Investigação das práticas proativas de desenvolvimento de software seguro adotadas por pequenos times de desenvolvimento

João Paulo de S. Rodrigues[1], Kleverton Robson Da Silva Cordovil[2], Orlando Shigueo Ohashi Junior[3], Renato Hidaka Torres[1]

[1]Faculdade de Computação – Universidade Federal do Pará (UFPA)
CEP – 66.075-110– Belém – PA – Brazil

[2]Instituto Federal do Pará – Campus Belém (IFPA)

[3] Instituto Cyberespacial – Universidade Federal Rural da Amazônia (UFRA)

joao.rodrigues@icen.ufpa.br, kleverton.cordovil@ifpa.edu.br,
orlando.ohashi@ufra.edu.br, renatohidaka@ufpa.br

Abstract. *With the growth of the dangers of the Internet, trusted computing has introduced a new development model in the software industry, imposing the adoption of secure practices throughout the software development cycle. Based on the study of Microsoft, Safecode and Touchpoints secure development cycles, this work developed a survey with the objective of investigating the secure software development practices adopted by small development teams. The observed results were to correlate them with the security guidelines of the three cycles studied and point out that important security practices such as penetration testing and static and dynamic code analysis are not often used by small teams.*

Resumo. *Com o crescimento dos perigos da internet, a computação confiável introduziu na indústria de software uma nova filosofia de desenvolvimento, impondo a adoção de práticas seguras durante todo o ciclo de desenvolvimento. A partir do estudo dos ciclos seguros de desenvolvimento da Microsoft, Safecode e Touchpoints, esse trabalho desenvolveu um survey com o objetivo de investigar as práticas de desenvolvimento de software seguro adotadas por pequenos times de desenvolvimento. Os resultados observados foram correlacionados com as diretrizes de segurança dos três ciclos estudados e apontaram que práticas de segurança importantes como teste de penetração e análise estática e dinâmica de código não são frequentemente utilizadas por times pequenos.*

1. Introdução

Em 2018, em um estudo realizado com 477 companhias, o Poneman Institute (2018) reportou que uma vulnerabilidade de *software* custou em média U$7.9 milhões de dólares para as empresas americanas e U$1.2 milhões de dólares para as empresas brasileiras. Nesse mesmo estudo, o Poneman Institute também reportou que o tempo médio para identificar e corrigir uma vulnerabilidade foi, respectivamente, de 197 e 69 dias.

Segundo o relatório da IBM security (2020), no Brasil, 25% das causas de vazamento de dados foram causadas por erros humano, 28% por erros no sistema e as demais causas por ataques mal intencionados. O elevado custo e tempo demandado para

corrigir as vulnerabilidades do projeto, apontados no parágrafo anterior, podem ser justificados pela metodologia de desenvolvimento adotadas pelas empresas de software.

Historicamente, a indústria de *software* focou suas estratégias de segurança no uso da abordagem chamada de penetrar e corrigir (L. Williams, 2019). Nessa abordagem, o teste de penetração avalia a segurança do sistema somente quando o desenvolvimento é concluído. Ataques conhecidos são simulados para verificar o nível de segurança do *software* implementado. Havendo falhas de segurança, os desenvolvedores reagem localizando e corrigindo a vulnerabilidade através de um *patch* de segurança. Apesar de sua eficácia, os custos acumulados pela demora na detecção e manutenção das falhas de segurança tende a crescer com o passar do tempo.

Diante dos problemas da abordagem de penetrar e corrigir, pesquisadores buscaram por medidas proativas para proporcionar o desenvolvimento de *softwares* seguros. Em 1998, McGraw (1998) iniciou esse movimento demonstrando que a análise proativa e rigorosa do *software* deve desempenhar um papel cada vez mais importante na avaliação e prevenção de vulnerabilidades em aplicativos. Em suas pesquisas, McGraw observou que a origem dos problemas de violação de segurança ocorre devido a erros de *design* e de codificação do *software*.

A partir do estudo conduzido por McGraw, surgiram as medidas proativas denominadas processos seguros do ciclo de vida de *software*. Estes processos trabalham profundamente a segurança do sistema em todas as etapas de desenvolvimento do produto. Na literatura, existem várias propostas de processos seguros do ciclo de vida de *software*. A título de exemplo, podemos citar o Microsoft *Security Development Lifecycle* (M. Howard, 2006), *Software Security Touchpoints* (G. McGraw, 2006) e a proposta da SAFECode (2018).

Na perspectiva de uma segurança proativa, considerando as diferentes metodologias de desenvolvimento de *software* seguro, é possível inferir que as diretrizes de um processo seguro do ciclo de vida de *software* devem ser incorporadas à metodologia de desenvolvimento adotada pelas equipes. Assim, esse trabalho visa realizar um *survey* para investigar as metodologias de desenvolvimento adotadas pelas equipes e quais as práticas proativas de segurança de *software* que são executadas por times de desenvolvimentos considerados pequenos (de 1 até 5 pessoas).

Extraídas estas práticas por meio da investigação, será feita a correlação com as recomendações de processos seguros do ciclo de vida de *software* e quantificada a maturidade do time no que diz respeito ao desenvolvimento proativo de *software* seguro. A partir da quantificação da maturidade das equipes, expõe-se como contribuição, recomendações de práticas proativas de desenvolvimento de *software* seguro que possam ser adotadas por pequenos times de desenvolvimento.

A escolha por times de até 5 pessoas foi fundamentada pela hipótese de que pequenas equipes de desenvolvimento não adotam práticas proativas para proporcionar o desenvolvimento de *software* seguro. Desse modo, após análise e correlação dos dados coletados, a hipótese será refutada se for constado que os participantes da pesquisa adotam pelo menos 70% das diretrizes de um processo seguro do ciclo de vida de *software*, porcentagem considerada segura pelos pesquisadores. Para fins de correlação, serão considerados as propostas de processos seguros do ciclo de vida de *software* de Microsoft, Touchpoints e SAFECode.

2. Ciclos seguros de desenvolvimento

O desenvolvimento de *software* é um processo que envolve pessoas, ferramentas de desenvolvimento, estratégia de organização e tomadas de decisões. Como resultado, milhões de linhas de código e de documentação são produzidas. Por se tratar de um processo complexo, é natural que o sistema desenvolvido apresente a curto ou a longo prazo, falhas de segurança. Para mitigar as falhas de segurança e manter a qualidade do *software*, grandes empresas estão incorporando as práticas do ciclo de desenvolvimento de *software* seguro. Independentemente do ciclo de desenvolvimento adotado, tais ações procuram inserir premissas de segurança em todas as fases do projeto.

O ciclo de desenvolvimento de software seguro da Microsoft foi desenvolvido com intuito de abranger todas as fases do projeto de software. O ciclo conta com recomendações de testes, verificações, gerenciamento de riscos de terceiros, treinamento das equipes de desenvolvimento, modelo de ameaça entre outras práticas de desenvolvimento. Após a implementação de um processo de desenvolvimento seguro pela Microsoft na indústria, demais processos foram sendo desenvolvidos aplicando variações do modelo da Microsoft.

O ciclo seguro da *Touchpoints*, desenvolvida por Gary McGraw, se destaca entre os modelos de desenvolvimento por se restringir apenas no *design* e codificação do projeto. A *Touchpoints* faz recomendação da revisão manual para equipes mais experientes sendo recomendado o uso de ferramentas automatizadas como substituto para equipes com menor experiência. Todos os testes de verificação e implementação recomendados pela Microsoft estão presentes no ciclo da *Touchpoints.*

O ciclo seguro da SAFECode reúne as recomendações dos ciclos da Microsoft e *Touchpoints* e adiciona como contribuição recomendação de padronização de codificação na equipe de desenvolvimento. Com o passar do tempo, sucessivas revisões e implementações de novas funções resultam em uma perca de produtividade nos times de desenvolvimento como apresentado em Stripe (2018).

3. Construção e aplicação do *survey*

A partir da revisão dos ciclos de desenvolvimento da Microsoft, *Touchpoints,* e SAFECode, se construiu as perguntas do *survey*. No total, 32 perguntas foram elaboradas com o objetivo de mapear as práticas de desenvolvimento de *software* utilizadas por times de desenvolvimento de 1 até 5 pessoas. A Tabela 1 apresenta o questionário produzido, bem como a correlação das perguntas com as diretrizes dos processos seguros estudados.

Cada pergunta foi distribuída entre as diretrizes de governança, *design*, implementação, verificação e operação. Estas diretrizes foram baseadas nas definições de funções de negócios do modelo de maturidade da OWASP SAMM e nos ciclos de desenvolvimento estudados. As cinco primeiras perguntas são questões de resposta aberta cujo objetivo é mapear o perfil do entrevistado, conforme ilustrado na Figura 1.

Com intuito de esclarecer as perguntas criou-se um documento de apoio (encurtador.com.br/bmHW3) que detalhava como cada prática de segurança deve ser realizada, segundo as recomendações da Microsoft (M. Howard, 2006), *Touchpoints* (G. McGraw, 2006) e SAFECode (2018). Desse modo, o entrevistado deveria responder qual a frequência de utilização de cada prática de segurança, em relação aos projetos que ele participa ou já participou.

Tabela 1. Correlação entre as perguntas e diretrizes de desenvolvimento seguida de seus respectivos pesos

Diretriz	Perguntas	Peso
	1. Quantos anos de experiência de mercado você tem?	-
	2. Qual ciclo de desenvolvimento de software sua equipe adota?	-
	3. Qual o tamanho da equipe de desenvolvimento?	-
	4. Quais são as principais linguagens de programação utilizadas no desenvolvimento de aplicações?	-
	5. Quais treinamentos de segurança de software a equipe participou nos últimos três anos?	-
GOVERNANÇA	6. A equipe possui um conjunto padrão de requisitos de segurança e procedimentos de verificação que abordam as obrigações de conformidade externa da organização?	2
	7. A organização utiliza métricas para medir o desempenho da equipe?	1
	8. A organização utiliza métricas para medir a segurança da aplicação?	2
	9. A organização revisa e atualiza regularmente as métricas de risco da aplicação?	3
	10. A organização utiliza métricas para avaliar os riscos de negócio?	1
DESIGN	11. A equipe possui um modelo de ameaças?	2
	12. A equipe de desenvolvimento especifica os requisitos de segurança durante o desenvolvimento?	1
	13. Os fornecedores (SOs, servidores, bibliotecas, componentes de terceiros etc.) atendem às responsabilidades de segurança e medidas de qualidade estabelecidos pela organização?	2
	14. A equipe possui uma lista de tecnologias recomendadas para o desenvolvimento de soluções?	2
	15. A organização impõe o uso de tecnologias recomendadas dentro das equipes de desenvolvimento?	3
	16. A organização impõe o uso de criptografia para os dados inseridos e armazenados na Aplicação?	1
	17. A organização possui um inventário das versões dos componentes e de suas dependências?	1
	18. A equipe lida com os riscos da dependência de terceiros por meio de um processo formal?	2
IMPLEMENTAÇÃ	19. A equipe de desenvolvimento possui um padrão interno de codificação, como regras de indentação, comentário, nomenclatura para as variáveis e funções etc.?	2
	20. A equipe utiliza ferramentas automatizadas de análise de código, como testes dinâmico (DAST) e estático (SAST)?	2
	21. A equipe faz revisão manual do código?	3
	22. A organização avalia regularmente as ameaças à arquitetura?	2
	23. A organização revisa regularmente os mecanismos de segurança da arquitetura?	2

O		
VERIFICAÇÃO	24. A equipe atualiza regularmente a arquitetura de risco com base nas descobertas da avaliação de segurança?	2
	25. A equipe realiza testes baseado em riscos da aplicação?	2
	26. A equipe aplica testes de penetração?	2
	27. A organização possui um processo formal para detecção de incidentes?	2
	28. A equipe analisa os dados de log para incidentes de segurança periodicamente?	1
OPERAÇÃO	29. A equipe responde aos incidentes detectados?	1
	30. A organização identifica e remove sistemas, aplicativos, dependências de aplicativos ou serviços que não são mais usados, atingiram o fim da vida útil ou não são mais ativamente desenvolvidos ou suportados?	1
	31. A organização monitora seus componentes e bibliotecas que não sofrem manutenção ou cujas versões antigas não são alvos de atualização?	2
	32. A organização faz buscas por vulnerabilidades e de seus componentes dentro de fontes oficiais?	1

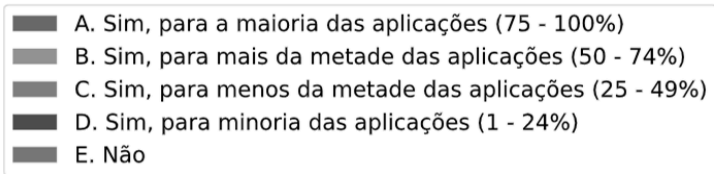


Figura 1. Alternativas das perguntas 6-32 correspondente ao nível de adoção de práticas seguras de desenvolvimento

A aplicação do questionário ocorreu durante os meses de novembro/2020 a fevereiro/2021. O questionário foi publicado na plataforma do Google Forms e divulgado nas redes sociais LinkedIn, Facebook e nos aplicativos de mensagens WhatsApp, Telegram e Discord. No total, foram coletadas 95 respostas, sendo 45 de desenvolvedores que participam de pequenos times de desenvolvimento.

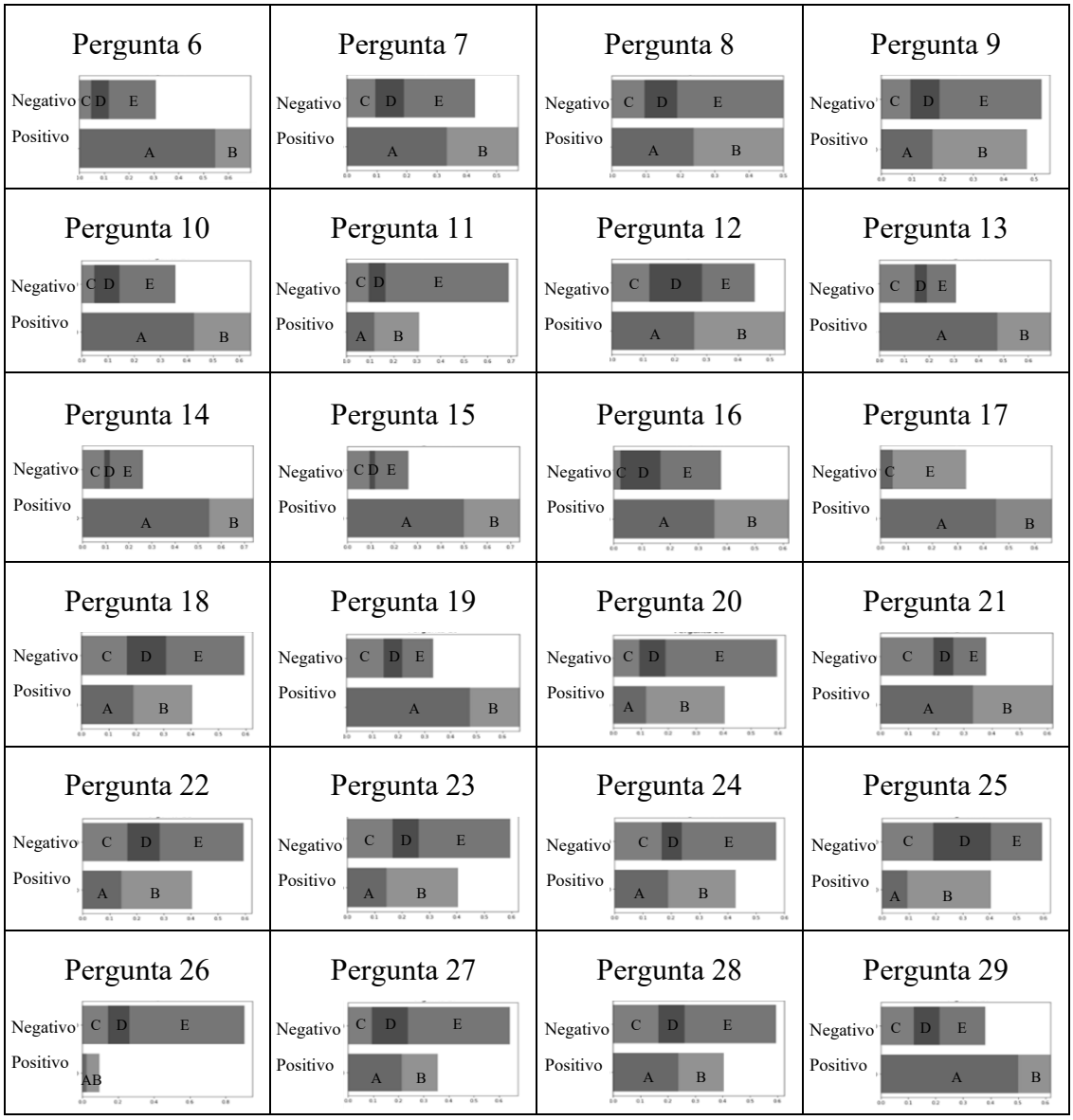
4. Análise das equipes entrevistadas

Conforme observado na Tabela 1, para cada pergunta, se atribuiu um peso que representa a complexidade da realização da prática. Os pesos 1, 2 e 3 foram distribuídos para determinar respectivamente as tarefas simples, moderadas e difíceis seguindo as recomendações do modelo de maturidade da OWASP SAMM (2020). Para computar a pontuação total de cada entrevistado, se utilizou a seguinte regra: para cada pergunta, se o entrevistado responder à alternativa A ou B da Figura 1, soma o peso, senão, soma zero.

Na Figura 2, é possível perceber a frequência das respostas de todos os entrevistados para cada pergunta. As repostas foram agrupadas em barras positivas e negativas. A barra positiva é formada pela frequência das respostas das alternativas A e B. A barra negativa corresponde às frequências das repostas C, D e E da figura 1. Para computar a pontuação geral dos entrevistados, se analisou o gráfico de cada pergunta. Se a barra positiva for

maior ou igual a barra negativa, então é somado o peso da pergunta à pontuação geral, senão, somamos zero.

Considerando o peso de cada pergunta definido na Tabela 1, a pontuação geral máxima que poderia ser obtida corresponde a 48 pontos. Para se obter essa pontuação é necessário que todas as barras positivas da Figura 2 sejam maiores ou iguais às barras negativas. Em termos práticos, esse comportamento indicaria que as equipes de desenvolvimento pequenas adotam todas as práticas de segurança em mais da metade dos projetos de desenvolvimento que participam. Entretanto, esse comportamento não foi observado em 12 das 27 práticas recomendadas pelos ciclos de desenvolvimento estudados. No gráfico 8, apesar das barras apontarem igualdade entre os dados, o resultado foi negativo com 1,1% de diferença. Tal comportamento implicou em uma pontuação total igual a 24 de 48 pontos possíveis. Na Figura 2, os dados coletados estão dispostos na ordem crescente das perguntas, no sentindo da direita para esquerda, de cima para baixo, para visualização dos resultados.



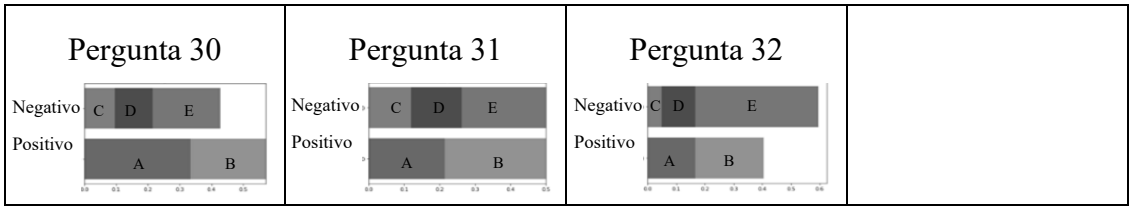


Figura 2. Análise dos dados coletados das questões 6-32.

O perfil de experiência dos pequenos se distribuiu com 33,3% dos entrevistados com 0-2 anos de experiência, 20% com experiência de 2-5 anos, com 17,7% de 5-10 anos e 29% com experiência superior a 10 anos. Os intervalos de experiência contemplaram os níveis de cargo *trainee*, júnior, pleno e sênior. Comparando o nível de adoção com tempo de experiência das equipes, se constatou que equipes com mais de 10 anos de experiência e equipe com 0-2 anos de experiência foram as que menos adotaram práticas de segurança, com média de 12,38 e 13,46 de adoção das práticas recomendadas. As demais equipes ficam com médias de 15,55 e 14,37 nas equipes de 2-5 anos e 5-10 anos de experiência na devida ordem.

O baixo nível de adoção refletiu em parte a carência do ensino e orientação de práticas seguras de desenvolvimento de *software* desde a academia e a falta de investimentos em educação das equipes empresas em seus funcionários. Dos 45 entrevistados apenas 17 haviam participado de algum treinamento de segurança nos últimos 3 anos.

Os dados apontaram uma negligência dos pequenos times de desenvolvimento nas práticas de uso e revisão de métricas de segurança. Igualmente, ocorre negligência nas práticas envolvendo a arquitetura do sistema, como revisão regular de ameaças, mecanismo de segurança, e atualizações com bases em descobertas de segurança. Ambos processos são primordiais para o fornecimento de dados de segurança do projeto, sendo recomendado o uso de tais práticas.

Apesar de um alto nível de adoção de criptografia pelos pequenos times de desenvolvimento a falha no uso de algoritmos de criptografia em dados inseridos e armazenados na aplicação é um dos principais fatores responsáveis pela exposição de dados sensíveis, segundo a OWASP (2017). Sendo necessário um reforço nas recomendações e orientações do uso de criptografia.

5. Recomendações para os times pequenos

No *design* do projeto, o modelo de ameaças é um item de extrema importância para o andamento do projeto, analisando de maneira preventiva possíveis falhas nos mecanismos de defesa e ataques ao sistema. No STRIDE, modelo de ameaças desenvolvido e recomendado pela Microsoft (2017), cada ameaça é uma violação de uma propriedade para um sistema, como autenticidade, integridade, confidencialidade e disponibilidade. Apesar de sua relevância a prática é adotada em pelo menos 50% das aplicações por 13 equipes, possuindo adesão inferior a 50% dos entrevistados.

A pesquisa também apontou a adoção de um padrão interno de codificação e à revisão manual do código, as demais práticas recomendas na diretriz de implementação foram depreciadas. O uso de ferramentas automatizadas de análise de código tem se mostrado crescente no mercado de *software*, novas tecnologias também vêm sido

implementadas para a fase de validação e verificação do produto como o uso de *bots* e inteligências artificiais. Para as equipes com o modelo ágil, que correspondem por 24 equipes entrevistadas, o uso destas ferramentas permite a agilidade e segurança na entrega do produto final.

Durante a fase de verificação, os pequenos times obtiveram os piores índices de adoção, sendo a aplicação de *pentest*, teste de penetração, a mais rejeitada, 64,3% de adoção em nenhuma das aplicações. Segundo dados publicados pela ISC^2 (2020), a escassez de profissionais de segurança continua sendo a principal ameaça para as organizações, segundo este mesmo artigo a escassez mundial de profissionais de segurança atingiu o patamar de 3,1 Milhões, sendo no Brasil uma carência de 331 mil. Apesar de ser teste indispensável, a esquipes de desenvolvimento podem optar por um desenvolvimento preventivo de segurança, focando esforços principalmente nas práticas de uso do modelo de ameaças, testes baseados em riscos, uso das ferramentas automatizadas e revisões manuais de código para equipes mais experientes. O teste baseado em riscos é aplicado durante o desenvolvimento do projeto, não sendo necessária a finalização do produto, a adoção da prática ajuda na otimização de testes e na priorização de revisões de possíveis vulnerabilidades.

O log de dados registra os eventos relevantes no sistema. Embora todas as práticas recomendas tenham como fim tornar o sistema mais seguro, nenhum sistema é 100% seguro, logo o log de dados contribui para restauração do estado original e compreendimento do comportamento do sistema durante algum evento, ajudando na revisão dos mecanismos de segurança e da arquitetura de risco.

Os ciclos de desenvolvimento de *software* seguro da Microsoft, *Touchpoints* e SAFECode, estão abertos a experimentação por parte de cada equipe entrevista para aprimoramento das práticas de segurança da organização.

6. Conclusão

Este trabalho se propôs a investigar as práticas de segurança de *software* adotadas por pequenos times de desenvolvimento. Para isso, foi construído um *survey,* considerando as diretrizes de segurança dos processos seguros do ciclo de vida de *software* da Microsoft, SAFECode e *Touchpoints*. No total, foram elaboradas 32 perguntas para mapear o perfil e práticas de segurança adotadas pelos entrevistados. Para analisar a maturidade das equipes, utilizou-se o modelo de maturidade de *software* da OWASP SAMM (2020) como base. Dada a aplicação do questionário, foram coletadas 95 respostas, sendo 45 de desenvolvedores que participam de pequenos times de desenvolvimento.

Ao realizar a sumarização dos resultados e correlacionar com as recomendações de segurança dos processos da Microsoft, SAFECode e *Touchpoints*, observou-se a carência de maturidade dos entrevistados no desenvolvimento de *software* seguro. Das 5 diretrizes analisadas, a maior carência dos times pequenos está na adoção de práticas seguras nas fases de implementação e verificação do ciclo de desenvolvimento. A análise revelou que os pequenos times de desenvolvimento ignoram testes de segurança de *software* importantes como os testes de penetração e análise estática e dinâmica de código.

Como já apresentado, a falta de adoção de práticas seguras observada nos times de desenvolvimento refletiu em parte a carência do ensino e orientação de práticas seguras

de desenvolvimento de *software* desde a academia. Assim, podemos concluir que a filosofia do desenvolvimento de *software* seguro precisa ser trabalhada desde a graduação. Nessa linha de pensamento, Elder et al. (2021) apresenta propostas de intervenção para mudança deste cenário no ensino de segurança nos cursos de ensino superior a partir dos padrões OWASP de segurança. Como trabalhos futuros, pretende-se seguir a proposta de Elder et al. (2021) e elaborar roteiros gamificados de teste de penetração e análise estática e dinâmica de código para alunos de graduação.

Referências

Poneman Institute (2018), "2018 cost of a data breach study: Global overview", https://securityintelligence.com/ponemon-cost-of-a-data-breach-2018/.

IBM Security (2020), "Relatório sobre o prejuízo de um vazamento de dados 2020", https://www.ibm.com/security/digital-assets/cost-data-breach-report/#/pt

L. Williams (2019), "Secure Software Lifecycle in CyBOK Version 1.0", The National Cyber Security Centre 2019.

G. McGraw (1998), "Testing for security during development: why we should scrap penetrate-and-patch", IEEE Aerospace and Electronic Systems Magazine, vol. 13, no. 4, pp. 13–15.

M. Howard (2006) and S. Lipner, The Security Development Lifecycle. Redmond, WA, USA: Microsoft Press.

G. McGraw (2006), Software Security: Building Security In. Addison-Wesley Professional.

SAFECode (2018), "Fundamental practices for secure software development: Essential elements of a secure development lifecycle program", 3th Edition, https://safecode.org/wp-content/uploads/2018/03/SAFECode_Fundamental_Practices_for_Secure_Software_Development_March_2018.pdf.

Stripe (2018), "Old and Bad Code Waste Billions",

https://www.i-programmer.info/news/99-professional/12118-old-and-bad-code-waste-billions.html.

OWASP (2020), "SAMM", https://github.com/OWASP/samm/blob/master/Supporting%20Resources/v2.0/OWASP-SAMM-v2.0.pdf.

OWASP (2017), "OWASP top 10 – 2017",

https://owasp.org/www-pdf-archive/OWASP_Top_10-2017_%28en%29.pdf.pdf.

Microsoft (2017), "Microsoft Threat Modeling Tool threats", https://docs.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-threats.

ISC^2 (2020), "Cybersecurity Workforce Study",

https://www.isc2.org/Research.

Sarah Elder, Nusrat Zahan, Val Kozarev, Rui Shu, Tim Menzies, Laurie Williams (2021), "Structuring a Comprehensive Software Security Course Around the OWASP Application Security Verification Standard",

https://arxiv.org/abs/2103.05088.