Uso de técnicas e ferramentas para detecção de vulnerabilidades: um *survey* com membros de equipes de desenvolvimento ágil de software

Lígia Cássia M. de Castro Santos[1], **Marcos Lordello Chaim**[1], **Edmir Parada V. Prado**[1]

[1]Escola de Artes, Ciências e Humanidades – Universidade de São Paulo (EACH-USP)
Caixa Postal 03828-000 – São Paulo – SP – Brazil

ligia.santos01@usp.br, chaim@usp.br, eprado@usp.br

Abstract. *Agile methods were created to address real and perceived weaknesses of traditional software development methods. However, due to the pressure on delivery of a software product within the estimated timeframe, security requirements are poorly measured or even overlooked. During agile software development, it is important to detect possible vulnerabilities during their life cycle, using processes, activities, techniques or tools. This article describes a survey that has been applied to members of software development teams that use agile methods. From the results of this research, we were able to identify and describe the state of practice, difficulties and opportunities to reconcile agile methods and software security.*

Resumo. *Métodos ágeis foram criados para sanar fraquezas reais e perceptíveis dos métodos tradicionais de desenvolvimento de software. Entretanto, pela pressão na entrega de um produto de software dentro do prazo, requisitos de segurança são pouco mensurados ou até deixado de lado. Durante o desenvolvimento ágil de software é importante detectar possíveis vulnerabilidades, utilizando processos, atividades, técnicas ou ferramentas. Este artigo descreve um survey que foi aplicado a membros de equipes de desenvolvimento de software que utilizam métodos ágeis. A partir dos resultados dessa pesquisa, foi possível identificar e descrever o estado da prática, as dificuldades e as oportunidades para conciliar métodos ágeis e segurança de software.*

1. Introdução

Atualmente, as organizações operam em um ambiente com constantes mudanças culturais, sociais, políticas e econômicas. Com isso elas devem responder rapidamente a essas mudanças e pressões competitivas [Sommerville 2011]. Processos de desenvolvimento de software que planejam especificar completamente os requisitos e, em seguida, projetar, construir e testar o sistema de forma sequencial e rígida acabam sendo inadequados para ambientes que sofrem constantes mudanças [Sommerville 2011]. Se os requisitos de software mudam, consequentemente o projeto, implementação e testes do sistema mudam também. Por conta disso, torna-se difícil obter um conjunto completo de requisitos estáveis de software [Pressman 2011]. No entanto, em ambientes com constantes mudanças, uma abordagem ágil é recomendável.

Devido à agilidade do processo de desenvolvimento e pressão na entrega, alguns softwares não têm uma análise completa da segurança para cada fase do ciclo de vida

e com isso podem surgir vulnerabilidades [OWASP 2010b]. De acordo com a OWASP (2010b), a maioria das vulnerabilidades nos sistemas de software são causadas pela falta de aplicação de políticas de segurança no projeto. Para evitar isso, é necessário analisar o uso de práticas de segurança pelos membros de equipes, bem como as motivações, dificuldades, limitações e lições aprendidas na implantação de técnicas e ferramentas de detecção de vulnerabilidades. Howard e Lipner (2006) apontam que práticas de segurança incorporadas no ciclo de vida de desenvolvimento de software diminuem a probabilidade de existirem vulnerabilidades no sistema. Existem processos de desenvolvimento de software seguro voltados para o modelo tradicional de desenvolvimento de software, porém, o seu uso no desenvolvimento ágil de software requer adaptações [Howard and Lipner 2006].

O objetivo deste artigo é identificar e descrever o estado da prática da adoção de técnicas e ferramentas de detecção de vulnerabilidades no contexto de desenvolvimento ágil de software. Pretende-se identificar as técnicas e ferramentas usadas, as motivações, as dificuldades, as limitações e as lições aprendidas. Para tanto, um *survey* foi elaborado com perguntas relacionadas a 18 técnicas e ferramentas de detecção de vulnerabilidades oriundas de três conhecidos processos de desenvolvimento de software seguro, a saber, Processo de McGraw (2004), Processo Leve e Abrangente de Aplicação de Segurança - OWASP CLASP (2010a) e o Processo de Howard e Lipner (2006). Esses processos foram identificados e selecionados por meio de revisão bibliográfica, bem como as técnicas e ferramentas em comum. A partir desses processos e suas atividades, foi desenvolvido um formulário aplicado a 110 membros de equipes que utilizam métodos ágeis. Este artigo apresenta parte dos resultados obtidos.

O artigo está estruturado da forma a seguir. Na Seção 2, são abordados os conceitos básicos sobre métodos ágeis, vulnerabilidades e processos de desenvolvimento seguro de software. A Seção 3 apresenta os trabalhos relacionados. A Seção 4 descreve a proposta do *survey* e a Seção 5 os resultados. Finalmente, as conclusões são apresentadas na Seção 6.

2. Conceitos básicos

Nesta seção, são apresentados os conceitos de métodos ágeis, vulnerabilidades e processos de desenvolvimento de software seguro.

2.1. Métodos ágeis

Agilidade tornou-se a palavra principal quando é descrito um moderno processo de desenvolvimento de software [Sommerville 2011]. Constantes mudanças podem ocorrer durante o desenvolvimento de software como: mudanças no software que está sendo criado, mudanças nos membros da equipe, mudanças devido a novas tecnologias ou mudanças no projeto que cria o produto. Exemplos de métodos ágeis conhecidos são: “Scrum” [Schwaber 2013], “Extreme Programming (XP)” [Beck 2000], “Lean” [Poppendieck and Poppendieck 2003], “Kanban” [Skarin 2015] “Agile Unified Process (AUP)” [Ambler 2006], “Crystal Method” [Cockburn 2004] e “Feature Driven Development (FDD)” [Palmer and Felsing 2001].

No desenvolvimento ágil, um produto de software é desenvolvido por indivíduos trabalhando em equipes e cada membro tem uma habilidade específica, as quais colabo-

ram para o sucesso do projeto [Howard and Lipner 2006]. Esse modelo de desenvolvimento prioriza a satisfação dos clientes e a entrega incremental de produto de software por meio de equipes de projetos pequenas e altamente motivadas, artefatos de engenharia de software mínimos e simplicidade no desenvolvimento [Beck et al. 2001]. Para Pressman (2009), o desenvolvimento ágil oferece benefícios importantes, no entanto, não é indicado para todos os tipos de projetos, softwares e situações. Para utilizá-lo, é essencial que a equipe adapte os processos e atividades existentes e mantenha apenas artefatos realmente essenciais, utilizando processos enxutos, enfatizando a estratégia de entrega incremental.

2.2. Vulnerabilidades

Uma vulnerabilidade em software é um conjunto de condições que podem levar à violação de uma política de segurança [Seacord and Householder 2005]. Tais condições podem ser oriundas da má especificação de requisitos de segurança, problemas com o *design*, práticas de codificação insegura, falhas nas atividades de garantia de segurança ou problemas de manutenção do sistema [Howard and Lipner 2006]. A vulnerabilidade é uma fraqueza na aplicação que permite que um atacante cause danos aos *stakeholders* de um sistema [OWASP 2010a]. Procedimentos de segurança e controles internos ou de implementação que podem ser explorados por ameaças são considerados vulnerabilidades [Nist 2009]. Assim, é de extrema importância detectar vulnerabilidades durante o desenvolvimento de software, pois informações perdidas, utilizadas incorretamente ou acessadas por pessoas não autorizadas podem prejudicar uma organização.

2.3. Processos de desenvolvimento de software seguro

Uma aplicação deve ser segura, estável, livre de riscos e danos, além de ter atividades de segurança incorporadas no seu ciclo de vida [OWASP 2010b]. Um software é considerado seguro se ele não permite o comprometimento da autenticidade, confidencialidade, integridade e disponibilidade dos seus dados. Tradicionalmente, a segurança não é tratada desde as primeiras fases do ciclo de vida de desenvolvimento de software [Khan and Zulkernine 2009].

De acordo com Khan e Zulkernine (2009), a melhor maneira de desenvolver um software seguro é incorporar práticas de segurança desde o seu início. A segurança no software pode ser incorporada durante as fases do ciclo de vida por meio de políticas de segurança, linguagens de especificação de requisitos de segurança, *design* seguro, padrões de codificação segura e métodos de garantia de segurança de software, como testes de penetração, análise estática de código e revisões de segurança de código [Howard and Lipner 2006]. Existem diversos processos que incorporam atividades de segurança no desenvolvimento de software e que podem ser adaptados para a utilização em métodos ágeis, a saber, Processo de McGraw (2004), Processo Leve e Abrangente de Aplicação de Segurança - OWASP CLASP (2010) e o Processo de Howard e Lipner (2006).

3. Trabalhos relacionados

Foram encontrados três trabalhos relacionados com o objeto dessa pesquisa. O primeiro investiga o uso de segurança de software, as competências e as necessidade de treinamento de equipes que utilizam métodos ágeis em duas organizações [Oyetoyan et al. 2016]. O segundo investiga a realização de testes de segurança em quatro equipes ágeis. O estudo compara como a segurança é gerenciada e organizada, bem como a combinação das

técnicas e ferramentas de teste de segurança usadas no ciclo de vida de desenvolvimento ágil de software [Cruzes et al. 2017]. Já no terceiro, foi realizada por [Khaim et al. 2016] uma revisão sistemática da literatura (RSL) que trata da aplicaçao de segurança ao longo do ciclo de vida do desenvolvimento ágil por especialistas em segurança. Este artigo, por sua vez, visa identificar e descrever o estado da prática, as motivações, as dificuldades, as limitações e as lições aprendidas no uso de técnicas e ferramentas para a detecção de vulnerabilidades por membros de equipes brasileiras de desenvolvimento ágil.

4. Proposta de pesquisa

Este artigo apresenta um *survey* que tem por objetivo identificar e descrever as técnicas e ferramentas de detecção de vulnerabilidades em desenvolvimento ágil de software adotadas por membros de equipes que utilizam métodos ágeis. Para atingir o objetivo do estudo, primeiramente, por meio de uma revisão bibliográfica, foram identificadas e descritas 18 técnicas e ferramentas de segurança que são comuns aos seguintes processos: OWASP CLASP, de McGraw e de Howard e Lipner. Elas foram classificadas em função das fases do ciclo de vida de desenvolvimento de software: análise de requisitos e modelagem, *design*, codificação, testes e produção. As técnicas e ferramentas identificadas estão listadas na Tabela 1.

Tabela 1. Técnicas e ferramentas para detecção de vulnerabilidades utilizadas pelos processos de segurança

Análise de requisitos e modelagem	**Codificação**
Especialista em segurança	Codificação segura
Requisitos de segurança	Revisão de segurança de código
Abuse/Misuse Cases/Stories	Ferramenta de revisão de código
Modelagem de ameaças	Ferramenta de análise estática de código
Modelagem de riscos	**Testes**
Ferramenta de modelagem de riscos e ameaças	Ferramenta de análise dinâmica de código
Design	*Fuzz Testing*
Design de segurança	Teste de penetração
Contramedidas de segurança	Testes baseados em riscos
Avaliação de vulnerabilidades	**Produção**
	Gestão de respostas a incidentes

Em seguida, foi elaborado um instrumento cujo objetivo é caracterizar quais técnicas e ferramentas são aplicadas pelos membros das equipes e quais suas motivações, dificuldades, limitações e lições aprendidas ao utilizá-las. O instrumento é composto por um questionário com 15 perguntas, elaborado por meio da ferramenta *Online* Pesquisa[1]. O instrumento completo poderá ser encontrado em: `https://github.com/SAEG1/InstrumentoEstudodeCaso.git`.

Foram selecionados 110 membros de equipes que utilizam métodos ágeis como processo de desenvolvimento de software. Tal seleção ocorreu por meio da busca por profissionais na rede de negócios LinkedIn[2]. Foram considerados membros de equipes nas quais técnicas e ferramentas de detecção de vulnerabilidades estão em processo de implantação, já estão implantadas ou ainda serão implantadas. A seleção de profissionais focou nas seguintes funções: especialista em segurança, analista de requisitos, arquiteto de software, desenvolvedor e testador de equipes adotam métodos ágeis. Porém, foram

[1]https://www.onlinepesquisa.com
[2]https://www.linkedin.com

encontradas outras funções específicas dos métodos ágeis, tais como: *scrum master*, *product owner* e *agile coach*, ou ainda funções tradicionais, como: gestores, líderes de equipes, coordenadores de TI, analistas de qualidade e cientistas de dados as quais foram consideradas na pesquisa.

Foi enviado um convite para todos os candidatos participarem do experimento, contendo uma breve descrição sobre o projeto, as etapas do experimento, sua duração, endereço de acesso para o questionário e termo de consentimento para manipulação do conteúdo das questões abertas. Para realizar inferências a partir dos dados obtidos por meio das questões abertas, foi utilizada a técnica de análise de conteúdo. Análise de conteúdo é um conjunto de técnicas de análise das comunicações que utiliza procedimentos sistemáticos e objetivos de descrição do conteúdo das mensagens. Assim, proporcionam o levantamento de indicadores, quantitativos ou não, permitindo a realização de inferência de conhecimento [Bardin 1991]. A partir da análise de conteúdo e suas inferências, foi possível avaliar o estado da prática em relação ao uso atual de técnicas e ferramentas de detecção de vulnerabilidades por membros de equipes que utilizam métodos ágeis, bem como motivações, dificuldades, limitações e lições aprendidas na sua implantação.

5. Resultados

A seguir serão apresentados os resultados obtidos do *survey*, a saber, métodos ágeis utilizados pelas equipes, uso atual das técnicas e ferramentas e características da sua implantação.

5.1. Métodos ágeis utilizados pelas equipes

A Tabela 2 mostra os métodos ágeis utilizados pelos membros das equipes. Uma mesma equipe pode desenvolver software com mais de um método ágil, por isso, foram obtidas 245 respostas, sendo 110 o total de respondentes. O cálculo da porcentagem é baseado no total de respostas. Dessa forma, é possível observar que o *framework* "Scrum"é o mais utilizado pelas equipes, obtendo 41,22% de uso. Em seguida, "Kanban", "Extreme Programming (XP)"e "Lean"possuem, respectivamente, 26,94%, 11,43% e 7,76%. Essa grande utilização deve-se ao fato de muitas equipes adotarem o *framework* "Scrum "conjuntamente com esses métodos ágeis para alcançarem seus objetivos de forma mais eficiente e eficaz. Os métodos ágeis menos utilizados, como: "Feature Driven Development (FDD)", "Agile Unified Process (AUP)", "Crystal Methods"e "Dynamic Systems Development Method (DSDM)", possuem, respectivamente, 4,08%, 2,45%, 2,04% e 1,22%. Outros métodos ágeis utilizados somam 2,86% das respostas obtidas.

Tabela 2. Métodos ágeis utilizados

		Quantidade %
Métodos ágeis utilizados	Scrum	41,22
	Kanban	26,94
	Extreme Programming (XP)	11,43
	Lean Software Development	7,76
	Feature Driven Development (FDD)	4,08
	Agile Unified Process (AUP)	2,45
	Crystal Methods	2,04
	Dynamics Systems Development Method (DSDM)	1,22
	Outros	2,86
	Total	**100%**

5.2. Uso atual das técnicas e ferramentas para detecção de vulnerabilidades

A Tabela 3 apresenta a relação das 18 técnicas e ferramentas de detecção de vulnerabilidades listadas de acordo com a revisão bibliográfica realizada. Elas foram classificadas nas seguintes fases: análise de requisitos e modelagem, *design*, codificação, testes e produção. A partir dessa classificação, foi possível analisar a porcentagem do uso atual de cada uma das técnicas e ferramentas, bem como sua utilização por fase do ciclo de vida de desenvolvimento de software. O cálculo está baseado nas 110 respostas obtidas.

Tabela 3. Uso atual de técnicas e ferramentas de detecção de vulnerabilidades

Fase	Técnicas e ferramentas	Sim %	Não %	Sim* %	Não* %
AR e mod.	Especialista em segurança	19,09	66,36	36,36	49,09
	Requisitos de segurança	21,82	63,64		
	Abuse/Misuse Case/Stories	17,27	68,18		
	Modelagem de ameaças	13,64	71,82		
	Modelagem de riscos	16,36	69,09		
	Ferramenta de modelagem de riscos e ameaças	13,64	71,82		
Design	*Design* de segurança	21,82	63,64	40,00	45,45
	Contramedidas de segurança	20,00	65,45		
	Avaliação de vulnerabilidades	32,73	52,73		
Codificação	Codificação segura	44,55	40,91	56,36	29,09
	Ferramenta de análise estática de código	34,55	50,91		
	Revisão de segurança de código	30,91	54,55		
	Ferramenta de revisão de código	41,82	43,64		
Testes	Ferramenta de análise dinâmica de código	31,82	53,64	41,82	43,64
	Teste de penetração	11,82	73,64		
	Fuzz testing	18,18	67,27		
	Teste baseado em riscos	17,27	68,18		
Prod.	Gestão de respostas a incidentes	21,82	63,64	21,82	63,64

* - Média do uso por fase do ciclo de vida — AR e mod. = Análise de requisitos e modelagem

A fase de codificação possui maior grau de utilização de técnicas e ferramentas se comparadas às demais, alcançando 56,36% de utilização. Esse alto grau de uso pode ser explicado pelo fato de a fase de codificação possuir técnicas e ferramentas mais conhecidas e acessíveis, com materiais e treinamentos disponíveis. Na fase de testes, a utilização atual chega a 41,82%. Mesmo possuindo técnicas e ferramentas conhecidas, como teste de penetração, *fuzz testing* e teste baseado em riscos, a porcentagem é baixa se comparado à fase de codificação. Esse dado pode ser explicado pelo fato de algumas técnicas e ferramentas, como o *fuzz testing* e o teste de penetração, terem um alto custo, serem mais complexas e por dificuldades na aquisição de manuais ou treinamentos. A fase de *design* é a terceira que possui técnicas e ferramentas mais utilizadas, alcançando 40% de utilização. Na fase de análise de requisitos e modelagem, o grau de uso está em 36,36%. Esse baixo grau de uso, comparado às outras fases, deve-se possivelmente à falta de adoção de técnicas e ferramentas pela equipe, seja por falta de conhecimento de sua existência, seja por negligencia da segurança durante as fases iniciais dos projetos, seja ainda pela pressão na entrega do produto. O grau de uso da gestão de respostas a incidentes encontra-se em 21,82%. As respostas em branco somam 14,55% das respostas obtidas.

5.3. Características da implantação das técnicas e ferramentas

As características da implantação das técnicas e ferramentas de detecção de vulnerabilidades foram obtidas por meio de perguntas abertas para os 110 membros de equipes

de desenvolvimento de software. As perguntas tratavam das motivações, dificuldades, limitações e lições aprendidas na sua implantação durante o desenvolvimento ágil de software. Todas as perguntas abertas tratadas nesta seção passaram pela técnica de análise de conteúdo conforme descrito na Seção 4. A Tabela 4 apresenta as motivações obtidas para a implantação de técnicas e ferramentas para detecção de vulnerabilidades no desenvolvimento ágil de software.

Tabela 4. Motivação para a implantação de segurança no desenvolvimento ágil de software

Motivações para a implantação de segurança	Quantidade %
Mitigação de riscos e possíveis ameaças	22,73
Qualidade do software	17,27
Exigência de segurança de software pelo cliente	10,91
Melhoria na gestão da segurança nos projetos	1,82
Falta de segurança dos sistemas existentes	0,91
Não há implantação de técnicas e ferramentas de detecção de vulnerabilidades	9,09
Não sei	5,45
Branco	31,82
Total	**100%**

Ao serem questionados sobre as motivações da implantação de técnicas e ferramentas para detecção de vulnerabilidades no desenvolvimento ágil de software, 22,73% dos membros das equipes citaram a mitigação dos riscos e possíveis ameaças como maior motivação. Já para 17,27% dos membros das equipes, a qualidade do software é a motivação mais relevante para seus projetos. Isto pode ser explicado pelo fato de muitas equipes considerarem segurança como um item de qualidade de software, ou seja, quanto mais seguro o software for, maior a sua qualidade. A exigência do cliente por segurança no desenvolvimento de software alcançou 10,91% de motivação. Essa motivação está relacionada a determinado domínio que exige um nível mínimo de segurança, como sistemas críticos, por exemplo. Melhoria na gestão de atividades de segurança e deficiência de segurança dos sistemas existentes obtiveram 1,82% e 0,91%, respectivamente. As dificuldades e limitações encontradas durante a implantação das técnicas e ferramentas estão apresentadas na Tabela 5.

Tabela 5. Dificuldades e limitações encontradas na implantação de atividades de segurança

Dificuldades e limitações	Quantidade %
Falta de conhecimento dos envolvidos no projeto	12,73
Dificuldade na aplicação das técnicas e ferramentas para detecção de vulnerabilidades	7,27
Prazos das entregas	6,36
Falta de sincronia da equipe	5,45
Resistência da equipe ou gerencia em adotar segurança no desenvolvimento de software	5,45
Alto investimento em segurança	1,82
Não há dificuldades na implantação	0,91
A implantação de segurança foi realizada por outra equipe	0,91
Problemas com correções de segurança após a entrega do software	0,91
Manter a compatibilidade com os sistemas legados	0,91
Não há implantação de técnicas e ferramentas de detecção de vulnerabilidades	16,36
Não sei	4,55
Branco	36,36
Total	**100%**

Ao serem questionados sobre as dificuldades e limitações encontradas na implantação das técnicas e ferramentas, 12,73% dos respondentes alegaram a falta de

conhecimento dos envolvidos nos projetos. Diversos fatores podem explicar esse alto índice, como: falta de incentivo à equipe em realizar cursos, falta de treinamentos ou dificuldade na busca por materiais relacionados às técnicas e ferramentas. As dificuldades na aplicação das técnicas e ferramentas obteve 7,27%. Essa dificuldade está relacionada à dificuldade de entendimento dos materiais disponibilizados, escassez de fóruns de discussão sobre possíveis dúvidas de manuseio da ferramenta ou por problemas internos à equipe. Dificuldades e limitações relacionadas ao prazo das entregas obteve 6,36% das respostas. Por conta de os métodos ágeis exigirem entregas frequentes de funcionalidades, a pressão pela entrega dentro dos prazos e custos estipulados podem fazer com que a segurança não seja implantada ou seja implantada de forma incorreta.

A falta de sincronia da equipe foi uma dificuldade para 5,45% dos membros das equipes. Tal fator pode ser ocasionado pela falta de especialista em segurança na equipe, já que não existe um orientador de boas práticas de segurança, bem como um responsável pela definição de políticas de segurança a serem seguidas durante os projetos. Resistência da equipe é um problema para 5,45% dos membros das equipes. Isto ocorre devido ao entendimento de que ferramentas de detecção de vulnerabilidades devem ser implantadas apenas nas fases finais do desenvolvimento ou por considerar inconveniente a implantação de segurança, pois nunca ocorreu um ataque ao sistema. Já o alto investimento em segurança obteve 1,82% das respostas dos membros das equipes. As demais dificuldades e limitações obtiveram 0,91% das respostas. As lições aprendidas durante a implantação das técnicas e ferramentas estão apresentadas na Tabela 6.

Tabela 6. Lições aprendidas com a implantação de técnicas e ferramentas para detecção de vulnerabilidades

Lições aprendidas durante a implantação	Quantidade %
Aumento da qualidade do software	9,09
Investimento em segurança e benefícios futuros	9,09
Feedback e treinamento entre membros da equipe	7,27
Evitar retrabalho por questões de segurança	6,36
Melhoria na aplicação das técnicas e ferramentas	2,73
Proteção da informação	1,82
A implantação de segurança foi realizada por outra equipe	0,91
Possibilidade de aplicar segurança em métodos ágeis	0,91
Não há implantação de técnicas e ferramentas de detecção de vulnerabilidades	20,00
Não sei	6,36
Branco	36,36
Total	**100%**

Em relação às lições aprendidas, 9,09% dos respondentes alegaram que houve o aumento da qualidade do software desenvolvido pela equipe. Ainda 9,09% citaram que o investimento em segurança acarreta benefícios futuros. Esses benefícios estão relacionados à reutilização de componentes considerados seguros em outros projetos, facilidade na correção de possíveis problemas e menor indíce de ocorrências de problemas de segurança na aplicação. *Feedback* e treinamento entre membros da equipe é uma lição aprendida para 7,27% dos respondentes, pois difundir o conhecimento entre os membros, incentivar treinamentos e cursos é a maneira mais acessível para as equipes. Diminuição de retrabalho por questões de segurança, melhoria na aplicação das técnicas e ferramentas e proteção da informação, respectivamente, obtiveram 6,36%, 2,73% e 1,82% das respostas dos membros das equipes. Já para 0,91% dos respondentes, a lição aprendida foi a possibilidade de aplicar segurança em métodos ágeis.

6. Conclusões

Este artigo teve como objetivo identificar e descrever o estado da prática da utilização de técnicas e ferramentas para detecção de vulnerabilidades durante o desenvolvimento ágil de software. Assim, foi possível identificar os métodos ágeis mais utilizados, o uso atual das técnicas e ferramentas, as motivações, as dificuldades, as limitações e as lições aprendidas. Não há indicações de que algum método ágil combine melhor com segurança de software. A forma com que cada equipe implanta as técnicas e ferramentas em métodos ágeis varia de acordo com a necessidade e contexto do projeto.

As motivações para utilização de segurança de software em projetos são diversas como mitigação de riscos e ameaças, garantia da qualidade do software e falta de segurança nos sistemas existentes. Porém, durante a implantação, podem surgir dificuldades e limitações, por exemplo: falta de treinamento da equipe, falta de sincronia dos envolvidos, alto investimento e resistência da equipe. Tais dificuldades podem ser amenizadas com reuniões entre os membros e com a disponibilização de materiais e treinamentos. Os benefícios relatados estão ligados ao aumento da qualidade do software produzido, à proteção da informação, ao treinamento e *feedback* entre membros, à melhoria na aplicação de técnicas e ferramentas e à possibilidade de conciliação entre métodos ágeis e segurança de software. O grau de uso das técnicas e ferramentas mais alto dá-se na fase de codificação, obtendo 37,95%. Isto pode ser explicado pelo fato de as ferramentas dessa fase serem mais conhecidas e flexíveis. Codificação segura, ferramenta de análise estática de código, revisão de segurança e ferramentas de revisão de código, obtêm, respectivamente, 44,55%, 34,55%, 30,91% e 41,82%. Porém, nenhuma delas possuem mais de 50% de uso. Uma limitação do trabalho é que os membros foram avaliados individualmente, o que não permite tirar conclusões sobre a equipe toda. Outra limitação foram a ausência de resposta dos participantes e a falta de clareza em algumas delas. Finalmente, a análise de conteúdo envolve um possível viés do pesquisador que realizou essa tarefa. O alto número de respostas em branco ou que não implantaram reflete a falta de oportunidade ou de preocupação em conciliar segurança e métodos ágeis.

Os resultados do *survey* permitiram avaliar o estado da prática do uso de ferramentas de técnicas e ferramentas para detecção de vulnerabilides em equipes ágeis brasileiras. Eles indicam que há uma parcela significativa de membros de equipes ágeis que relatam não saber ou que não responderam quais as motivações, as dificuldades ou as lições aprendidas no uso dessas técnicas. Além disso, o uso atual das técnicas e ferramentas é abaixo de 50% em todas as fases de desenvolvimento. Esses dados indicam que atividades de segurança são ainda pouco utilizadas por esses profissionais. Possíveis ações como a colaboração entre membros, o compartilhamento de informação e a auto-organização por meio de treinamentos e disponibilização de material podem fazer com que as melhores técnicas e ferramentas sejam adotadas pelas equipes ágeis.

Como trabalhos futuros, pretende-se estabelecer correlações entre os dados obtidos, por exemplo, grau de aptidão nas técnicas e ferramentas e seu uso atual; método ágil utilizado e dificuldades percebidas; e uso de técnica ou ferramenta e benefício observado.

Referências

Ambler, S. W. (2006). "The agile unified process (AUP)". Disponível em: <http://www.ambysoft.com/unifiedprocess/agileUP.html>.

Bardin, L. (1991). *Análisis de contenido*, volume 89. Ediciones Akal.

Beck, K. (2000). *"Extreme programming explained: embrace change"*. Addison-Wesley Professional, Boston, MA, USA, 2ª edition.

Beck, K., Beedle, M., et al. (2001). "Manifesto ágil de software". Disponível em: <http://agilemanifesto.org >.

Cockburn, A. (2004). *"Crystal clear: a human-powered methodology for small teams"*. Addison-Wesley Professional, USA, 1ª edition.

Cruzes, D. S., Felderer, M., Oyetoyan, T. D., Gander, M., and Pekaric, I. (2017). "How is security testing done in agile teams? a cross-case analysis of four software teams". In *International Conference on Agile Software Development.*, pages 201–216. Springer.

Howard, M. and Lipner, S. (2006). *"The security development lifecycle"*. Microsoft Press, WA, USA, 1ª edition.

Khaim, R., Naz, S., Abbas, F., Iqbal, N., Hamayun, M., and Pakistan, R. (2016). A review of security integration technique in agile software development. *International Journal of Software Engineering & Applications*, 7(3).

Khan, M. U. A. and Zulkernine, M. (2009). "A survey on requirements and design methods for secure software development". *School of Computing, Queen's University, Canada.*

Nist (2009). "National institute of standards and technology: national vulnerability database". Disponível em: <https://www.nist.gov/national-vulnerability-database>.

OWASP (2010a). "CLASP concepts". Disponível em: <https://www.owasp.org >.

OWASP (2010b). "Top ten project". Disponível em: <https://www.owasp.org/index >.

Oyetoyan, T. D., Cruzes, D. S., and Gilje, M. J. (2016). "An empirical study on the relationship between software security skills, usage and training needs in agile settings". *Availability, Reliability, and Security in Information Systems.*

Palmer, S. R. and Felsing, M. (2001). *"A practical guide to feature-driven development"*. Pearson Education, Upper Saddle River, United States, 1ª edition.

Poppendieck, M. and Poppendieck, T. (2003). *"Lean software development: an agile toolkit"*. Addison-Wesley, United States, 1ª edition.

Pressman, R. S. (2011). *"Engenharia de software: uma abordagem profissional"*. McGraw Hill, NY, EUA, 7ª edition.

Schwaber, K. S. (2013). "Um guia definitivo para o SCRUM: as regras do jogo". 26:3–15.

Seacord, R. C. and Householder, A. D. (2005). "A structured approach to classifying security vulnerabilities". Technical report, Carnegie-Mellon University.

Skarin, M. (2015). *"Real-world Kanban: do less, accomplish more with lean thinking"*. The Pragmatic Bookshelf, USA, 1ª edition.

Sommerville, I. (2011). *"Software engineering"*. Pearson Education, Boston, Massachusetts, 9ª edition.