API que adota um sistema de ID incremental, verifica-se que um número específico é atribuído, como exemplificado pelo valor 56. Essa correlação direta entre o número atribuído e a quantidade total de usuários no sistema, neste caso, 56 usuários, evidencia uma possível lacuna de segurança. Uma observação adicional de relevância é que, ao alterar o número do ID dentro do intervalo de 1 a 56, torna-se viável a visualização dos dados de outros usuários.

A ausência de uma política robusta de segurança, com níveis de acesso adequados, pode resultar em comprometimento da segurança dos dados. Diante desse contexto, a adoção de UUIDs na API apresenta diversas vantagens, tais como descentralização na criação de identificadores únicos, facilidade na sincronização de

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

dados, a capacidade de ocultar a quantidade de registros criados em uma tabela do banco de dados e a dificuldade de manipulação das URLs 15].

Este resumo expandido culmina em um estudo conciso acerca da segurança da informação no contexto da pesquisa científica, materializado por meio do compartilhamento de código da API denominada SandFly. Esta API está acessível para consulta na URL "https://github.com/infopontes/sandfly", destacando a aplicação de diversas tecnologias, a saber: Python versão 3.11 como linguagem de programação; Poetry como ferramenta para gerenciamento de pacotes e ambiente virtual; Git para versionamento de código; Github para gestão e armazenamento das versões do código; Alembic para configuração e administração de migrações do banco de dados; e Docker para a criação de containers da aplicação e do banco de dados PostgreSQL [12].

A condução desse estudo visa ampliar as discussões relacionadas à segurança da informação e ao desenvolvimento de software seguro, proporcionando uma base sólida para o compartilhamento seguro de dados na pesquisa científica. Este trabalho não apenas oferece uma implementação prática dessas considerações, mas também busca promover uma reflexão mais abrangente sobre a interseção entre segurança da informação e práticas de desenvolvimento seguro em cenários científicos.

4. Considerações finais

A condução deste estudo evidenciou a relevância intrínseca da segurança e eficiência de sistemas computacionais no âmbito das instituições de ensino e pesquisa, especialmente diante do acréscimo significativo no volume de dados gerados.

O avanço contínuo das tecnologias e a crescente prática de compartilhamento de dados têm propiciado a ampla adoção de APIs modernas em diversas soluções computacionais. Nesse contexto, destaca-se a imperatividade do desenvolvimento de software orientado a testes como uma estratégia fundamental para mitigar as vulnerabilidades de ataques que potencialmente afetam essas aplicações.

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

Através dessa abordagem, almeja-se não apenas ampliar a conscientização acerca da necessidade de aprofundamento e fomento no tema em questão, mas também compartilhar conhecimento para aprimorar a segurança da informação no domínio da pesquisa científica.

Assim, este trabalho, além de sinalizar para a importância intrínseca da segurança cibernética, busca, ainda, efetivamente contribuir para a promoção de práticas seguras e eficientes no âmbito das instituições acadêmicas e de pesquisa.

5. Declaração de direitos

6. Referências

1. Altintas, Ilkay; Barney, Oscar; Jaeger-Frank, Efrat. Provenance collection support in the kepler scientific workflow system. In International Provenance and Annotation Workshop, p. 118-132, 2006.
2. Beck, Kent. Test driven development: By example. Addison-Wesley Professional, 2022.
3. Cruz, Sérgio Manuel Serra da; Campos, Maria Luiza Machado; Mattoso, Marta. Towards a taxonomy of provenance in scientific workflow management systems. In: IEEE. 2009 Congress on Services-I, p. 259–266, 2009.
4. Díaz-Rojas, Josué Alejando et al. Web api security vulnerabilities and mitigation mechanisms: A systematic mapping study. In: IEEE. 2021 9th International Conference in Software Engineering Research and Innovation (CONISOFT), p. 207–218, 2021.

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

5. Filho, Ailton Santos; Feitosa, Eduardo Luzeiro. Dificuldades em detectar ataques a web apis em arquivos de log. ResearchGate, 2020.
6. Idris, Muhammad; Syarif, Iwan; Winarno, Idris. Web application security education platform based on owasp api security project. EMITTER International Journal of Engineering Technology, p. 246–261, 2022.
7. Lathkar, Malhar. Introduction to fastapi. In: High-Performance Web Apps with FastAPI: The Asynchronous Web Framework Based on Modern Python: Springer, p. 1–28, 2023.
8. Leach, Paul; Mealling Michael; Salz, Rich. A universally unique identifier (uuid) urn namespace. RFC Editor, 2005.
9. Leal, André Luiz de Castro; Cruz, Sérgio Manuel Serra da. Transparência em experimentos científicos apoiados em proveniência: Uma perspectiva para workflows científicos transparentes. 2014.
10. Matos, Mariéli Fernandes. Desenvolvimento de um software para aquisição de dados de uma estação de recarga de veículos elétricos utilizando a metodologia de desenvolvimento guiados a testes. Trabalho de Conclusão de Curso (TCC) da IFSC, p. 12-16, 2021.
11. Mattoso, Marta; Cruz, Sérgio Manuel Serra da. Gerência de workflows científicos: oportunidades de pesquisa em bancos de dados. In: Proceedings of the 23rd Brazilian symposium on Databases. p. 313–314, 2008.
12. MENDES, Eduardo. FastAPI do ZERO. Disponível em: <https://fastapidozero.dunossauro.com/>. Acesso em: 10 set. 2023.
13. Moresi, Eduardo. Metodologia da pesquisa. Universidade Católica de Brasília, v. 108, n. 24, p. 5, 2003.
14. Oliveira, Bruno. pytest Quick Start Guide: Write better Python code with simple and maintainable tests. Packt Publishing Ltd, 2018.

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

15. Paixão, João Roberto da. O que é UUID? Por que usá-lo? Disponível em: <https://medium.com/trainingcenter/o-que-%C3%A9-uuid-porque-us%C3%A1-lo-ad7a66644a2b>. Acesso em: 29 out. 2023.
16. Palacio, Virginia Fernández. Desarrollo de gateway de seguridad en python con fastapi. UCrea, 2022.
17. Rodrigues, Marcelo Pontes. SeqRibbonHIV – Sistema integrado de acompanhamento epidemiológico, clínico, laboratorial e terapêutico de pacientes portadores do HIV/AIDS. Dissertação (Mestrado) - Instituto Oswaldo Cruz, 2010.
18. Sampaio, José Pedro Gomes. Introdução de Desenvolvimento de Software Orientado aos Testes. Dissertação do Instituto Superior de Engenharia do Porto, 2019.
19. Simmhan, Yogesh; Plale, Beth; Gannon, Dennis. A survey of data provenance in e-science. ACM Sigmod Record, ACM New York, NY, USA, v. 34, n. 3, p. 31–36, 2005.
20. Workflow Management Coalition. Disponível em: <https://wfmc.org/>. Acesso em: 28 out. 2023.