ESCOLA DE APERFEIÇOAMENTO DE OFICIAIS DA AERONÁUTICA
DIVISÃO DE ENSINO
CURSO DE APERFEIÇOAMENTO DE OFICIAIS 3º/2024

CARLOS **RENATO** DE ANDRADE FIGUEIRÊDO, Cap Av

Implantação de ferramentas de análise de vulnerabilidades no CCA-SJ para incremento da proteção cibernética na FAB

Rio de Janeiro
2024

ESCOLA DE APERFEIÇOAMENTO DE OFICIAIS DA AERONÁUTICA
DIVISÃO DE ENSINO
CURSO DE APERFEIÇOAMENTO DE OFICIAIS 3º/2024

CARLOS **RENATO** DE ANDRADE FIGUEIRÊDO, Cap Av

Implantação de ferramentas de análise de vulnerabilidades no CCA-SJ para incremento da proteção cibernética na FAB

Trabalho de conclusão de curso apresentado à Escola de Aperfeiçoamento de Oficiais da Aeronáutica como requisito parcial para aprovação no Curso de Pós-Graduação *Lato Sensu* em Liderança com Ênfase em Gestão no COMAER.
Linha de Pesquisa: Ciência, Tecnologia e Inovação
Orientador: Márcio Henrique Teixeira de Souza, Ten Cel Av

Rio de Janeiro
2024

CARLOS **RENATO** DE ANDRADE FIGUEIRÊDO, Cap Av

Implantação de ferramentas de análise de vulnerabilidades no CCA-SJ para incremento da proteção cibernética na FAB

Trabalho de conclusão de curso apresentado ao Curso de Aperfeiçoamento de Oficiais da Escola de Aperfeiçoamento de Oficiais da Aeronáutica.

Aprovado por:

_____

Presidente, Márcio Henrique Teixeira de Souza, Ten Cel Av - EAOAR

_____

Eduardo Mendes Marcondes, Maj Av - EAOAR

Rio de Janeiro

2024

RESUMO

A proteção cibernética no contexto militar é de extrema importância estratégica, especialmente no desenvolvimento de *software* para sistemas críticos da Força Aérea Brasileira (FAB). A proteção de dados e a integridade dos sistemas de informação são essenciais para garantir a eficácia das operações militares e a defesa nacional. Nesse cenário, o Centro de Computação da Aeronáutica de São José dos Campos (CCA-SJ) desempenha um papel fundamental no desenvolvimento e gerenciamento de sistemas que suportam a infraestrutura de Comando e Controle da FAB, além de outros sistemas de cunho operacional e corporativo. Contudo, o processo atual de desenvolvimento de *software* apresenta fragilidades, como a falta de ferramentas automatizadas para identificar vulnerabilidades de segurança nas fases iniciais do ciclo de desenvolvimento. Este trabalho defende a implantação de ferramentas de análise estática e dinâmica de vulnerabilidades de código na esteira de desenvolvimento de *software* do CCA-SJ, trazendo dois principais benefícios: otimizar o tempo de desenvolvimento e aumentar a segurança através da adição de uma camada extra de verificação. Consequentemente, a integração dessas ferramentas permitirá identificar e corrigir vulnerabilidades nas fases iniciais, reduzindo retrabalhos e o tempo gasto em coordenações com o Centro de Computação da Aeronáutica de Brasília (CCA-BR). Ademais, a união as técnicas *white-box* e *black-box* ampliará a cobertura dos testes de segurança, resultando em *softwares* mais seguros e robustos. Adicionalmente, essa melhoria fortalecerá a posição estratégica da FAB, assegurando a superioridade de informações e reforçando a proteção cibernética, essencial para a defesa da soberania nacional.

Palavras-chave: segurança de *software*; análise estática de vulnerabilidades; análise dinâmica de vulnerabilidades.

1 INTRODUÇÃO

O Centro de Computação da Aeronáutica de São José dos Campos (CCA-SJ) é responsável pelo desenvolvimento e gestão de sistemas operacionais e corporativos da Força Aérea Brasileira (FAB), incluindo os sistemas críticos de Comando e Controle (C2) e o sistema de gestão documental (SIGADAER), que é utilizado por mais de 200 Organizações Militares. Devido à importância desses sistemas, é essencial que os profissionais de Tecnologia da Informação da FAB adotem processos eficientes e seguros para garantir a proteção das informações sensíveis (Brasil, 2020).

Diante desse cenário, o CCA-SJ deve revisar continuamente o processo de desenvolvimento de *software* para torná-lo mais eficiente e seguro. Para isso, é fundamental adotar uma abordagem que inclua a segurança desde o início do ciclo de desenvolvimento, além de promover ajustes na esteira de desenvolvimento. Essa esteira é composta por práticas que visam integrar, testar e implantar *software* de forma contínua e confiável, garantindo que melhorias sejam aplicadas de maneira estruturada (Amaro; Pereira; Da Silva, 2022).

Para Zaydi e Nassereddine (2022), integrar práticas de segurança em todo o ciclo de vida do *software* permite descobrir vulnerabilidades precocemente. Khan *et al.* (2022, apud Abiona *et al.*, 2024) afirmam que, historicamente, a segurança é considerada uma etapa isolada no ciclo de vida do desenvolvimento de *software*, muitas vezes inserida posteriormente ou delegada a uma equipe de segurança distinta. Abiona *et al*. (2024), alertam que esse método pode resultar na negligência de vulnerabilidades e riscos de segurança ou no tratamento tardio desses problemas durante o desenvolvimento, o que pode acarretar falhas de segurança custosas e com resolução lenta.

No contexto do CCA-SJ, observa-se uma situação similar àquela descrita por Khan *et al.* (2022, apud Abiona *et al.*, 2024), na qual o processo de desenvolvimento não integra mecanismos de análise de segurança desde o início. Como resultado, embora as etapas de desenvolvimento e testes sejam executadas conforme planejado, a identificação de vulnerabilidades de segurança ocorre apenas em uma etapa posterior, quando o código já está finalizado.

Outra unidade da FAB conduz a verificação final, que é o Centro de Computação da Aeronáutica de Brasília (CCA-BR). Este Centro realiza a análise de vulnerabilidades do *software* por meio de testes *black-box* (sem acesso ao código-fonte), o que pode deixar algumas falhas ocultas. Quando se identificam problemas, o *software* é retornado ao CCA-SJ para as

devidas correções e, posteriormente, reenviado ao CCA-BR para uma nova rodada de verificação. Esse ciclo é repetido quantas vezes for necessário.

Um exemplo prático dessa situação ocorreu durante o desenvolvimento do FABSign, um sistema de assinatura digital. Após a conclusão da primeira versão do *software*, ele foi encaminhado para o CCA-BR para a realização dos testes de segurança. Cerca de quatro meses depois, o Centro de Brasília enviou um relatório com a identificação de vulnerabilidades. Algumas delas, entretanto, poderiam ser detectadas e corrigidas ainda na fase de desenvolvimento, caso houvesse uma integração mais efetiva de práticas de segurança desde o início do processo.

Portanto, a ausência de ferramentas automáticas de análise de vulnerabilidades na esteira de desenvolvimento no CCA-SJ gera um processo mais demorado e propenso a retrabalhos, além de potencialmente deixar vulnerabilidades importantes passarem despercebidas até fases muito avançadas do ciclo de desenvolvimento.

Diante do exposto, este trabalho propõe a implantação de ferramentas de análise estática e dinâmica de vulnerabilidades de código na esteira de desenvolvimento de *software* do CCA-SJ. Essa implantação busca garantir dois benefícios principais: o primeiro é a otimização do tempo de desenvolvimento, enquanto o segundo corresponde ao aumento da segurança através da adição de uma camada extra de verificação.

2 DESENVOLVIMENTO

A constante evolução tecnológica e o aumento das ameaças cibernéticas tornaram a segurança de *software* um aspecto crucial para instituições que lidam com informações sensíveis, como a FAB. No contexto militar, a proteção de dados e a integridade dos sistemas de informação são essenciais para garantir a continuidade das operações e a defesa nacional (Brasil, 2018).

Alinhada à Concepção Estratégica Força Aérea 100, que prioriza a proteção cibernética e a inovação tecnológica para garantir superioridade de informações, os sistemas desenvolvidos no CCA-SJ exigem uma abordagem de segurança robusta e integrada desde as fases iniciais de concepção e desenvolvimento (Brasil, 2018).

Diante disso, propõe-se que o CCA-SJ integre dois tipos de ferramentas de segurança à sua esteira de desenvolvimento: ferramentas de análise estática (SAST) e de análise dinâmica (DAST).

As ferramentas SAST, que analisam o código sem executá-lo, são fáceis de usar e ajudam a detectar vulnerabilidades no código-fonte, como validação inadequada e permissões excessivas (Hsu, 2018). Cowel, Lotz e Timberlake (2023) destacam que é possível utilizar mais de uma ferramenta SAST por projeto, pois elas se complementam. Por outro lado, as ferramentas DAST, como OWASP ZAP e *Burp Suite*, simulam ataques reais para identificar falhas de segurança durante a execução do *software* (Hsu, 2018).

2.1 OTIMIZAÇÃO DE TEMPO

O Plano de Tecnologia da Informação da Aeronáutica define o CCA-BR como responsável pela segurança da informação, devendo atuar ativamente nos *softwares* do COMAER (Brasil, 2020). Um exemplo disso é observado no ofício que estabelece as diretrizes para o Projeto FABDoc (também conhecido como SIGADAER 7) e confere ao CCA-BR a responsabilidade pelos testes de vulnerabilidades do projeto (Brasil, 2021).

Dessa forma, a FAB aborda a segurança de forma independente do processo de desenvolvimento de *software*, o que exige uma interação constante entre o CCA-SJ e o CCA-BR. Essa interação ocorre da seguinte maneira: o Centro de São José dos Campos formaliza uma solicitação para análise de segurança de *software*, e o Centro de Brasília realiza os testes e emite um relatório detalhado das vulnerabilidades encontradas. Em seguida, o CCA-SJ analisa o relatório e implementa as correções necessárias, um processo que pode demandar tempo significativo, especialmente quando as vulnerabilidades estão relacionadas à arquitetura do sistema. Após as correções, é necessária uma nova rodada de verificações, que se repete enquanto houver vulnerabilidades pendentes ou novas vulnerabilidades forem encontradas.

Para Abiona *et al.* (2024), a abordagem de tratar segurança de maneira separada do desenvolvimento do *software* pode resultar num consumo de tempo desnecessário. Jha *et al.* (2023) afirmam que testes manuais (incluindo testes de segurança) são mais demorados do que testes automatizados e, com isso, os autores afirmam que testes mais rápidos fazem com que o *software* possa ser disponibilizado para produção mais rapidamente.

O processo para análise de segurança do sistema FABSign exemplifica o tempo exigido para testes e correções. Em 18 de agosto de 2023, o CCA-SJ expediu o ofício n° 506/SAI/2025, solicitando uma Análise de Vulnerabilidades (Brasil, 2023a). O relatório do CCA-BR, emitido em 18 de dezembro de 2023 (Brasil, 2023b), detalhou as vulnerabilidades identificadas, e a correção delas foi concluída pelo CCA-SJ em 15 de fevereiro de 2024, com o envio do ofício

n° 74/SDDM/296 (Brasil, 2024) à Diretoria de Tecnologia da Informação da Aeronáutica. Esse processo, do pedido inicial à finalização da primeira rodada de testes, durou cerca de sete meses.

Vale destacar que, com a implantação de mecanismos para identificar vulnerabilidades nas etapas iniciais do desenvolvimento, o número de testes adicionais com o CCA-BR poderia ser significativamente reduzido, diminuindo assim o tempo total de entrega do *software*.

Um dos problemas descritos no relatório de vulnerabilidades do FABSign foi a exposição do certificado de usuário por meio de uma URL gerada pelo sistema. A correção excedeu o tempo planejado, pois exigiu uma reformulação no método de geração e armazenamento dos certificados, o que impactou algumas funcionalidades interdependentes. Caso uma ferramenta DAST fosse utilizada desde as etapas iniciais do desenvolvimento, essa vulnerabilidade seria detectada e corrigida mais rapidamente, evitando retrabalho e minimizando o impacto na integração das demais funcionalidades.

Portanto, para se obter otimização do tempo de desenvolvimento deve-se implantar ferramentas de análise estática e dinâmica de vulnerabilidades de código na esteira de desenvolvimento de *software* do CCA-SJ.

2.2 AUMENTO DE SEGURANÇA

A análise de segurança realizada pelo CCA-BR simula cenários reais para identificar vulnerabilidades que um invasor pode explorar, utilizando testes conhecidos como *black-box*. Segundo Hsu (2018), esse tipo de teste é adequado para situações no qual o objetivo é simular a perspectiva do invasor.

Myers (2011) define os testes do tipo *black-box* como aqueles que se concentram em verificar se o comportamento do *software* está de acordo com as especificações, sem levar em conta a estrutura interna do programa. Nesse tipo de teste, realiza-se a análise exclusivamente a partir das entradas e saídas do sistema. Para alcançar uma cobertura completa, seria necessário testar exaustivamente todas as entradas possíveis. No entanto, conforme a complexidade do *software* aumenta, o número de combinações de entradas cresce exponencialmente, tornando impraticável a verificação de todos os casos. Assim, a utilização isolada de testes *black-box* não é suficiente para identificar todas as vulnerabilidades presentes no sistema.

Um exemplo ilustrativo de teste *black-box* é a verificação dos campos de pesquisa de documentos no SIGADAER. Em um teste *black-box*, o analista de segurança tentaria explorar comportamentos inesperados, observando a reação do sistema. Ele poderia testar a entrada de caracteres especiais, como aspas ou símbolos, e até tentar injetar código malicioso para verificar

se o sistema é vulnerável a ataques. Esse tipo de abordagem busca identificar falhas ao manipular as entradas sem conhecimento prévio da lógica interna do *software*.

Por outro lado, os testes do tipo *white-box* abordam a segurança a partir de uma perspectiva interna, analisando a lógica e a estrutura do código-fonte. Esse método, que também é discutido por Myers (2011), permite a inspeção minuciosa de pontos críticos, como operadores condicionais e fluxos de decisão. Contudo, assim como no *black-box*, a complexidade do código pode tornar impraticável a cobertura total, dado o crescimento exponencial de caminhos possíveis dentro do programa.

Utilizando o mesmo exemplo de pesquisa de documentos no SIGADAER, um teste do tipo *white-box* poderia ser realizado. Nesse caso, ao analisar a parte do código referente ao exemplo proposto, uma ferramenta SAST analisaria se existem mecanismos de sanitização para caracteres inseridos nos campos antes de serem processados. A sanitização envolve a remoção ou modificação de caracteres especiais, como aspas, barras invertidas ou símbolos de controle, que podem ser usados para injetar comandos maliciosos. Esse tipo de prática poderia evitar problemas de comprometimento da integridade e da confidencialidade dos dados no sistema.

Dessa forma, para alcançar uma proteção mais abrangente, é necessário combinar diferentes abordagens de testes de segurança. Li (2020) explica que ferramentas SAST utilizam a técnica *white-box*, enquanto ferramentas DAST aplicam a técnica *black-box*.

Ao incorporar ferramentas SAST e DAST diretamente na esteira de desenvolvimento, o CCA-SJ garantiria uma camada adicional de proteção. A utilização de ferramentas SAST durante a fase de Integração Contínua (CI) ajudaria a identificar vulnerabilidades no código-fonte no início do processo, enquanto a implementação de DAST durante a fase de Entrega Contínua (CD) permitiria simular ataques em um ambiente mais próximo ao de produção, reforçando a integridade do sistema antes da liberação final.

Portanto, o aumento da segurança através da adição de uma camada extra de verificação pode ser obtido através da implantação de ferramentas de análise estática e dinâmica de vulnerabilidades de código na esteira de desenvolvimento de *software* do CCA-SJ.

3 CONCLUSÃO

A segurança cibernética no contexto militar, especialmente no desenvolvimento de *software* para sistemas críticos como os utilizados pela FAB, é de extrema importância estratégica. A proteção dos dados e a integridade dos sistemas de informação são elementos

essenciais para a defesa nacional, garantindo que as operações militares possam ser conduzidas com eficácia e segurança.

Nesse cenário, o CCA-SJ tem um papel crucial no desenvolvimento de sistemas que suportam a infraestrutura de Comando e Controle (C2) da FAB, além de outros sistemas corporativos utilizados por centenas de Organizações Militares. No entanto, o processo de desenvolvimento de *software* atualmente utilizado no CCA-SJ ainda apresenta fragilidades, principalmente devido à ausência de ferramentas automáticas para a identificação precoce de vulnerabilidades de segurança no código. Além disso, a estratégia atual de delegar a análise de vulnerabilidades exclusivamente ao CCA-BR não só aumenta o tempo total do ciclo de vida do desenvolvimento do *software*, mas também deixa o sistema suscetível a erros que poderiam ser identificados e corrigidos precocemente.

Em resposta a esses desafios, este trabalho defendeu a implantação de ferramentas de análise estática e dinâmica de vulnerabilidades de código na esteira de desenvolvimento de *software* do CCA-SJ proporcionando os seguintes benefícios: otimização do tempo de desenvolvimento e aumento da segurança através da adição de uma camada extra de verificação.

A otimização do tempo de desenvolvimento advém da capacidade de automatizar a verificação de vulnerabilidades nas fases inicias do desenvolvimento do *software*, evitando a propagação de falhas para as etapas finais, o que causaria menos retrabalho e menos rodadas de teste de segurança junto ao CCA-BR. Por outro lado, o aumento da segurança é alcançado ao combinar técnicas de análise estática e dinâmica, ampliando a cobertura dos testes de segurança.

A adoção dessas ferramentas permitirá que o CCA-SJ entregue sistemas mais seguros e eficientes, que atendam aos elevados padrões de qualidade e segurança exigidos. Além disso, ao aprimorar seus processos de desenvolvimento, o CCA-SJ contribuirá diretamente para o fortalecimento estratégico da FAB, assegurando a superioridade de informações. Isso é essencial para aumentar a efetividade das operações militares e reforçar a proteção cibernética dos sistemas, consequentemente garantindo a defesa da soberania nacional.

REFERÊNCIAS

ABIONA, Oluwatosin Oluwatimileyin *et al*. The emergence and importance of DevSecOps: Integrating and reviewing security practices within the DevOps pipeline. **World Journal of Advanced Engineering Technology and Sciences**, [s. l.], v. 11, n. 2, p. 127-133, 2024. Disponível em: https://doi.org/10.30574/wjaets.2024.11.2.0093. Acesso em: 27 set. 2024.

AMARO, Ricardo; PEREIRA, Ruben; DA SILVA, Miguel Mira. Capabilities and practices in DevOps: a multivocal literature review. **IEEE Transactions on Software Engineering**, [s. l.], v. 49, n. 2, p. 883-901, 2022. Disponível em: https://doi.org/10.1109/tse.2022.3166626. Acesso em 28 set. 2024.

BRASIL. Ministério da Defesa. Comando da Aeronáutica. **Ofício 35/SDPC/2560**. Brasília: Centro de Computação da Aeronáutica de Brasília, 18 dez. 2023b. Assunto: Teste de Segurança em Ferramenta para Assinatura Avançada. Disponível em: https://sigadaer.sti.intraer/documento/M2QP331. Acesso em: 27 out. 2024.

BRASIL. Ministério da Defesa. Comando da Aeronáutica. **Ofício 74/SDDM/296**. São José dos Campos: Centro de Computação da Aeronáutica de São José dos Campos, 15 fev. 2024. Assunto: Teste de Segurança em Ferramenta para Assinatura Avançada. Disponível em: https://sigadaer.sti.intraer/documento/X5DLPRU. Acesso em: 27 out. 2024.

BRASIL. Ministério da Defesa. Comando da Aeronáutica. **Ofício 164/TIIS/1866**. São Paulo: Diretoria de Tecnologia da Informação da Aeronáutica, 24 ago. 2021. Assunto: Projeto FABDoc. Disponível em: https://sigadaer.sti.intraer/documento/CCIC618. Acesso em: 27 out. 2024.

BRASIL. Ministério da Defesa. Comando da Aeronáutica. **Ofício 506/SAI/2025**. São José dos Campos: Centro de Computação da Aeronáutica de São José dos Campos, 18 ago. 2023a. Assunto: Análise de Vulnerabilidade de Ferramenta para Assinatura Avançada. Disponível em: https://sigadaer.sti.intraer/documento/ZUM9GNJ. Acesso em: 27 out. 2024.

BRASIL. Ministério da Defesa. Comando da Aeronáutica. Portaria n° 1.597/GC3, de 10 de outubro de 2018. Aprova a reedição da DCA 11-45 "Concepção Estratégica - Força Aérea 100". **Boletim do Comando da Aeronáutica**, Rio de Janeiro, n. 180, 15 out. 2018. Disponível em: http://www.sislaer.fab.mil.br/terminalcendoc/acervo/detalhe/3937. Acesso em: 08 out. 2024.

BRASIL. Ministério da Defesa. Comando da Aeronáutica. Portaria n° 3/CONTI, de 17 de dezembro de 2020. Aprova a reedição do Plano de Tecnologia da Informação da Aeronáutica. **Boletim do Comando da Aeronáutica**, Rio de Janeiro, n. 233, p. 484-559, 22 dez. 2020. Disponível em: http://www.sislaer.fab.mil.br/terminalcendoc/acervo/detalhe/12987. Acesso em: 08 out. 2024.

COWELL, Christopher; LOTZ, Nicholas; TIMBERLAKE, Chris. **Automating DevOps with GitLab CI/CD Pipelines**: Build efficient CI/CD pipelines to verify, secure, and deploy your code using real-life examples. Birmingham: Packt Publishing Ltd, 2023.

HSU, Tony Hsiang-Chih. **Hands-On Security in DevOps:** Ensure continuous security, deployment, and delivery with DevSecOps. Birmingham: Packt Publishing Ltd, 2018.

JHA, Amitkumar V. *et al*. From theory to practice: Understanding DevOps culture and mindset. **Cogent Engineering**, [s. l.], v. 10, n. 1, p. 2251758, 2023. Disponível em: https://doi.org/10.1080/23311916.2023.2251758. Acesso em 28 set. 2024.

LI, Jinfeng. Vulnerabilities mapping based on OWASP-SANS: a survey for static application security testing (SAST). **Annals of Emerging Technologies in Computing**, [s. l.], pp. 1-8,

Vol. 4, No. 3, 1st July 2020. Disponível em: https://doi.org/10.33166/aetic.2020.03.001. Acesso em 28 set. 2024.

MYERS, Glenford J.; SANDLER, Corey; BADGETT, Tom. **The art of software testing**. Hoboken: John Wiley & Sons, 2011.

ZAYDI, Mounia; NASSEREDDINE, Bouchaib. DevSecOps practices for an agile and secure it service management. **Journal of Management Information and Decision Sciences**, v. 23, n. 2, p. 1-16, 2020.