UNIVERSIDADE
FEDERAL RURAL
DE PERNAMBUCO

Carlos Magnum Matias Bezerra

Políticas para desenvolvimento de software seguro em times ágeis.

Recife

2019

Carlos Magnum Matias Bezerra

Políticas para desenvolvimento de software seguro em times ágeis.

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Universidade Federal Rural de Pernambuco – UFRPE

Departamento de Computação

Curso de Bacharelado em Ciência da Computação

Orientador: Prof.ª Dr.ª Suzana Cândido de Barros Sampaio

Recife

2019

Dados Internacionais de Catalogação na Publicação
Universidade Federal Rural de Pernambuco
Sistema Integrado de Bibliotecas
Gerada automaticamente, mediante os dados fornecidos pelo(a) autor(a)

C284p Bezerra, Carlos Magnum Matias
Políticas para desenvolvimento de software seguro em times ágeis. / Carlos Magnum Matias Bezerra. - 2019.
110 f.

Orientadora: Suzana Cândido de Barros Sampaio.
Inclui referências e apêndice(s).

Trabalho de Conclusão de Curso (Graduação) - Universidade Federal Rural de Pernambuco, Bacharelado em Ciência da Computação, Recife, 2019.

1. Segurança de Software. 2. Desenvolvimento Seguro de Software. 3. Desenvolvimento Ágil. 4. Políticas de Desenvolvimento Seguro. 5. Metodologias Ágeis. I. Sampaio, Suzana Cândido de Barros, orient. II. Título

CDD 004

MINISTÉRIO DA EDUCAÇÃO E DO DESPORTO
UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO (UFRPE)
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

http://www.bcc.ufrpe.br

FICHA DE APROVAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO

Trabalho defendido por **CARLOS MAGNUM MATIAS BEZERRA** às 13:00 do dia 06 de dezembro de 2019, no Auditório do Departamento de Computação - DC – Sala 07, como requisito para conclusão do curso de Bacharelado em Ciência da Computação da Universidade Federal Rural de Pernambuco, intitulado " **Políticas para desenvolvimento de software seguro em time ágeis**", orientado por Suzana Cândido de Barros Sampaio e aprovado pela seguinte banca examinadora:

Suzana Cândido de Barros Sampaio
DC/UFRPE

Carlos Julian Menezes de Araujo
DC/UFRPE

À todos aqueles que buscam por uma jornada que nunca vai terminar, mas sempre nos levar ao próximo a desafio ser vencido...

Agradecimentos

Agradeço à minha família, por prover suporte em todos os momentos de minha vida pessoal, profissional e acadêmica. Em especial a minha mãe Eliane Maria, por ser essa mulher guerreira, que apesar de todas as dificuldades me criou, com incentivos e cobranças para que eu chegasse a momentos como esse, de fim de um ciclo e de início de outros. A meu pai Pedro Alexandre por sempre me incentivar a estudar e crescer como pessoa. A minhã irmã Maria Luciana por servir de exemplo de superação em diversos aspectos da minha vida e ter contribuindo ativamente na escrita desse trabalho. Ao meu irmão Pedro Henrique e a minha irmã Heloisa por estarem presentes na minha vida, provendo momentos de descontração e diversão. Aos todos os meus tios, pelo carinho, puxões de orelha, comemorações e incentivo sempre.

A meus colegas de curso, que me ajudaram a passar por todos os momentos dentro da UFRPE, tantos pelos de alegria e conquistas, quanto pelos de dificuldades e superação. Sempre com tempo para conversar de tudo um pouco, de ajudar durante o decorrer das disciplina, de sair para festas para descontrair e outros tantos momentos que passamos juntos.

Aos professores e funcionários da UFRPE por estarem dispostos a me passar conhecimento e me ajudar nos problemas durante esse período.

A minha orientadora, Suzana Sampaio, pela paciência necessária para me orientar, por ter me feito acreditar que era possível e que não podemos desistir.

A meus companheiros da Tempest, que me fizeram crescer como profissional, me incentivaram e auxiliaram a concluir esse trabalho.

Aos meus professores e colegas da UFCG e do IFPE-BJ que fizeram parte desse caminho que percorri para chegar até aqui.

E a tantas outras pessoas que me apoiaram e incentivaram durante minha vida.

"Milagres só acontecem com as pessoas que não desistem."
(Emporio Ivankov - One Piece)

Resumo

Os valores e princípios das metodologias ágeis visam entregar valor ao negócio de forma rápida, sendo essa agilidade, um dos fatores críticos para o sucesso dessas abordagens. Sucesso esse, crucial para atender uma demanda crescente por *software*, gerada pela necessidade de digitalização dos negócios. Atualmente, outra preocupação, também advinda dessa necessidade, vem ganhando mais visibilidade, a segurança cibernética. Muitos negócios que migraram para o mundo digital não tiveram a devida preocupação com as ameaças presentes nesse novo cenário. Como consequência, muitos problemas de segurança vem sendo expostos nos últimos anos, tais problemas, muitas vezes, acarretam em prejuízos de natureza financeira e social. Sendo *software* um dos ativos computacionais mais expostos a ameaças de segurança, a busca por desenvolvê-los de forma segura, torna-se uma demanda, cada vez mais evidente. Neste cenário, surge a necessidade de incluir práticas de segurança no cotidiano dos times de ágeis. Com o intuito de contribuir para a resolução desses desafios, este trabalho tem como objetivo: estabelecer políticas de desenvolvimento seguro, agregando práticas destinadas a inclusão de atributos de segurança em projetos de *software*, a serem desenvolvidos por times ágeis. Para tanto, se fez necessário, realizar uma pesquisa exploratória por meio de um levantamento bibliográfico, a fim de se obter uma compreensão inicial do problema e das principais abordagens utilizadas para tratá-lo. A partir da bibliografia encontrada, foram consideradas as pesquisas que apresentavam práticas para a inclusão de atividades de segurança em projetos de software. Com base em critérios predefinidos, foi conduzida uma filtragem inicial das práticas levantadas. O sub-conjunto resultante, foi avaliado por um time ágil de desenvolvimento, com experiencia em segurança. O resultado dessa avaliação, evidenciou as 14 (catorze) práticas mais relevantes e as etapas do processo de desenvolvimento, ao qual elas pertenciam. A partir das práticas selecionadas, 5 (cinco) políticas foram estruturadas e organizadas, em seguida avaliadas por especialistas, através de entrevistas, para julgar, seus benefícios para segurança e aplicabilidade em times ágeis. Por meio do resultado dessa avaliação, foi possível concluir, que 3 (três) das políticas, eram suficientemente aplicáveis a times ágeis e contribuíam para segurança do projeto. Já 2 (duas) delas, apesar dos benefícios para segurança, ainda necessitam de ajustes para que a incorporação seja viável, no mundo ágil.

Palavras-chave: Segurança de Software, Segurança, Desenvolvimento Seguro de Software, Desenvolvimento Ágil, Políticas de Desenvolvimento Seguro, Metodologias Ágeis.

Abstract

The values and principles of agile methodology ain to aggregate value to the business in a quick manner, being this agility, one of the critics factors for the success of this approaches. This success is crucial to attend the constant rising demand for software, generated by the necessity of digital transformation. Currently, another concern, also resulting of this necessity, has gained more visibility, the cyber security. A lot of companies that have migrated for the digital world did not have the due caution with the present threats in this scenery. As consequence, many security problems have being espoused in this past few years, this problems, tent to result in social and financial losses. Software being one of the computational assets with the biggest expulser to this threats, the search for cybersecurity tend to become a demand. In this scenery, emerge the necessity to include security practices to the everyday live of the agile team. With the intent to contribute to the resolution of this task, this paper has as his goal to: establish cyber security policies, aggregating practices destined to the inclusion of security attributes to software projects, being developed for agile teams. To achieve this, it was necessary to perform a exploratory research through a bibliographic survey, to obtain an initial understanding of the problem and of the main approaches used to solve it. From the bibliography that it was found it was considered only the research that present practices that included cyber security activities on software projects. Based on predefined criteria, an initial filtering of the surveyed practices was conducted. The resulting subset was evaluated by an agile development team with security expertise. The result of this evaluation evidenced the 14 (fourteen) most relevant practices and the stages of the development process to which they belonged. From the selected practices, five (5) policies were structured and organized, then evaluated by experts, through interviews, to judge their benefits for safety and applicability in agile teams. From the result of this assessment, it was concluded that 3 (three) of the policies were sufficiently applicable to agile teams and contribute to project safety. Already 2 (two) of them, despite the security benefits, still need adjustments to make the incorporation viable in the agile world.

Keywords: Software Security, Security, Secure Software Development, Agile Development, Secure Development Policies, Agile Methodologies.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

MSDL	Microsoft Software Development Life Cycle
CLASP	Comprehensive, Lightweight Application Security Process
SSTP	Software Security Touchpoints
CC	Common Criteria
FDD	Feature Driven Development

Sumário

1 INTRODUÇÃO

O processo de desenvolvimento de *software* é um tema amplamente pesquisado, tendo em foco a necessidade de aperfeiçoar as metodologias utilizadas. Cada vez mais, organizações buscam por métodos que atendam suas necessidades de se adaptar a ambientes complexos e mudanças sucessivas (STOICA; MIRCEA; GHILIC-MICU, 2013). As metodologias ágeis surgiram formalmente em 2001, com o lançamento do manifesto ágil (MANIFESTO..., 2001). Tais metodologias buscavam atender essas necessidades, por meio de mecanismos que possibilitem, adaptação à mudança em qualquer estágio do desenvolvimento, entregas constantes de *software* funcionando em períodos curtos de tempo e colaboração contínua com o cliente (HIGHSMITH; COCKBURN, 2001).

O grande volume e a movimentação constante de dados na internet torna-se, cada vez mais, um alvo para indivíduos mal intencionados, que detenham capacidade intelectual e computacional. Com isso também cresce a preocupação de empresas e do público comum com a segurança dos seus dados, os quais, muitas vezes, estão vulneráveis na *web*. Vulnerabilidade, essa, comprovada por meio dos casos de ciberataques amplamente noticiados em vários tipos de mídias, tais como: o vazamento de dados de clientes da Vivo Vivo... (2019), a invasão do celular do ministro da justiça brasileiro 'Spoofing'... (2019), o *malware Wanna Cry* Entenda... (2017), entre muitos outros.

Esses ataques, geralmente, têm como alvos principais as empresas. Porém, não estão limitados ao escopo corporativo, podendo, muitas vezes, alcançar os usuários comuns. Os ataques cibernéticos são realizados tendo como finalidade obter dados sigilosos, ocultar informações, de alguma maneira, valiosas, retaliar determinadas ações governamentais ou da iniciativa privada ou, em alguns casos, pelo simples prazer gerado pelo desafio de invadir sistemas computacionais. Sejam quais forem as intenções, esses ataques sempre geram danos financeiros ou de outro caráter para aqueles que tiveram seus dados comprometidos, logo, atualmente, é crescente a busca por segurança computacional (BODDEN, 2018). Diversas iniciativas podem comprovar o crescimento dessas preocupações, são exemplos: o *Project Zero* do Google Project... (2019), o projeto *The Open Web Application Security Project* (OWASP) OWASP... (2019) e a criação de leis de proteção de dados, como a LGPD (Brasil) LGPD... (2019) e a GPDR (Reino Unido) GDPR (2018).

Desenvolver *softwares* com atributos de segurança em seus requisitos, como forma de prevenir ataques, é essencial (ESSAFI; LABED; GHEZALA, 2007). A necessidade crescente por segurança de software e a adoção das metodologias ágeis como forma de desenvolver software de modo eficiente, levantam questões relativas à integração desse dois conceitos. Conciliar estes dois mecanismos no processo desenvolvimento é um problema de pesquisa para diversos estudiosos (Oueslati; Rahman; Othmane, 2015), (Barbosa; Sampaio, 2015), (BACA; CARLSSON, 2011), (Terpstra; Daneva; Wang, 2017), (NICOLAYSEN et al., 2010), (BACA, 2012), (SIPONEN; BASKERVILLE; KUIVALAINEN, 2005), (ESSAFI; LABED; GHEZALA, 2007), (ADELYAR; NORTA, 2016), (BEZNOSOV; KRUCHTEN, 2004).

Desta forma é aceitável levantarmos seguinte questão: É possível estabelecer políticas de segurança, agregando práticas já apontadas como eficazes e aplicáveis a times ágeis, visando facilitar o entendimento e uso dessas? A partir de buscas na literatura e compreensão de como aplicar práticas de segurança a projetos de *software*, este trabalho propõe políticas de segurança, que atendam times ágeis de maneira sucinta e objetiva. Essas políticas devem ser refinadas e confirmadas juntos a profissionais com experiência tanto em desenvolvimento e métodos ágeis, quanto com expertise em segurança, de modo a comprovar ou não sua eficácia.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Estabelecer políticas de desenvolvimento seguro, agregando práticas destinadas a inclusão de atributos de segurança em projetos de software, a serem desenvolvidos por times ágeis.

1.1.2 Objetivos Específicos

- Levantar práticas de segurança para desenvolvimento ágil de projetos;
- Propor políticas para o desenvolvimento de *software* seguro, agregando práticas destinadas a inclusão de atributos de segurança em projetos de *software*, com base em revisão da literatura;
- Avaliar as políticas, junto a especialistas de segurança, quanto a sua completude, consistência e aplicabilidade em times ágeis;

1.2 JUSTIFICATIVA

É crescente a dependência que diversos negócios tem do uso de ferramentas *software*. Some-se a isso o fato de que diversas falhas, em termos de requisitos de segurança, descobertas, são decorrentes da falta de atividades de segurança durante os desenvolvimento dessas ferramentas (KANNIAH; MAHRIN, 2016). Apesar disso, na busca por entregar produtos de *software* de forma mais eficiente e com maior valor para o cliente, muitas empresas de desenvolvimento migraram seus processos para metodologias ágeis, tais como: Scrum e XP, as quais possibilitam desenvolvimento de modo mais rápido (BACA; CARLSSON, 2011). Contudo, o uso dessas metodologias impõe algumas limitações na inclusão de atividades de segurança (BACA; CARLSSON, 2011), (Villamizar et al., 2018), (Terpstra; Daneva; Wang, 2017), (NICOLAYSEN et al., 2010) e (Bansal; Jolly, 2014).

Nesse contexto é possível estabelecer a necessidade de pesquisar formas de incluir atividades de segurança em processos de times de desenvolvimento ágil. (Villamizar et al., 2018) apontam para a falta de uma visão integrada de requisitos de segurança nas práticas ágeis. Empresas de pequeno e médio porte, que aplicam metodologias ágeis em seus times, geralmente não utilizam nenhuma prática específica para tratar problemas de segurança, mesmo que seus produtos estejam expostos a riscos (NICOLAYSEN et al., 2010). Outros autores, como (Barbosa; Sampaio, 2015), (BOWEN et al., 2014), (DAUD, 2010) indicam, de modo contundente, a importância de integrar segurança ao processo de desenvolvimento e não, apenas, como parte isolada dele. A partir dessas perspectivas é possível notar, a importância de buscar por formas de integrar o modelo de desenvolvimento ágil e a inclusão de atividades voltadas à segurança.

1.3 ORGANIZAÇÃO

O presente trabalho está organizado da seguinte forma: Capítulo 1: a introdução, aqui apresentada, incluindo contextualização da temática, objetivos geral e específicos, justificativa e esta organização do trabalho. Capítulo 2: traz conceitos relacionados a metodologias ágeis, desenvolvimento e segurança de *software* são abordados, de maneira a possibilitar uma base de conhecimento para que o leitor consiga entender a proposta defendida nesta pesquisa. Capítulo 3: descreve a metodologia de pesquisa utilizada. Capítulo 4: propõe políticas de segurança, as quais foram definidas na pesquisa. Capítulo 5: avalia as políticas propostas por meio da opinião de profissionais. Capítulo 6: apresenta as conclusões acerca da pesquisa realizada, as

considerações finais e proposta de trabalhos futuros relacionados ao tema.

2 REFERENCIAL TEÓRICO

2.1 METODOLOGIAS ÁGEIS

O termo ágil surgiu no início de 2001 durante um encontro entre um grupo de desenvolvedores, o qual resultou no lançamento do manifesto ágil de desenvolvimento Agile... (2019). O principal foco desta metodologia são as pessoas e como elas desempenham seu papel, tendo sempre a colaboração e a auto organização dos time como premissas Agile... (2019). O manifesto ágil é composto por 4 valores, quais sejam:

1. Indivíduos e interações sobre processos e ferramentas;
2. *Software* que trabalha sobre uma documentação completa;
3. Colaboração do cliente na negociação do contrato;
4. Responsividade a mudanças após seguir um plano.

Além desse valores o manifesto apresenta 12 princípios, os quais são descritos abaixo:

1. Garantir a satisfação do cliente por meio da entrega contínua de *softwares* em funcionamento;
2. Mudanças nos requisitos são bem vindas, ainda que tardias, desde que possibilitem o alcance de vantagens competitivas aos clientes;
3. As entregas aos clientes devem ser frequentes;
4. Colocar pessoas de negócio e desenvolvedores para trabalhar juntos diariamente durante todo o projeto;
5. Construir projetos em torno de indivíduos motivados, fornecendo um ambiente de confiança para que eles façam o seu trabalho;
6. A maneira mais eficiente de trocar informações dentro de um time de desenvolvimento é a comunicação face a face;
7. *Software* funcionando é a principal medida de progresso;

8. Processos ágeis devem promover desenvolvimento sustentável, de modo que patrocinadores e desenvolvedores sejam capazes de manter um ritmo constante indefinidamente;

9. Atenção contínua a excelência técnica e bom design melhoram a agilidade;

10. Simplicidade é essencial;

11. As melhores arquiteturas, requisitos e projetos emergem de times auto organizáveis;

12. Em intervalos regulares o time se reúne, para refletir sobre como se tornar mais efetivo e, apropriadamente, ajustar comportamento.

Após o surgimento das metodologias ágeis, foi crescente a adoção desses métodos em empresas de desenvolvimento de *software* (CHOUDHARY; RAKESH, 2016), (STOICA; MIRCEA; GHILIC-MICU, 2013). Diversos *frameworks* foram criados com base nesse conceito abrangente, Scrum (SUTHERLAND; SCHWABER, 2013), XP (BECK, 2000), Crystal (COCKBURN, 2004) e FDD (PALMER; FELSING, 2001), são alguns exemplos. Alguns conceitos dessa metodologia são abordados durante essa pesquisa. O *Product Backlog* é uma lista de todos os requisitos conhecidos do projetos (SUTHERLAND; SCHWABER, 2013). A *Definition of "Done"* é que todas as atividades do *Product Backlog* devem ter claramente uma definição das condições para que sejam aceitas como concluídas (SUTHERLAND; SCHWABER, 2013). *Pair Programming* é um métodos que permite que dois desenvolvedores implementem, de forma colaborativa, uma atividade do projeto (AGILE..., 2019). *Planning Poker* é um *"game"* para estimativa, usado por muitas equipes ágeis, em que os membros do time em consenso, atribuem um valor de complexidade para as atividades, geralmente *User Stories* (AGILE..., 2019). *User Stories* são uma maneira de descrever comportamento do usuário de acordo com os requisitos levantados para o *software* (AGILE..., 2019).

Os métodos ágeis obtiveram vantagens sobre as metodologias clássicas, devido a sua capacidade de se adaptar a mudanças de forma eficiente (STOICA; MIRCEA; GHILIC-MICU, 2013). Devido a complexidade dos ambientes corporativos, agilidade se tornou uma condição necessária para obtenção de vantagem competitiva no mercado (STOICA; MIRCEA; GHILIC-MICU, 2013). As entregas interativas e geralmente com prazos curtos, normalmente semanas, dos métodos ágeis, agregam valor ao *software*, de maneira rápida (STOICA; MIRCEA; GHILIC-MICU, 2013).

Apesar das vantagens apresentadas, esse tipo de abordagem possui algumas limitações, como a dependência de cooperação e comunicação efetiva com o cliente, para o sucesso do projeto (CHOUDHARY; RAKESH, 2016). Para aplicar métodos ágeis com sucesso, são necessárias algumas premissas, como comunicação eficiente, evoluir e adaptar os requisitos constantemente e manter um time tecnicamente preparado e motivado (CHOUDHARY; RAKESH, 2016).

Devido a crescente adoção das metodologias ágeis, outra preocupação emergente para as metodologias é a inclusão atividade de segurança no processo. Desenvolvimento seguro é um tópico cada vez mais abordado, devido às diversas ameaças as quais um *software* está exposto. Para abordar o tema, é necessário que tenhamos atividades voltadas para incluir segurança, dentro do processo de desenvolvimento ágil, conservando seus valores e princípios.

2.2 DESENVOLVIMENTO DE *SOFTWARE* E SEGURANÇA

Ciber Segurança é uma coleção de ferramentas, políticas, conceitos, salvaguardas, diretrizes, gerenciamento de riscos, ações, treinamento, práticas, garantias tecnológicas e tecnologias que pode ser usada para proteger os ambientes, a organização e os ativos dos usuários (SOLMS; NIEKERK, 2013). Organização e ativos de usuário incluem dispositivos de computação conectados, pessoal, infraestrutura, aplicativos, serviços, sistemas de telecomunicações e a totalidade das informações transmitidas e/ou armazenadas no ambiente cibernético (SOLMS; NIEKERK, 2013).

A segurança tem a intenção de garantir a obtenção e manutenção das propriedades computacionais da organização e ativos do usuário contra riscos relevantes, presentes na internet. Para isso é necessário garantir três atributos principais: confidencialidade, integridade e disponibilidade (SOLMS; NIEKERK, 2013). A confidencialidade é uma propriedade que garante a ausência de divulgação não permitida pelo proprietário das informações (Barbosa; Sampaio, 2015). O atributo de integridade garantem que não haverão alterações, não autorizadas (Barbosa; Sampaio, 2015). A disponibilidade está associada à pontualidade do serviço ou das informações fornecidas pelo sistema (Barbosa; Sampaio, 2015).

Software é um ativo computacional de uma organização, logo medidas de segurança adequadas devem ser aplicadas a este. É possível definir *software* seguro como sendo aquele que, racionalmente, é plausível afirmar que não pode ser intencionalmente forçado a falhar e continua a funcionar da maneira como foi projetado, mesmo quando sob tentativas de comprometer sua confiabilidade (Oueslati; Rahman;

Othmane, 2015), (MCGRAW, 2004). Nesse aspecto, para afirmar que determinado *software* é seguro, devemos apresentar provas conclusivas de que este não está vulnerável a ataques para os quais foi desenvolvido para estar protegido. Para atingir este objetivo, alguns princípios de design de segurança, são considerados muito importantes (SAFECODE..., 2018), são eles:

- *Economy of Mechanism*: mantenha o design do sistema o mais simples e pequeno possível;
- *Fail-safe Defaults*: baseie as decisões de acesso em permissão implícita ao invés de negação.
- *Complete Mediation*: todo acesso a cada objeto deve ser verificado para autorização.
- *Least Privileges*: todo programa e todo usuário do sistema deve operar usando o mínimo dos privilégios necessários para concluir o trabalho.
- *Least Common Mechanism*: minimize a quantidade de mecanismo comuns a mais de um ou a todos os usuários.
- *Psychological Acceptability*: é essencial que a interface seja projetada para facilitar uso, fazendo com que usuários apliquem, rotineira e automaticamente, os mecanismos de proteção corretamente.
- *Compromise Recording*: algumas vezes é sugerido, que mecanismos que registrem com segurança, que um comprometimento da informação ocorreu pode ser usado no lugar de mecanismos mais complexos, que impedem completamente a perda.
- *Defense in Depth*: projete o sistema para resistir a ataques, mesmo que uma única vulnerabilidade seja descoberta ou que um único recurso de segurança venha a ser subvertido. A defesa em profundidade pode envolver vários níveis de segurança ou projetar o sistema para travar, ao invés de permitir que o invasor obtenha controle completo.
- *Fail Securely*: um contraponto à defesa em profundidade é que um sistema deve ser projetado para permanecer seguro, mesmo que encontre um erro ou trave.
- *Designing for Update*: é provável que nenhum sistema permaneça livre de vulnerabilidades de segurança para sempre, portanto, os desenvolvedores devem planejar a instalação segura e confiável de atualizações de segurança.

Seguir os princípios acima apresentados pode ser uma tarefa desafiadora, contudo, ao seguir estas recomendações, o processo de desenvolvimento está mais propenso a atingir os objetivos de segurança (SAFECODE..., 2018). Segurança deve ser considerada um requisito não funcional crítico, o qual necessita ser incorporado integralmente pelas metodologias de desenvolvimento (Villamizar et al., 2018). Vulnerabilidades de *software*, normalmente, são causadas por negligências em aspectos de segurança durante o desenvolvimento (KANNIAH; MAHRIN, 2016). Falta de requisitos de segurança (Villamizar et al., 2018), falhas técnicas de implementação (FRIJNS; BIERWOLF; ZIJDERHAND, 2018), (ANIS et al., 2018), ausência de validação de segurança (ESSAFI; LABED; GHEZALA, 2007) e falta de experiência e cuidado com segurança por parte dos envolvidos no projeto (BOWEN et al., 2014), podem ser citados como alguns desses aspectos.

Detectar e corrigir falhas de segurança, pontualmente, é uma abordagem muito comum durante o desenvolvimento de *software*, porém muitos autores apontam para sua ineficiência (ESSAFI; LABED; GHEZALA, 2007). Logo uma das principais preocupações, ao incorporar segurança em um projeto de *software*, é garantir que sejam consideradas todas as fases do ciclo de desenvolvimento (KANNIAH; MAHRIN, 2016), (Sodanil et al., 2015), (NICOLAYSEN et al., 2010), (Oyetoyan; Cruzes; Jaatun, 2016), (BOWEN et al., 2014), (SIPONEN; BASKERVILLE; KUIVALAINEN, 2005), (KERAMATI; MIRIAN-HOSSEINABADI, 2008). O ciclo de desenvolvimento de *software* seguro inclui os processos de treinamento, requisitos, planejamento, implementação, verificação, entrega e resposta (Sodanil et al., 2015). Cada um desses processos deve conter suas práticas relacionadas à segurança, as quais serão aplicadas durante as fase do ciclo de vida de desenvolvimento.

O processo de desenvolvimento seguro é o principal meio, utilizado por empresas de engenharia de *software*, para a adoção de práticas de segurança (Oueslati; Rahman; Othmane, 2015). É possível encontrar diversos processos com esse propósito, os quais são apresentados por organizações presentes no mercado. O *Security Development Lifecycle da Microsoft* (MSDL) Microsoft... (2019), o *Comprehensive, Lightweight Application Security Process* (CLASP) do *The Open Web Application Security Project* (OWASP) (CLASP, 2019), são exemplos desses processos. Além disso, tem-se conjuntos de práticas e normas para desenvolvimento seguro, como *Seven Touchpoints for Software Security* (TP) Seven... (2019) e *Common Criteria* (CC) common... (2019).

A seguir apresentamos como as metodologias ágeis estão sendo avaliadas no contexto de segurança.

Processo	**Descrição para Segurança**
Treinamento	Um empenho para entender conceitos básicos e recentes para desenvolvimento de software seguro e privacidade, pode ajudar bastante as organizações a reduzir o número de vulnerabilidade no software.
Requisitos	Considerar bases de segurança e privacidade, para analisar como alinhar qualidade e requisitos funcionais, a custos e regras de negócio.
Planejamento	Incluir especificações de projeto e realizar análise de risco, para ajudar a mitigar riscos de segurança e privacidade durante o andamento do projeto.
Implementação	Ajudar o usuário final em decisões sobre formas seguras de implantar o software. Estabelecer práticas recomendadas para evitar, detectar e remover problemas de segurança do código.
Verificação	Garantir que o software desenvolvido cumpre os padrões de segurança e privacidade estabelecidos nas fases anteriores.
Entrega	Se preparar para a entrega do produto, incluindo planos para realizar tarefas pós entrega e tratar vulnerabilidades que venham a acontecer.
Resposta	Estar disponível para esclarecer dúvidas relacionadas a ameaças e vulnerabilidades emergentes.

Tabela 1 – Processos de desenvolvimento seguro.

Fonte – Adaptado de (Sodanil et al., 2015)

2.3 METODOLOGIAS ÁGEIS E SEGURANÇA

Como demonstrado, anteriormente, metodologias ágeis vem sendo amplamente utilizadas para o desenvolvimento de projetos de *software*. Porém, muitos autores (ADELYAR; NORTA, 2016), (Oueslati; Rahman; Othmane, 2015), (BACA; CARLSSON, 2011), apontam a falta de suporte para atividades de segurança no ciclo de desenvolvimento ágil. Este problema ocorre porque a maioria das práticas de segurança não foram feitas para os métodos ágeis (Barbosa; Sampaio, 2015). Esta característica é apontada como uma dificuldade natural dos métodos ágeis em lidar com requisitos não funcionais (Villamizar et al., 2018), (Terpstra; Daneva; Wang, 2017). Além disso, segundo (NICOLAYSEN et al., 2010), caso as práticas ágeis sejam seguidas, rigorosamente, o cliente precisa requisitar e priorizar segurança para que ela seja incluída como requisito. Apesar de todos os contrapontos apresentados, adequar práticas de segurança ao mundo ágil é uma tarefa importante e diversas abordagens e avaliações já foram realizadas com esse intuito, o que será demonstrado a seguir.

Em (Oueslati; Rahman; Othmane, 2015), foram levantados 20 desafios para aplicação de práticas de segurança em metodologias ágeis. Todos os desafios apresentados foram avaliados, de modo a apresentar quais deles tinham relação com os valores e princípios do manifesto ágil. Esse trabalho foca em levantar temas carentes de análise, dentro do contexto de segurança no mundo ágil. Já (Barbosa; Sampaio, 2015) foca na apresentação de práticas que contribuam com a melhora do processo de desenvolvimento de *software* ágil do ponto de vista de segurança. Para cada prática, foi apresentada uma descrição e seus benefícios, além de uma validação por meio de entrevistas com especialistas para corroborar com sua importância. As autoras busca guiar a inclusão de segurança em projetos ágeis de maneira simplificada.

(BACA; CARLSSON, 2011) apresenta uma abordagem mais quantitativa de avaliação de práticas de segurança para desenvolvimento de *software*. No estudo foram levantadas diversas práticas de diferentes processos de desenvolvimentos seguro, para cada uma delas o autor solicita as especialistas que pontuem-nas, quanto a custo e benefício. A partir dessa pontuação foi proposto um novo processo ágil de desenvolvimento seguro, contendo as práticas com melhor desempenho. Outro método quantitativo é usado por (Bansal; Jolly, 2014), para demonstrar compatibilidade de práticas de segurança com características ágeis. Por meio do cálculo do valor de agilidade média, apresentado no trabalho, é possível aferir, quão ágil é determinada prática. O intuito principal é indicar quais práticas são mais compatíveis e elencar suas fases no ciclo de vida do processo de desenvolvimento.

Uma análise da integração entre o processo de desenvolvimento ágil e atividades relacionadas a segurança é apresentada em (SINGHAL et al., 2012). Para realizar esta análise, a pesquisa cita as atividades de segurança adotadas e as características ágeis desejadas. Em cima destes conceitos foi realizado o cálculo do CASAF (*Comparative analysis of security activity factor*), um índice que deve indicar o grau de integração que a atividade possui em relação às características ágeis desejadas. Se tratando de uma pesquisa quantitativa, apresenta um método discreto, para comparar atividades de segurança em relação a aplicação em métodos ágeis, porém, não apresenta comprovação da efetividade do método e das práticas de segurança adotadas.

(KERAMATI; MIRIAN-HOSSEINABADI, 2008) apresenta a integração de atividades de segurança no processo ágil, através do cálculo do grau de decaimento de agilidade para cada uma das atividades propostas, em relação a características ágeis. A partir dos resultados, o artigo indica o uso dos dados coletados, a partir de uma medida chamada *Agility Reduction Tolerance*, como forma de selecionar as

atividades que devem ser incluídas. A proposta não apresenta nenhuma forma de validação da efetividade do método apresentado.

(Villamizar et al., 2018) aponta uma análise sistemática de abordagens para adicionar segurança no processo de requisitos do metodologias ágeis. São apresentadas 21 abordagens, cada uma delas é analisada, de modo a encontrar respostas para as questões levantadas pelo autor. A pesquisa tenta congregar uma visão unificada da compreensão do tema abordado (ADELYAR; NORTA, 2016) ao expor um estudo de caso, em que são levantados e explanados desafios relacionadas a segurança para com as metodologias ágeis. A partir de entrevistas com 4 times de desenvolvimento, a pesquisa apresenta uma avaliação das atividades realizadas por desenvolvedores e clientes, em relação aos princípios ágeis por esta expostos.

(Terpstra; Daneva; Wang, 2017) demonstra uma percepção sobre segurança no mundo ágil, extraída de postagens de praticantes de métodos ágeis, sobre o tema, em fóruns de discussão em redes sociais. Essa pesquisa levanta conceitos que indicam problemas relacionados ao tema, os quais são divididos em 5 (cinco) categorias distintas. Além disso, o estudo apresenta conceitos que indicam estratégias copiadas de outros times, as quais foram distribuídas em 3 categorias. O trabalho tem enfoque na interação entre profissionais de times ágeis e requisitos de segurança.

(NICOLAYSEN et al., 2010) conduz uma série de entrevistas em organizações que utilizam metodologias ágeis para desenvolvimento de *software*. O intuito foi compará-las a esforços presentes em desenvolvimento ágil distribuído, enfocando em como a segurança é abordada nesse contexto. Como resultado os autores apresentam uma extensão aos métodos ágeis, incluindo atividades de segurança. Uma pesquisa cujo escopo se delimita a perspectiva do profissional sobre os efeitos da inclusão de atividades de segurança em times ágeis é apresentada em (BARTSCH, 2011). Com o objetivo de entender como desenvolvedores e clientes interagem e utilizam atributos de segurança em seus projetos, o estudo relata as descobertas realizadas sobre a perspectiva de segurança em projetos, a partir das entrevistas realizadas com os profissionais envolvidos.

(Oyetoyan; Cruzes; Jaatun, 2016) apresenta um estudo empírico voltado para descobrir a relação entre treinamento, práticas e habilidade de segurança que possibilite o desenvolvimento de *software* seguro no mundo ágil. A partir de práticas extraídas dos processos MSDL, CC, TP e CLASP, a pesquisa mostra uma análise de dados extraídos de dois times de desenvolvimento ágil, com o intuito de entender como os pontos levantados estão relacionados, no ambiente dos times e com as práticas expostas. Os resultados da pesquisa são importantes para guiar deci-

sões de segurança em projetos de *software*, mas, segundo os autores, ainda existem problemas a serem estudados nessa área.

(BOSTRÖM et al., 2006) traz uma extensão do XP, incluindo atividades de segurança dentro do ciclo de vida do *framework* ágil, de modo a ter a minimizar o efeito delas na agilidade do processo. Neste estudo são apresentados 7 (sete) passos adicionais a serem incorporados no XP, a fim de cumprir os requisitos de segurança de projetos. Também utilizando XP como base, (DAUD, 2010) apresenta uma forma iterativa de lidar com segurança em projetos ágeis. O autor apresenta uma proposta de ciclo de vida de projetos de *software*, incluindo atividade de segurança. Para cada fase explicitada, a pesquisa descreve como essas atividades de segurança seriam executadas durante o processo.

Uma demonstração sobre como integrar requisitos de segurança, em processos ágeis, é apresentada em (SIPONEN; BASKERVILLE; KUIVALAINEN, 2005). Para tanto, são apresentados elementos chave de segurança, a serem incluídos durante o desenvolvimento ágil. Esses elementos são abordados quanto aos seus objetivos para cada fase do processo de desenvolvimento adotado na pesquisa. Para exemplificar o uso desses elementos, os autores apresentam, a incorporação da proposta no *framework* ágil FDD. (CHÓLIZ; VILAS; MOREIRA, 2015) contempla um estudo de caso que versa sobre os impactos da mudança de metodologias clássicas para metodologia ágeis de desenvolvimento. O caso, em questão, é analisado sob o aspecto da realização de teste de *software* voltados para segurança, demonstrando, principalmente, como esses testes passaram a serem executados após a mudança de metodologia e como decorreu a adequação desses dentro do novo cenário de desenvolvimento.

O *Protection Poker* é um método de priorização de atividades, similar ao *Planning Poker* já utilizado em times ágeis (WILLIAMS; MENEELY; SHIPLEY, 2010). Diferente deste último, o primeiro prioriza os itens, tendo como parâmetro o risco de segurança associado. Com uma forma de aplicação muito similar ao *Planning Poker*, este "*game*" visa estimular mais discussão e preocupação com segurança durante o projeto. Mas, por se tratar de apenas uma prática, cobre apenas uma parte do processo de desenvolvimento seguro. Apesar disso, sua compatibilidade natural com os métodos ágeis, faz dele uma técnica importante, para segurança no mundo ágil.

O *Security Backlog* é outra prática com propósito específico (AZHAM; GHANI; ITHNIN, 2011), sendo este incorporado como parte do processo de desenvolvimento utilizando Scrum. O trabalho expõe como esta atividade seria integrada dentro do

ciclo de vida de *software*, de acordo com as preocupações relativas aos princípios básicos de segurança levantados pelos autores. Abordando de forma objetiva a atividade proposta, a pesquisa traz uma visão concisa do *Security Backlog*, porém, ataca o problema em apenas uma das partes do processo.

A partir de uma base de dados de vulnerabilidades chamada *Common Vulnerabilities and Exposures* (CVE), (WANG; GUPTA; NIU, 2018) introduz uma metodologia para extração de requisitos de segurança a partir das falhas listadas na base. Esta pesquisa descreve os benefícios da utilização de tal prática no mundo ágil, seja no uso de casos reais ou como material de apoio durante a fase de levantamento de requisitos. Apesar de apresentar um método mais crível de inclusão de segurança, o trabalho só aborda apenas uma das fases do desenvolvimento ágil.

(BEZNOSOV; KRUCHTEN, 2004) busca demonstrar se práticas relacionadas a inclusão de segurança em projetos de *software* são compatíveis com metodologias ágeis. Uma série de atividades de segurança propostas foram categorizadas em atividades adequadas, independentes, semi-automatizadas e inadequadas. Após categorizar as atividades a pesquisa apresenta estratégias para adequar as atividades semi-automatizadas e inadequadas, uma vez que as outras duas, supostamente, não necessitam de nenhum outro mecanismo para serem aplicadas.

Como visto, diversas pesquisa abordam a inclusão de atividade de segurança em metodologias ágeis. Muitas dessas, argumentam que muito ainda precisa ser estudado, a fim de encontrar um equilíbrio entre segurança e agilidade. Partindo desta necessidade, este trabalho pretende explorar as práticas expostas nesses artigos, selecionando as mais importantes para compor políticas que visem demonstrar de forma geral, como, quando e quem deverá aplicá-las no processo ágil. Além disso, este trabalho irá apresentar os principais desafios e vantagens do uso dessas políticas, dando ao público uma visão mais unificado dos problemas e possíveis soluções, de forma simples e objetiva.

3 METODOLOGIA DE PESQUISA

O presente trabalho tem o intuito de propor políticas de desenvolvimento seguras por meio do uso de práticas para times ágeis, com isso em mente, optou-se por aplicar uma abordagem de pesquisa exploratória (KOTHARI, 2004). Para tanto, se fez necessário, realizar um levantamento bibliográfico, a fim de se obter uma compreensão inicial do problema e das principais abordagens utilizadas para tratá-lo. Com base no levantamento de dados bibliográfico feito a partir de ampla revisão da literatura da área foi possível explorar o campo estudado e entender o contexto da temática. Em relação ao processo de investigação e validação da solução, este trabalho segue uma metodologia qualitativa. Segundo (DENZIN; LINCOLN, 2006), a pesquisa qualitativa envolve uma abordagem interpretativa do mundo, o que significa que seus pesquisadores estudam as coisas em seus cenários naturais, tentando entender os fenômenos em termos dos significados que as pessoas a eles conferem. Seguindo essa linha de raciocínio, (VIEIRA; ZOUAIN, 2005) afirmam que a pesquisa qualitativa atribui importância fundamental aos depoimentos dos atores sociais envolvidos, aos discursos e aos significados transmitidos. Sendo assim, pela natureza dos dados coletados, utilizou-se de métodos qualitativos para obter e interpretar estes. A figura 1, apresenta uma ilustração do fluxo das etapas de pesquisa, descritas neste capitulo.

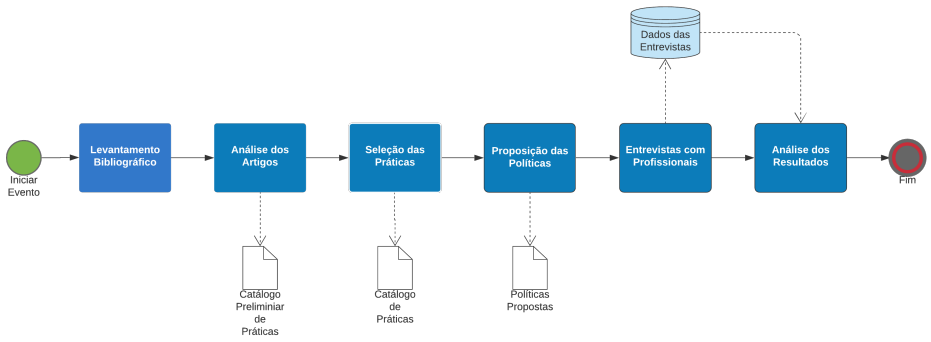

Figura 1 – Fluxograma das etapas de pesquisa

Fonte – O Autor

Engenho de Busca	**URL**
IEEE Xplore Digital Library	<https://ieeexplore.ieee.org/>
Springer Link	<https://link.springer.com/>
Science Direct	<https://www.sciencedirect.com/>
Google Scholar	<https://scholar.google.com/>

Tabela 2 – Engenhos de busca.

Fonte – O Autor

3.1 LEVANTAMENTO BIBLIOGRÁFICO

Foi realizada uma revisão exploratória da literatura, considerando artigos que apresentavam práticas para a inclusão de atividades de segurança em projetos de software. Mesmo com cunho exploratório, buscou-se utilizar de práticas de pesquisa que permitam a replicação do presente estudo. Os motores de busca utilizados para encontrar fontes relevantes estão discriminados na Tabela 2. Foram aplicadas restrições de ano 2010-2020. As buscas foram realizadas utilizando os seguintes termos de busca: (”Security”AND ”Development”AND (”Lifecycle”OR ”Process”) AND (”Practices”OR ”Policies”OR ”Challenges”OR ”Factors”)) e (”Security”AND ”Development”AND “Agile” AND (”Lifecycle”OR ”Process”) AND (”Practices”OR ”Policies”OR ”Challenges”OR ”Factors”)).

3.2 ANÁLISE DOS ARTIGOS

Na primeira fase de análise das obras encontradas a partir da aplicação das *strings* de busca citada na subseção anterior, foi realizada uma filtragem por título e resumos dos artigos. Essa filtragem visa avaliar a coerência dos artigos encontrados em relação ao tema de pesquisa abordado. Em um segundo momento, as obras filtradas pela primeira análise, foram avaliadas com base na presença de práticas de desenvolvimento seguro. Após a avaliação desses artigos, foi realizado um mapeamento de práticas neles abordadas, resultando em um total de 100 práticas, apresentadas no Apêndice 1. Nesse catalogo inicial, foi possível perceber práticas duplicadas ou com o mesmo proposito. A partir da informações extraídas de suas fontes, foram removidas duplicadas e agrupadas similares, reduzindo à 43 (quarenta e três), apresentadas no Apêndice 2.

3.3 SELEÇÃO DAS PRÁTICAS

Com todas as práticas levantadas, foi realizada uma preleção, para reduzir o escopo e viabilizar a proposição e validação das políticas, junto a especialistas. Para realizar esta seleção foram determinados alguns critérios para inclusão das práticas, sendo estes.

a) Quantidade de citações na literatura consultada, um mínimo de 3 referências foi estabelecido para que a prática pudesse ser incluída.

b) Relação com metodologias de desenvolvimento ágeis, práticas diretamente ligada às metodologias clássicas foram removidas.

c) Avaliação por parte de desenvolvedores de software experientes em segurança, do escopo resultante de práticas a partir dos passos “a” e “b”.

Os passos “a” e “b” resultaram na redução de 43 (quarenta e três) práticas, para um total de 21 (vinte e uma). Com o objetivo de maximizar o impacto das políticas propostas neste trabalho, o passo ”c”foi executado. A avaliação se deu por meio da apresentação dessas práticas, a um grupo de desenvolvedores de software. Para execução desse passo, foi selecionado um time de desenvolvimento ágil, de uma empresa especializada em segurança, do porto digital de Recife-PE. Optou-se por esse time, pois os desenvolvedores, possuíam experiencia tanto em desenvolvimento ágil, quanto, em aspectos de segurança de *software*. Esse processo buscou responder os seguintes questionamentos:

1. Essas práticas podem ser relacionadas com metodologias ágeis?
2. É possível agrupar uma ou mais dessas práticas?
3. Qual a importância da aplicação da prática em relação ao desenvolvimento de software seguro?
4. Em qual fase do desenvolvimento estas práticas deve estar incorporada?

A partir do resultado dessa primeira avaliação, exposto no Apêndice 3, foram priorizadas 14 práticas, a serem incorporadas às políticas aqui propostas. As demais práticas, foram removidas do escopo do trabalho, estando expostas no Apêndice 4, junto ao motivo da remoção.

3.4 PROPOSIÇÃO DAS POLÍTICAS

Com 14 (catorze) práticas, pré selecionadas na etapa anterior, foi estabelecida uma proposta das políticas. Estas foram definidas levando em consideração o processo de desenvolvimento, ao qual, cada uma das práticas pertencia. Esta informação foi extraída, tanto dos artigos analisados, quanto, da resposta do último questionamento feito pela validação do passo "c", da etapa anterior.

3.5 ENTREVISTAS COM PROFISSIONAIS

Com a proposição das políticas definidas, uma última etapa de validação foi iniciada. Nela foram apresentadas as políticas e suas práticas de modo a confirmar sua efetividade, quanto a ganhos de segurança, em projetos de software de times ágeis. Para poder realizar essa validação com coerência, três perfis profissionais foram escolhidos: Especialistas em segurança, desenvolvedores de times ágeis com experiência em segurança e especialistas em métodos ágeis. Cada uma das políticas foi apresentada com o intuito de responder os seguintes questionamentos.

1. A política faz sentido no contexto desenvolvimento seguro para times ágeis?
2. As práticas agregadas fazem sentido em relação à descrição e objetivos da política?
3. A partir de sua experiência, seria possível aplicar essa política em times ágeis? Se sim, como?
4. Você vê alguma restrição ou desafio quanto a aplicação dessa política? Se sim, quais?
5. Qual o impacto dessa política e suas práticas, em relação ao processo de desenvolvimento de *software* seguro?
6. Alguma outra prática poderia ser incluída nessa política para torná-la mais completa?

3.6 ANÁLISE DOS RESULTADOS

Finalmente, com a definição das políticas e resultados das entrevistas, foi realizada uma última análise, visando finalizar a proposta das políticas, objetivada por essa pesquisa. A partir dessa análise, foi realizada uma revisão, removendo ou

alterando práticas para refletir as opiniões coletadas durante as entrevistas. Além disso, uma última inspeção das práticas excluídas nas etapas anteriores foi realizada, com o intuito confirmar que nenhumas delas deveria ser resgatada e incluída nas políticas propostas.

3.7 FECHAMENTO DO CAPÍTULO

Neste capítulo apresentamos a metodologia de pesquisa utilizada pelo presente trabalho. Foram descritos os passos para realizar o levantamento bibliográfico e analisar as obras que vieram a servir como base para o desenvolvimento deste trabalho. O processo de avaliação das práticas e formulação das políticas foi definido. Finalmente, foi explicitado o método utilizado para finalizar a proposta das políticas dessa pesquisa.

4 POLÍTICAS DE DESENVOLVIMENTO SEGURO

Este capítulo apresenta as 5(cinco) políticas propostas, agregando as 14 (catorze) práticas selecionadas, com intuito de incluir segurança no processo ágil de desenvolvimento de *software*. A figura 2 apresenta um *mind map* ilustrando as políticas e suas respectivas práticas.

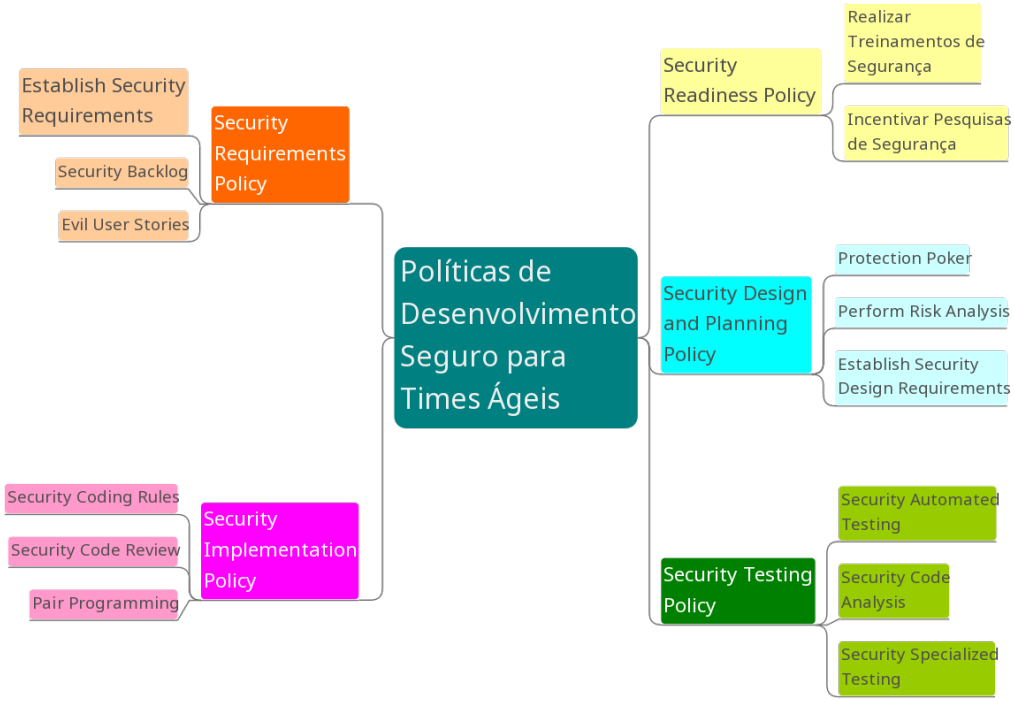

Figura 2 – Mind Map das políticas propostas e suas respectivas práticas

Fonte – O Autor

4.1 ESTRUTURA DAS POLÍTICAS

Estrutura de apresentação das políticas

1. Nome da Política;
2. Descrição da política de um modo geral, quanto às suas características e objetivos;

3. Lista de práticas a serem abordadas de modo a cumprir os objetivos dessa política;
    - Nome da Prática;
    - Descrição da prática de um modo geral, quanto às suas características e objetivos;
    - Principais metodologias para aplicar a prática dentro do processo de desenvolvimento;
    - Momentos do ciclo de desenvolvimento onde a prática deve ser aplicada;
    - Participantes do projeto devem estar envolvidos com essa prática;
    - Restrições no uso da prática, que a tornam inválida ou causam problemas futuros;
    - Principais citações dentro da literatura consultada;
4. Principais desafios encontrados ao aplicar a política em times ágeis;
5. Impactos da utilização dessa política, na segurança do projeto;

A figura 3 ilustra a estrutura de uma política, como apresentado acima.

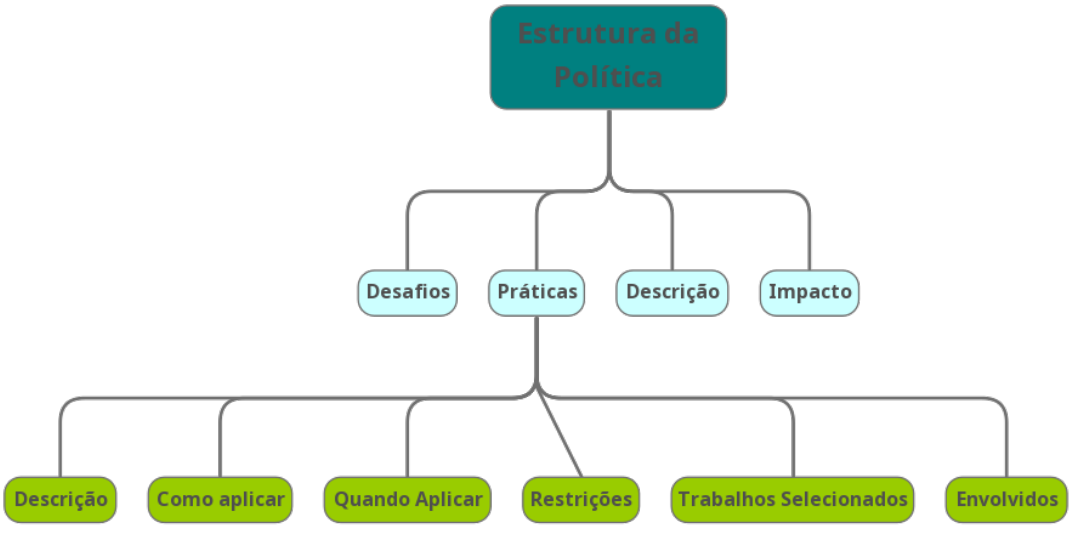

Figura 3 – Estrutura do conteúdo das políticas

Fonte – O Autor

4.2 SECURITY READINESS POLICY

4.2.1 Descrição

Ter um time de desenvolvimento, apto a lidar com aspectos de segurança de *software*, é primordial para entregar um produto que atenda aos requisitos dessa área, principalmente, em se tratando de times ágeis (Oueslati; Rahman; Othmane, 2015). Essa política propõe as principais práticas do processo de desenvolvimento de *software*, com o intuito de conscientizar e orientar times ágeis, acerca das características relevantes no concernente a segurança da informação. Apesar do enfoque no time de desenvolvimento, se possível, essa política deve ser estendida, a todos os envolvidos no processo de desenvolvimento (Clientes, *Stakeholders* e outros). A seguir descrevemos as práticas agregadas e suas principais características. São elas: Realizar Treinamentos de Segurança e Incentivar Pesquisas em Segurança.

4.2.2 Práticas

1. Provide Security Training

    a) Descrição

    Essa prática pretende preparar, educar e capacitar as partes interessadas no projeto sobre a importância da segurança. Durante os treinamentos, é essencial abordar tópicos como política de privacidade, conceitos básicos, grandes ataques, estratégias de defesa e modelagem de ameaças.

    b) Como Aplicar

    Existem diversas formas de aplicar treinamentos voltados para segurança, dentre elas podemos citar: contratar especialistas para promover esses treinamentos dentro do ambiente do projeto; adquirir plataformas de treinamentos que abordem o tema de segurança sob diferentes pontos de vista, de modo que o treinamento seja feito, quando surge a necessidade; incentivar treinamentos externos, feitos de forma individual ou de forma conjunta pelos envolvidos no projeto; Estimular o uso das diversas formas de treinamentos on-line, os quais muitas vezes são gratuitos, e das múltiplas fontes de materiais relacionados ao tema, tais como livros, artigos, blogs, postagens, considerados como sendo de boa qualidade.

    c) Quando Aplicar

    O melhor momento para introduzir o treinamento é no início do projeto, podendo, este, ser reforçado periodicamente, a fim de contribuir com a

excelência e conscientização técnica (Barbosa; Sampaio, 2015). Porém, essa abordagem não é mandatória, é possível incluir essa prática durante o ciclo do projeto, para abordar especificidades do mesmo ou mesmo atualizar membros do time. Além disso, sempre que novos profissionais são incluídos nos times, estes devem ser, minimamente, treinados para se adequarem aos princípios básicos de segurança definidos.

d) Envolvidos

Todos os interessados no projeto e ativamente participantes nos processos de desenvolvimento, de acordo com as necessidades específicas de cada papel.

e) Restrições

Uma vez tendo estabelecido os treinamentos pertinentes para os envolvidos, dentro da disponibilidade de recursos do projeto ou instituição, não existem restrições específicas para aplicação dessa prática.

f) Trabalhos Relacionados

(Oueslati; Rahman; Othmane, 2015), (KANNIAH; MAHRIN, 2016), (MICROSOFT..., 2019), (Sodanil et al., 2015), (Villamizar et al., 2018), (NICOLAYSEN et al., 2010), (Barbosa; Sampaio, 2015), (Oyetoyan; Cruzes; Jaatun, 2016), (Bansal; Jolly, 2014), (SINGHAL et al., 2012), (BARTSCH, 2011), (CLASP, 2019)

2. Incentive Security Research

a) Descrição

Estabelecer e promover práticas de incentivo a pesquisas relacionadas à segurança. Muitas vulnerabilidades são descobertas ao longo do tempo, por isso, para se proteger, é necessário investigar possíveis vulnerabilidades associadas ao projeto para implementar contramedidas. A execução desta pesquisa chama a atenção do time para novas formas de ataques de segurança, vulnerabilidades já reportadas e formas de proteção contra essas ameaças. É importante sempre manter o time atualizado quanto a esses aspectos, de modo que ele possa agir de forma mais eficiente diante de prováveis ameaças.

b) Como Aplicar

As pesquisas em segurança podem acontecer de diversas maneiras, tais como: busca por novas técnicas e conceitos em portais de divulgação desses conteúdos, como os da OWASP e *Microsoft Technical Blog*; assinar

canais de atualização de linguagens, ferramentas, sistemas operacionais e outros (*Microsoft Security Response Center, Golang Blog*); realizar fóruns de discussão de segurança em redes sociais (Linkedin) ou plataformas específica; explorar sites como CVE Details, Exploits Database que apresentam uma coleção atualizada de falhas de segurança; participar de eventos (H2HC, Eco Party, Black Hat) com foco em segurança, seja como palestrante ou ouvinte. Além dessas, podemos citar outras formas de pesquisar sobre segurança, sendo que, o mais importante é incentivar essa prática, seja por incentivos financeiros ou visibilidade dos profissionais que aderem as atividades propostas.

c) Quando Aplicar

Não existe um momento específico para desenvolver pesquisa de segurança. É possível determinar períodos nos quais indivíduos realizem trabalho de pesquisa, sem que, para tanto, os ciclos do projeto sejam afetados. Outro momento de pesquisa ocorre quando surgem necessidades específicas, em uma das fases do projeto, que precisem ser atendidas de imediato. Além disso, muitas das abordagens apontadas acima podem fazer parte do cotidiano dos profissionais.

d) Envolvidos

Todos os interessados no projeto e aqueles participantes ativos no processo de desenvolvimento, sempre em acordo com os assuntos de interesse e cada papel desempenhado.

e) Restrições

Tendo definido como incentivar esse processo de pesquisa, dentro da disponibilidade de recurso do projeto ou instituição, não existem restrições específicas para essa prática.

f) Trabalhos Relacionados

(Oueslati; Rahman; Othmane, 2015), (Barbosa; Sampaio, 2015), (Villamizar et al., 2018), (WANG; GUPTA; NIU, 2018), (FRANQUEIRA et al., 2011)

4.2.3 Desafios

A falta de capacitação e conscientização do time e do cliente, quanto a segurança, é apontado como um desafio no processo de desenvolvimento, não sendo, necessariamente, causado pela utilização de práticas ágeis (Oueslati; Rahman; Othmane, 2015). Prover treinamento para o desenvolvedores pode acrescer o custo de

desenvolvimento. Desse modo, manter a equipe treinada implica em custo repassado para o cliente, o que, muitas vezes, pode não estar disposto a pagar por isso (Oueslati; Rahman; Othmane, 2015), (KANNIAH; MAHRIN, 2016) e (Villamizar et al., 2018).

4.2.4 Impacto

Prover treinamento a todos os envolvidos no projetos, ajuda a prepará-los para lidar efetivamente com os requisitos de segurança (Barbosa; Sampaio, 2015), (KANNIAH; MAHRIN, 2016), (Sodanil et al., 2015) e (Villamizar et al., 2018) . Essa prática também contribui para que os envolvidos tomem consciência da importância da segurança nos projetos e suas consequências. Isso ajuda a derrubar uma mentalidade voltada para evitar assumir responsabilidade sobre o problema. Segundo (Barbosa; Sampaio, 2015) e (KANNIAH; MAHRIN, 2016), manter o time constantemente atualizado através de incentivos a pesquisa, contribui para a detecção e resolução mais eficiente de problemas relacionados a segurança. Além disso esses incentivos podem manter o time mais motivado.

Manter procedimentos de segurança ajuda a padronizar resolução de problemas e ajudar novos membro entrar no projeto mais facilmente como levantado em (KANNIAH; MAHRIN, 2016). Além dessas contribuições aplicar esta política pode aumentar a quantidade de especialistas de segurança no time, principalmente se utilizado junto a técnicas de transferências de conhecimentos como a demonstrada em (Sodanil et al., 2015). Com isso podemos poupar membros com expertise em segurança, de uma sobrecarga de tarefas, preocupação essa apontada por (KANNIAH; MAHRIN, 2016). Essa política também tem um impacto positivo quando partirmos para o mundo ágil, uma vez que este propõe o foco nas pessoas como valor e uma busca constante por excelência técnica como princípio (MANIFESTO..., 2001).

4.3 SECURITY REQUIREMENTS POLICY

4.3.1 Descrição

Levantar requisitos é uma tarefa de suma importância para qualquer projeto de *software*. Geralmente, no processo de times ágeis, o levantamento desses requisitos é feito no início de cada iteração, junto ao cliente, de forma a descobrir quais serão implementados durante aquele ciclo. Normalmente, apenas requisitos apontados pelo cliente são contemplados. Dessa forma, requisitos não funcionais, como segurança, performance e usabilidade, podem acabar sendo deixados de lado ou não priorizados

(Barbosa; Sampaio, 2015), (Villamizar et al., 2018), (Terpstra; Daneva; Wang, 2017), (NICOLAYSEN et al., 2010). Apesar de ser considerado um requisito não funcional, segurança vem se tornando, cada vez mais, um atributo essencial de *software*, dado o panorama de ameaças presentes nos mais diversos ambientes de implantação desses sistemas e a busca por confidencialidade de dados (BODDEN, 2018). Definir requisitos de segurança, de forma clara, objetiva e coerente, é muito importante para desenvolver um *software* seguro (KANNIAH; MAHRIN, 2016), (WANG; GUPTA; NIU, 2018). Com o intuito de mitigar problemas relacionados a falhas de segurança, causados pela falta de evidenciação desses requisitos, essa política propõe práticas, que incluam requisitos de segurança como parte do processo de times ágeis. Esta tarefa, é essencial desde o início do projeto, sendo perpetuada durante as iterações até a conclusão do *software* (NICOLAYSEN et al., 2010). As práticas incluídas nessa política são: *Define Security Requirements*, *Security Backlog* e *Evil User Stories*, as quais são apresentadas com mais riqueza de detalhes, a seguir.

4.3.2 Práticas

1. Define Security Requirements

    a) Descrição

    Esta prática visa identificar funcionalidades relacionadas a segurança, expressas de forma explícita, dentro de um projeto de *software* (BACA; CARLSSON, 2011), (COMMON..., 2019). É necessário garantir que requisitos de segurança sejam discutidos pelo time de desenvolvimento, junto ao cliente e a outros *stakeholders*. De modo que, estes requisitos não sejam postos de lado, devido ao enfoque em requisitos funcionais, por isso a necessidade de explicitamente considerá-los a fim de que sejam inclusos no esforço necessário para desenvolver a aplicação (CLASP, 2019), (KANNIAH; MAHRIN, 2016), (BOWEN et al., 2014), (SIPONEN; BASKERVILLE; KUIVALAINEN, 2005). É válido salientar que existem tanto requisitos específicos para segurança, quanto requisitos funcionais que devem expor características de segurança, como parte de sua definição (BACA; CARLSSON, 2011), (SEVEN..., 2019), (MCGRAW, 2004) e ambos devem devem ser levados em consideração.

    b) Como Aplicar

    Assim como os requisitos funcionais, os requisitos de segurança devem ser levantados junto ao cliente e demais *stakeholders*. Da mesma forma que o time ajuda o cliente a entender e definir requisitos funcionais do

projeto, ele também deve o fazer para atributos de segurança. Algumas abordagens apontam para definição de requisitos de segurança, via bases de vulnerabilidades já conhecidas, apontando, principalmente, requisitos que o cliente pode não estar ciente de existência ou importância (WANG; GUPTA; NIU, 2018). Para ter sucesso quanto a definição de requisitos de segurança é necessário garantir que estes se adaptem a mudanças dos requisitos do projetos (OTHMANE; ANGIN; BHARGAVA, 2014), (SIPONEN; BASKERVILLE; KUIVALAINEN, 2005). Principalmente, em relação a times ágeis que consideram essas mudanças parte necessária ao projeto e, por conseguinte, a entrega de valor ao cliente (MANIFESTO..., 2001). Nem todos os requisitos de segurança precisam estar incorporados dentro do desenvolvimento, em alguns casos, é possível assumir os riscos associados a não implementação destes (FRANQUEIRA et al., 2011). Mas, ainda assim, precisamos expor os mesmos, para analisar a possibilidade de assumir tais riscos.

c) Quando Aplicar

Estes requisitos devem ser atualizados e revistos a cada ciclo para refletirem as mudanças que ocorrem naturalmente no ciclo de projetos ágeis e também considerarem mudanças nos cenários de ameaça (MICROSOFT..., 2019), (BOWEN et al., 2014), (SIPONEN; BASKERVILLE; KUIVALAINEN, 2005), (DAUD, 2010).

d) Envolvidos

O principais envolvidos nesse processo são o time de desenvolvimento e o cliente. Porém, não se exclui a participação de outros envolvidos que possam contribuir para uma maior completude dos requisitos levantados.

e) Restrições

Existem restrições relativas ao tempo e custo do projeto, nem todos os requisitos serão levantados em virtude da limitação desses recursos. Este tipo de restrição deve ser acordada com o cliente, considerando prazos, objetivos e riscos assumidos pela não execução dessa tarefa de forma adequada.

f) Trabalhos Relacionados

(Barbosa; Sampaio, 2015), (KANNIAH; MAHRIN, 2016), (CLASP, 2019), (MICROSOFT..., 2019), (Sodanil et al., 2015), (BACA; CARLSSON, 2011), (SEVEN..., 2019), (COMMON..., 2019), (Villamizar et al., 2018), (Terpstra; Daneva; Wang, 2017), (NICOLAYSEN et al., 2010), (Oyetoyan; Cru-

zes; Jaatun, 2016), (Bansal; Jolly, 2014), (BOSTRÖM et al., 2006), (BOWEN et al., 2014), (SIPONEN; BASKERVILLE; KUIVALAINEN, 2005), (SINGHAL et al., 2012), (WANG; GUPTA; NIU, 2018), (BARTSCH, 2011), (MUNETOH; YOSHIOKA, 2013), (DAUD, 2010), (MCGRAW, 2004), (FRANQUEIRA et al., 2011), (OTHMANE; ANGIN; BHARGAVA, 2014)

2. Security Backlog

   a) Descrição

   Após o levantamento dos requisitos de segurança do projeto, pode ser criado um *Security Backlog*, que deve conter a lista de atividades relativas aos itens de segurança apontados (Barbosa; Sampaio, 2015), (NICOLAYSEN et al., 2010). Esse *Backlog* não precisa estar, necessariamente, separado do *Product Backlog*, porém, existem algumas vantagens nessa abordagem, como, por exemplo, manter um item de segurança associado a mais de um item do *backlog* de requisitos funcionais (NICOLAYSEN et al., 2010).

   b) Como Aplicar

   O *Security Backlog* segue os mesmo modos de operação do *Product Backlog*, já bastante difundido em times ágeis. A decisão de separar ou não as atividades de segurança deve ser um decisão de *Product Owner* e do time, baseada em critérios estabelecidos a partir de especificidades do projeto ou do time.

   c) Quando Aplicar

   O *Security Backlog* deve ser levantado e atualizado a cada ciclo de definição de requisitos do projeto. Contudo, nada impede que no decorrer de outras fases do projeto novos itens sejam adicionados na lista, para serem discutidos, priorizados e implementados nos ciclos seguintes.

   d) Envolvidos

   O *Product Owner* deve ser responsável por cuidar do *Security Backlog*, assim como, o faz com o *Product Backlog*. Também podemos apontar a presença de um *Security Master* que ficaria com essa responsabilidade (Barbosa; Sampaio, 2015), (Terpstra; Daneva; Wang, 2017), (AZHAM; GHANI; ITHNIN, 2011). Porém, nem sempre é possível dispor desse papel no time de desenvolvimento. Além disso, todo o time pode contribuir para essa lista de itens de segurança.

   e) Restrições

Não existe nenhuma restrição específica aplicável a essa prática.

f) Trabalhos Relacionados

(Barbosa; Sampaio, 2015), (Villamizar et al., 2018), (NICOLAYSEN et al., 2010), (AZHAM; GHANI; ITHNIN, 2011)

3. Evil User Stories

a) Descrição

Evil User Stories descrevem cenários de ameaças de segurança ao sistema, demonstrando como o sistema pode ser afetado por atacantes (BACA; CARLSSON, 2011). O objetivo principal é estressar as atividades do *Product Backlog* e do *Security Backlog*, de modo a detectar possíveis vulnerabilidades e fraquezas do sistemas que podem ser exploradas por usuários mal intencionados (Barbosa; Sampaio, 2015), (BACA; CARLSSON, 2011), (SEVEN..., 2019).

b) Como Aplicar

As *Evil User Stories* devem ser documentadas, de modo bem parecido as *User Stories* (Barbosa; Sampaio, 2015). Mas, é necessário que quem as escrever tenha um bom conhecimento em segurança para chegar o mais próximo possível de cenários reais de ataque.

c) Quando Aplicar

Esses artefatos devem ser definidos em paralelo aos itens do *Product Backlog* e do *Security Backlog*, durante os ciclos iterativos do projeto.

d) Envolvidos

Por haver a necessidade ter conhecimento em segurança para a escrita efetiva desses artefatos, podemos ligar essas atividades ao papel do *Security Master* (Barbosa; Sampaio, 2015). Porém, caso este papel não esteja disponível no time, essa responsabilidade pode ser passada ao membro com mais experiência na área de segurança. Mas, é importante que outros membros do time e *stakeholders* estejam envolvidos nessa prática para garantir que múltiplos pontos de vista sejam explorados (Barbosa; Sampaio, 2015).

e) Restrições

A necessidade de um profissional com experiência em segurança é tida como uma restrição para essa prática. Na maioria das vezes times ágeis não possuem esse perfil técnico a disposição. Tempo para performar essas

atividades é outra restrição aplicável, essa tarefa não é simples e pode não ser eficiente se realizada de maneira inapropriada.

f) Trabalhos Relacionados

(Barbosa; Sampaio, 2015), (Villamizar et al., 2018), (NICOLAYSEN et al., 2010), (BOSTRÖM et al., 2006), (BOWEN et al., 2014), (CHÓLIZ; VILAS; MOREIRA, 2015), (SIPONEN; BASKERVILLE; KUIVALAINEN, 2005), (BARTSCH, 2011), (WILLIAMS; MENEELY; SHIPLEY, 2010), (MUNETOH; YOSHIOKA, 2013), (DAUD, 2010), (OTHMANE; ANGIN; BHARGAVA, 2014).

4.3.3 Desafios

Quando falamos em desenvolvimento ágil, mudanças nos requisitos, sempre, são uma constante durante os ciclos do projeto. Desse ponto de vista adicionar segurança pode ser desafiador, mudanças em requisitos funcionais induzem mudanças nos requisitos de segurança, como consequência (Oueslati; Rahman; Othmane, 2015), (SIPONEN; BASKERVILLE; KUIVALAINEN, 2005), (ESSAFI; LABED; GHEZALA, 2007). Toda mudança requer mais tempo de desenvolvimento, aumentando, consequentemente, os custos do projeto. Apesar das metodologias ágeis incluírem essa prática, como forma de entregar valor ao cliente, promover mais mudanças, pode levar o cliente a despriorizar os itens de segurança, em prol de atividades que na sua visão, agregam mais valor ao negócio (Oueslati; Rahman; Othmane, 2015), (Villamizar et al., 2018). Em adição a isso, duas novas atividades dentro da fase de requisitos, podem ter um impacto negativo na agilidade do time, consumindo tempo de iterações normalmente curtas (Oueslati; Rahman; Othmane, 2015), (Barbosa; Sampaio, 2015). Temos que considerar, também, que essas práticas exigem do time de desenvolvimento, uma certa familiaridade com segurança, para que sejam empregadas de modo efetivo (BOWEN et al., 2014).

4.3.4 Impacto

Implantar essa política de requisitos implica em mapear necessidades de segurança do projeto, de forma que existe uma diminuição na possibilidade de essas atividades serem deixadas de lado em proveito dos requisitos funcionais. Adotar práticas como *Security Backlog* e *Evil User Stories* apoiam o time ágil, mantendo as atividades de segurança mapeadas de forma similar a outras atividades. Desse modo, será possível afirmar, durante as entregas, quais mecanismos de segurança fo-

ram implementados e de forma que será possível manter coerência com a descrição de *software* seguro, a qual já foi apresentada neste trabalho.

4.4 SECURITY DESIGN AND PLANNING POLICY

4.4.1 Descrição

Problemas de segurança nem sempre estão associados a falta de requisitos de segurança, falhas de implementação ou falta cuidados relativos a segurança em geral, também podem gerar este tipo de problema. Muitas vezes esses problemas são inseridos no projeto por erros de design e planejamento, como, por exemplo, não avaliar riscos associados a requisitos funcionais ou não priorizar corretamente quais problemas devem ser implementados. Durante as fases de planejamento e design, deve haver um cuidado com design de arquitetura, planejamento de interfaces e acompanhamento do projeto, de modo a evitar a inserção de erros, no processo, que venham a se torna falhas de segurança. Além disso, é necessário monitorar as ações, de forma a entender quais os impactos causados no processo, auxiliando a tomada de decisões sobre os próximos passos do projeto. Essa política tem o objetivo de apresentar práticas envolvendo questões de design e planejamento que sejam voltadas para a implantação de mecanismos de segurança durante o processo. As práticas cobertas são: *Protection Poker*, *Risk Analysis* e *Security Design Requirements*.

4.4.2 Práticas

1. Protection Poker

    a) Descrição

    Essa prática baseada no *Planning Poker* tem o intuito de priorizar atividades de desenvolvimento de acordo com o risco de segurança a elas associado. O valor dado a cada atividade deve seguir a sequência de fibonacci. No *Protection Poker* quanto maior o valor dado, maior o risco de segurança associado (Barbosa; Sampaio, 2015). Essa prática é uma forma colaborativa para guiar a priorização e possibilitar aos envolvidos o aperfeiçoamento de seus conhecimentos em segurança, dado o tempo concedido para discutir e defender seus pontos de vista, sobre os riscos de segurança (WILLIAMS; MENEELY; SHIPLEY, 2010).

    b) Como Aplicar

Essa prática segue o mesmo princípio de aplicação do *Planning Poker*. Primeiro, a atividade a ser analisada, votada e explicada para todos os presentes. Em seguida, uma rodada de discussão ocorre, para que todos tenham espaço para expressar suas considerações e levantar questionamentos sobre a implementação da atividade e os riscos envolvidos. Em um terceiro momento, todos colocam a pontuação que acham coerente para atividade em uma "carta" que pode ser feita em um pedaço de papel ou em aplicativos específicos para esse tipo de atividade. Quando todos estão com seus votos preparados, é solicitado que todos o apresentem simultaneamente. Após a votação ocorrer, caso não haja um consenso nos votos apresentados, aqueles com os valores mais baixo e mais alto, respectivamente, apresentam suas considerações sobre seus votos. Em seguida tem-se uma nova votação, caso ainda não haja consenso, é possível realizar outras rodadas até chegar a um consenso ou findar o tempo hábil para discussão da atividade (WILLIAMS; MENEELY; SHIPLEY, 2010).

c) Quando Aplicar

O *Protection Poker* deve ser conduzido durante as sessões de planejamento do próximo ciclo de desenvolvido, incluindo as atividades selecionadas para implementação.

d) Envolvidos

O time de desenvolvimento do projeto.

e) Restrições

Não existem restrições específicas para aplicação dessa prática.

f) Trabalhos Relacionados

(Barbosa; Sampaio, 2015), (Villamizar et al., 2018), (BOWEN et al., 2014), (WILLIAMS; MENEELY; SHIPLEY, 2010)

2. Perform Risk Analysis

a) Descrição

Analistas de segurança analisam o risco de segurança para os requisitos do projeto, de modo que seja possível classificá-los de acordo com o nível de risco apresentado. Desconsiderar a análise de risco nas fases iniciais leva a problemas caros no futuro (BACA; CARLSSON, 2011). O risco, normalmente, deve ser determinado de aplicação para aplicação. O objetivo final, seria realizar o levantamento de todos os riscos. Isso é

capturado no Documento de Avaliação de Risco, que é refinado nas iterações subseqüentes (BACA; CARLSSON, 2011). Os riscos de segurança, ao contrário dos riscos de projeto, não são considerados durante todo o processo de desenvolvimento.

b) Como Aplicar

(FRANQUEIRA et al., 2011) propõem uma maneira de incorporar análise de risco em times ágeis. Durante as iterações do desenvolvimento, deve haver micro iterações de avaliação, tratamento e aceitação de riscos. Essas iterações curtas recebem insumo da aplicação de práticas de segurança e de bases de dados empíricos armazenados em catálogos públicos com CVE. Suas saídas alimentam a atualização de requisitos e funcionalidades de segurança e fornecem feedback para um processo de gerenciamento de segurança que coleta riscos residuais, gradualmente. Esses riscos residuais que não foram tratados, pelas mitigações introduzidas no sistema na iteração em execução, devem ser avaliados novamente na próxima iteração, e, assim, sucessivamente.

c) Quando Aplicar

Iterativamente durante todo o desenvolvimento do *software*. Porém, na fase inicial do projeto, vai ser demandada, do time, maior acuracidade nessa análise.

d) Envolvidos

Todos os membros do time de desenvolvimento. Tanto os que atuam analisando quanto aqueles atuam provendo insumos para a realização da atividade. O cliente também deve contribuir com seu ponto de vista sobre os riscos em relação às regras de negócio.

e) Restrições

A natureza da atividade é custosa, tanto em termos de tempo quanto de demanda de recursos financeiros. Salienta-se a necessidade de ter profissionais com experiência em segurança para realizar a atividade; estar baseado em documentação extensa do trabalho realizado; e definir critérios para todo o projeto. Estes são aspectos apontados por (FRANQUEIRA et al., 2011) como sendo pontos restritivos da aplicação no mundo ágil. Porém, a proposta apresentada neste mesmo estudo e descrita acima pode mitigar alguns desses pontos.

f) Trabalhos Relacionados

(BACA; CARLSSON, 2011), (SEVEN..., 2019), (COMMON..., 2019), (NICOLAYSEN et al., 2010), (Terpstra; Daneva; Wang, 2017), (FRANQUEIRA et al., 2011)

3. Establish Security Design Requirements

   a) Descrição

   Normalmente, o desenvolvimento seguro é aquele que inclui atividades que ajudam os engenheiros a implementar requisitos de modo seguro, ou seja, a medida que esses requisitos são bem projetados com relação à segurança o processo é considerado seguro. Para conseguir isso, os engenheiros, normalmente, contam com recursos de segurança, como criptografia, autenticação, log, entre outros. Em muitos casos, a seleção ou implementação de recursos de segurança se mostrou tão complicada que as escolhas de design ou implementação provavelmente resultaram em vulnerabilidades. Portanto, é extremamente importante que eles sejam aplicados de forma consistente e com um entendimento consistente da proteção que eles fornecem (MICROSOFT..., 2019).

   b) Como Aplicar

   Nas primeiras iterações uma arquitetura inicial deve ser definida. O arquiteto percorre a lista de requisitos iniciais do produto, para obter uma visão mais generalista do sistema e tenta descobrir o recurso de segurança arquitetural necessário. É importante que a arquitetura escolhida não imponha restrições a possíveis recursos de segurança que possam vir a ser necessários posteriormente. Todos os desenvolvedores devem avaliar a arquitetura inicial juntos e discutir como ela se comporta com os requisitos de segurança. Pode ser necessário discutir possíveis problemas de implementação. Discutir isso em plenário garante que todos os conhecimentos combinados sejam usados e que muitos problemas futuros possam ser evitados (NICOLAYSEN et al., 2010). O entrevistado 8 sugere, que além desses procedimentos, seja criado um documento simples, no qual sejam inclusas as decisões de arquitetura, comunicação, entre outros aspectos. Esta seria uma forma de manter o registro de quais, como e por que motivos determinadas decisões foram tomadas.

   c) Quando Aplicar

   Além de uma decisão generalista da arquitetura, durante as primeiras iterações, sempre que requisitos de *software* forem levantados, o time terá

que conversar sobre essas decisões, principalmente, aquelas que envolvam maior risco, conforme avaliação (NICOLAYSEN et al., 2010). Desse modo poderão ser encontradas soluções que cumpram o requisito e que sejam aplicáveis ao estado atual do *software*, antes que este seja de fato selecionado para implementação. Para que isso possa acontecer, é necessário um cuidado extra, em relação a essas decisões nos primeiros ciclos do projeto. Em (NICOLAYSEN et al., 2010), vemos um caso em que as decisões desse tipo foram realizadas em paralelo com a implementação, causando como efeito o fato de que esse processo deixou de ser ágil, além de fazer com que muitas decisões de segurança fossem deixadas de lado.

d) Envolvidos

Os membros do time de desenvolvimento, incluindo arquitetos de *software*, designers e desenvolvedores.

e) Restrições

A experiência dos profissionais, tanto, em relação a segurança, quanto, a como decidir quais recursos disponíveis para alcançar os objetivos de segurança são considerados restrições para essa prática. O treinamento para realizar tal tarefa é levantado como um ponto essencial em (Oyetoyan; Cruzes; Jaatun, 2016).

f) Trabalhos Relacionados

(MICROSOFT..., 2019), (Sodanil et al., 2015), (BACA; CARLSSON, 2011), (Oyetoyan; Cruzes; Jaatun, 2016), (NICOLAYSEN et al., 2010), (BOSTRÖM et al., 2006)

4.4.3 Desafios

O principal desafio apontado são as restrições de tempo e custo impostas por algumas das práticas citadas acima, visto sua aplicação em times ágeis. Além disso, realizar treinamentos com o time todo, a fim de elevar o nível de expertise a um parâmetro suficiente para aplicação dessa política, também, pode ser visto como desafio.

4.4.4 Impacto

Como visto durante as práticas, tomar decisões sobre o planejamento e design do projeto contribui, sobretudo, para que problemas surjam tardiamente durante o desenvolvimento. Práticas como *Perform Risk Analysis* e *Protection Poker* incentivam membros do time a discutir sobre segurança, ainda durante o planejamento,

levantando falhas antes que estas ocorram. Além disso, essa atividades servem como insumo para negociar com o cliente quais requisitos de segurança devem ser implementados e para sinalizar os possíveis problemas causados pela não priorização destes.

4.5 SECURITY IMPLEMENTATION POLICY

4.5.1 Descrição

Normalmente, a fase de implementação está mais suscetível a inclusão de problemas de segurança, os quais podem ser, e muitas vezes são, incorporados de forma involuntária. Muitos desenvolvedores não estão cientes sobre a forma correta de implementar certas funcionalidades de modo a evitar falhas de segurança (KANNIAH; MAHRIN, 2016). Sendo assim, diversos problemas podem ser incluídos durante o processo. É virtualmente impossível evitar todos os erros de implementação, tanto os relacionados a requisitos funcionais, quanto os relacionados a segurança, sendo estes ainda mais desafiadores e mais improváveis de serem evitados em sua totalidade. Times ágeis já aplicam diversas modos de lidar com erros de programação, tais como, estabelecer padrões de desenvolvimento. Com o intuito de adicionar mecanismos que evitem problemas desse tipo, ou seja, relacionados a segurança, essa política propõe práticas que ajudem os desenvolvedores a evitar, identificar e corrigir falhas, ainda, na fase de implementação, são elas: *Security Coding Rules*, *Secure Code Review* e *Pair Programming*.

4.5.2 Práticas

1. Security Coding Rules

    a) Descrição

    Diversos times de desenvolvimento aplicam práticas de implementação como forma de manter a qualidade do código, sendo que, uma das mais difundidas é o Clean Code (MARTIN, 2009). De modo semelhante essas regras podem ser definidas em relação segurança. Durante a fase de implementação dos requisitos do *software* muitas vulnerabilidade podem se inseridas de modo não intencional. Com o intuito de evitar os erros mais comuns nesse processo, times de desenvolvimento devem estabelecer o conjunto de regras de desenvolvimento, com foco em boas práticas ligadas a segurança de *software*. Essa regras devem especificar alguns

pontos importantes, como o tratamento de entradas dos usuários ou que se evite o uso de funções obsoletas (NICOLAYSEN et al., 2010), (BACA; CARLSSON, 2011), (MICROSOFT..., 2019). É importante manter em mente que para alcançar objetivos de segurança para o projeto, devemos escrever um código manutenível e resiliente em relação a segurança, isso pode ser otimizado por meio do uso dessa prática (KANNIAH; MAHRIN, 2016), (CLASP, 2019).

b) Como Aplicar

O principal ponto para aplicação é a definição de quais regras os programadores de seguir para escrever um código seguro. Muitas dessas regras podem ser elencadas por meio de treinamentos e pesquisas, como apontado na Política de Readiness. Além disso, as linguagens de programação geralmente mantem guias para desenvolvimento seguro como (JAVA..., 2001), de onde podem ser extraidas essas regras. Outras práticas citadas na literatura também podem ajudar a definir esses padrões, como, por exemplo, a definição e o uso de padrões de criptografia bem como o uso de ferramentas aprovadas (MICROSOFT..., 2019), (Sodanil et al., 2015), (BACA; CARLSSON, 2011).

c) Quando Aplicar

Essas regras devem ser seguidas durante todo o processo de implementação do *software*, em especial, a itens com alto risco associado.

d) Envolvidos

Todo o time de desenvolvimento do projeto.

e) Restrições

Uma vez definidas, essas regras dependem, exclusivamente, do bom senso dos programadores em seguí-las. Pode não ser fácil acompanhar o uso dessas regras, porém alguns métodos, como análise estática de código e revisão de código, podem apresentar um feedback sobre como os desenvolvedores estão performando.

f) Trabalhos Relacionados

(KANNIAH; MAHRIN, 2016), (Sodanil et al., 2015), (BACA; CARLSSON, 2011), (NICOLAYSEN et al., 2010), (Oyetoyan; Cruzes; Jaatun, 2016), (Bansal; Jolly, 2014), (BOSTRÖM et al., 2006), (BARTSCH, 2011), (MUNETOH; YOSHIOKA, 2013), (DAUD, 2010), (CLASP, 2019)

2. Security Code Review

a) Descrição

Times de desenvolvimento, muitas vezes, usam *Code Review* como prática para estabelecer e manter padrões de qualidade de código. Está prática consiste na realização de uma revisão manual do código de desenvolvimento, sendo esta executada por outros membros do time, antes que esse código vá para os repositórios (BERNHART; MAUCZKA; GRECHENIG, 2010). Ferramentas de versionamento como o bitbucket da Atlassian incluem funcionalidade para tal prática (MAKING..., 2001). Encontrar e remediar erros durante a implementação é uma forma eficiente de evitar que essa falhas se perpetuem. E se incluirmos cuidados com segurança nesse processo também será possível evitar diversas vulnerabilidades nos *software* (BERNHART; MAUCZKA; GRECHENIG, 2010). *Security Code Review* é uma extensão dessa prática, a qual propõe que os desenvolvedores observem, também, possíveis problemas de segurança, durante a revisão do código. Os revisores analisam o código, e se pertinente adicionam comentários em trechos que podem levar a falhas de segurança. Essa técnica objetiva evitar que falhas conhecidas sejam perpetuadas durante a fase de implementação. Além disso, é possível que programadores com menos experiência em segurança atentem para essas falhas e aprendam sobre segurança durante as discussões geradas por comentários adicionadas durante a revisão.

b) Como Aplicar

Com o auxílio de ferramentas, como as que foram apresentadas, é possível disponibilizar o código desenvolvido para revisão e indicar os revisores desejados. Adicionar revisores que tenham experiência em segurança contribui para um melhor resultado. Após a revisão, o responsável pelo código se volta para os comentários e resolve os problemas apontados. Um fator importante, é atentar para o uso das *Secure Coding Rules*, elas ajudam, inclusive nos casos em que os desenvolvedores têm pouca experiência em segurança, na percepção de problemas e possibilita o acompanhamento do uso da regras. Tendo sido aprovado pelos revisores, o código é adicionado, de fato, ao repositório principal da aplicação. É preciso estabelecer, a partir das ferramentas, as regras para que o código seja incorporado no código da aplicação e o quantitativo mínimo de aprovações, para tanto. Utilizar essa prática pode aumentar a conscientização dos desenvolvedores em relação ao desenvolvimento seguro de código.

c) Quando Aplicar

Essa prática deve ser aplicada durante toda a fase de implementação de *software*.

d) Envolvidos

Todos os desenvolvedores responsáveis pela implementação do *software*.

e) Restrições

Não existem restrições quanto ao uso dessa prática. No entanto, a presença de profissionais, com expertise em segurança no time, permite que esta apresente um melhor desempenho, no concernente a evitar erros de segurança

f) Trabalhos Relacionados

(BACA; CARLSSON, 2011), (Oueslati; Rahman; Othmane, 2015), (Terpstra; Daneva; Wang, 2017), (Villamizar et al., 2018), (Oyetoyan; Cruzes; Jaatun, 2016), (CHÓLIZ; VILAS; MOREIRA, 2015)

3. Pair Programming

a) Descrição

Pair Programming é um conceito ágil que define a situação onde os desenvolvedores codificam em pares, resolvendo e revisando problemas de forma conjunta, enquanto o código é escrito (BACA; CARLSSON, 2011). Desenvolver dessa forma leva os desenvolvedores a compartilharem conhecimentos, enquanto estão trabalhando nos requisitos durante o desenvolvimento. Essa prática pode incentivar desenvolvedores com mais experiência em segurança a passarem seus aprendizados sobre a área para aqueles menos experientes, via *Pair Programming* em requisitos que envolvem atributos de segurança.

b) Como Aplicar

Para aplicar esta prática basta que dois membros do time desenvolvam uma atividade de implementação de modo cooperativo. O modo como essa cooperação vai se dar pode ser estabelecido pelos próprios desenvolvedores.

c) Quando Aplicar

Essa prática pode ser aplicada em qualquer momento do projeto na fase de implementação.

d) Envolvidos

Todos os membros do time de desenvolvedores.

e) Restrições

Tempo e tamanho do time são restrições para essa prática, uma vez que cada desenvolvedor pode trabalhar em cima de uma atividade, agilizando o processo de implementação. Além disso pode não haver membros disponíveis no time para aplicar esse tipo de abordagem.

f) Trabalhos Relacionados

(BACA; CARLSSON, 2011), (Oyetoyan; Cruzes; Jaatun, 2016), (BOWEN et al., 2014)

4.5.3 Desafios

As práticas aqui proposta, já são de certo modo, aplicadas dentro do processo de desenvolvimento ágil. No entanto, no contexto de segurança, a necessidade dos profissionais possuírem maior familiaridade com o tema, pode ser vista como um desafio. Além disso, monitorar se estas práticas estão sendo realmente aplicadas é um ponto crucial, porém, complexo de ser implantado. A dependência da conscientização dos próprios desenvolvedores pode se tornar um empecilho e precisa ser levada a sério.

4.5.4 Impacto

O uso das práticas apontadas evita que erros básicos de segurança sejam cometidos pelos desenvolvedores. Além disso, erros que muitas vezes passam despercebidos podem ser apontados durante as revisões de código, possibilitando sua correção ainda no processo de implementação. Vale a pena salientar que essa práticas podem moldar a mentalidade do profissional, de modo que este busque cada vez mais por qualidade no desenvolvimento, busca esta, apontada como sendo um dos princípios do manifesto ágil.

4.6 SECURITY TESTING POLICY

4.6.1 Descrição

Testar sempre foi um atributo essencial em qualquer projeto de *software*, muitos times de desenvolvimento veem o processo de teste de *software* como o principal mecanismo para evitar falhas no sistema. O processo de teste não só encontra esses erros no momento em que eles estão presentes, mas, também evita que eles voltem a acontecer em mudanças futuras no código fonte. Além disso, ter um sistema com

uma cobertura de testes aceitável, dá confiança ao desenvolvedor para realizar mudanças e/ou melhorias no código, atividade comum em times ágeis. Essa confiança é pautada em cima de uma certeza racional de que problemas inseridos durante o processo serão levantados durante a execução dos testes. Por mais que testes estejam presentes no processo de muitos times ágeis, geralmente, eles estão associados aos requisitos funcionais e muitas vezes testes de segurança passam despercebidos, tanto por não terem sido levantados esses requisitos quanto por uma falta de preocupação com o aspecto segurança. Para podermos trazer testes, como ferramenta de inclusão de segurança em projetos de *software*, essa política apresenta práticas relacionadas ao processo de testes. São elas: *Security Automated Testing*, *Security Code Analysis*, *Security Specialized Testing*.

4.6.2 Práticas

1. Security Automated Testing

    a) Descrição

    Testes automatizados são uma prática bastante implantada em diversos projetos de *software*. O intuito dessa práticas é implementar funções de verificação que comprovam a corretude dos requisitos implementados (NICOLAYSEN et al., 2010). Aproveitar a implementação desses testes e incluir validação de funcionalidades relativas a segurança pode aumentar a confiabilidade do *software*. Além da validação do código implementado, outro benefício do *software* é adicionar uma certeza racional, que quando mudanças ocorrerem, o teste deverá garantir que o código, já validado, continue funcionando como deveria. Deste ponto de vista, caso tenhamos verificações relacionadas a segurança, é possível afirmar que código desenvolvido para prover segurança não será desvirtuado em mudanças futuras (NICOLAYSEN et al., 2010). Além disso vulnerabilidades corrigidas e cobertas com novos casos de teste tem pequena chance de voltarem a acontecer, uma vez que temos uma garantia, que se algo diferente do padrão esperado acontecer, o desenvolvedor vai saber imediatamente após as execuções dos testes. Geralmente não é necessário escrever testes com alto padrão de qualidade de código, basta que esse cubra diversos cenários reais de uso do sistemas, tanto funcionais quanto de segurança.

    b) Como Aplicar

    A maioria das linguagens de programação, senão todas, possuem seus próprios módulos para teste automatizados, além disso, existem diver-

sas bibliotecas para esse propósito. Ao escrever teste automatizados para segurança deve-se assegurar de implementar a maior quantidade de cenários possível e que estes tenham a maior proximidade possível de cenários reais de uso. Além disso, é necessário cobrir a maior quantidade de requisitos possível, com casos de testes, incluindo, principalmente, itens dos *Security Backlog* e as *Evil User Stories* (NICOLAYSEN et al., 2010). Claro que testes automatizados não conseguem cobrir todos os pontos de segurança do sistema, mas, como já explicitado anteriormente, eles provêm a garantia de que comportamentos já previstos e *bugs* reportados não devem voltar a acontecer.

c) Quando Aplicar

A escrita de testes ocorre, principalmente, durante a fase de implementação do *software*. Durante essa fase o desenvolvedor escreve o código relativo aos requisitos, bem como os testes relativos ao código implementado (NICOLAYSEN et al., 2010). Existem práticas, como o TDD, que apontam para a escrita dos testes antes mesmo da implementação da atividade. Criar testes, antes do código da aplicação em si, permite que o desenvolvedor analise como deve desenvolver os requisitos, levantando Corner Cases e outros pontos de vista sobre a atividade (NICOLAYSEN et al., 2010).

d) Envolvidos

A equipe de desenvolvedores do time.

e) Restrições

Geralmente teste unitários são vistos como uma das atividades custosas, do ponto de vista de implementação, uma vez que são mais atividades para o desenvolvedor implementar. Logo, tempo e custo podem ser uma restrição para essa abordagem. Todavia, muitos benefícios são gerados por sistemas que incluem uma boa cobertura dos testes, valendo a pena a adição de custos ao desenvolvimento. É válido elucidar que esse custo tende a diminuir com o tempo (NICOLAYSEN et al., 2010).

f) Trabalhos Relacionados

(NICOLAYSEN et al., 2010), (SINGHAL et al., 2012), (BARTSCH, 2011), (MUNETOH; YOSHIOKA, 2013), (DAUD, 2010), (FRANQUEIRA et al., 2011)

2. Security Code Analysis

a) Descrição

O código fonte é um artefato produzido por todos os projetos de *software*. Garantir que esse código seja analisado em busca de falhas de segurança é essencial para validar a segurança do *software* implementado (BACA; CARLSSON, 2011), (SEVEN..., 2019). Essa prática consiste, basicamente, em três metodologias de verificação, quais sejam: *Static Code Analysis*, *Dynamic Code Analysis* e *Fuzzy Testing*. A primeira analisa o código puro, a partir de regras de verificação predefinidas, em busca de trechos de código que apresentem problemas de segurança apontados pelas regras adotadas. A Segunda analisa o código, em tempo de execução, juntamente como todo o código de bibliotecas e de terceiros incorporados ao código implementado. A terceira aplica diversos tipos de entradas predefinidas para o *software* e busca por padrões anômalos no resultado (BACA; CARLSSON, 2011), (MICROSOFT..., 2019). A partir dos resultados apresentados por essas abordagens, é possível aferir a conformidade do código com os padrões de segurança definidos nos requisitos do projeto.

b) Como Aplicar

Para todas as abordagens acima, existem ferramentas que fazem essas verificações, basta que elas sejam configuradas corretamente. A OWASP mantém uma lista de ferramentas com esse intuito, (VULNERABILITY..., 2001). Também podemos encontrar ferramentas específicas de linguagens, como: (GO..., 2001). O time de desenvolvimento deve encontrar essas ferramentas, de acordo com as suas necessidades, a depender da linguagem e dos processos utilizados. Com essas ferramentas, o time deve configurá-las e executá-las em cima da base de código do projeto. Os resultados devem ser analisados e, se necessário, devem ser adicionadas tarefas ao backlog para corrigir os problemas encontrados.

c) Quando Aplicar

A configuração das ferramentas de verificação de código devem ser realizada nas primeiras iterações do projeto (CHÓLIZ; VILAS; MOREIRA, 2015). Uma vez que elas estejam configuradas, o time deve encontrar o melhor momento para executá-las. Testes realizados por ferramentas podem ser executados em todas as iterações de desenvolvimento (CHÓLIZ; VILAS; MOREIRA, 2015). Além disso, times que trabalham com CI/CD podem adicionar a execução dessas ferramentas dentro dos seus respectivos pipelines de execução. Dessa forma, a cada integração do

software a base de código principal, uma rodada de verificação será realizada (PAULE; DÜLLMANN; HOORN, 2019). Executar essas ferramentas, pontualmente, também é uma abordagem que pode ser usada para validar a correção de determinados bugs.

d) Envolvidos

Todo o time de desenvolvimento do projeto.

e) Restrições

Ferramentas para realizar análise de código, são muitas vezes pagas, esse custo geralmente é elevado (KANNIAH; MAHRIN, 2016). Apesar de existirem ferramentas gratuitas com esse propósito, estas podem não performar tão bem quanto desejado. Existe um risco associado a decisão de usar um ferramenta paga ou gratuita, isso deve ser discutido em colaboração com o cliente. Além do custo, não existem outras restrições para aplicar esta prática.

f) Trabalhos Relacionados

(Barbosa; Sampaio, 2015), (MICROSOFT..., 2019), (Sodanil et al., 2015), (BACA; CARLSSON, 2011), (NICOLAYSEN et al., 2010), (Oueslati; Rahman; Othmane, 2015), (SEVEN..., 2019), (KANNIAH; MAHRIN, 2016), (Oyetoyan; Cruzes; Jaatun, 2016), (Bansal; Jolly, 2014), (BOSTRÖM et al., 2006), (CHÓLIZ; VILAS; MOREIRA, 2015), (SINGHAL et al., 2012), (MUNETOH; YOSHIOKA, 2013), (MCGRAW, 2004), (BODDEN, 2018), (FRANQUEIRA et al., 2011), (KANNIAH; MAHRIN, 2016), (CLASP, 2019)

3. Security Specialized Testing

a) Descrição

Naturalmente, por mais que os desenvolvedores busquem por implementar o *software* com as melhores práticas de segurança, erros continuam a serem incorporados no *software* involuntariamente. Como vulnerabilidades são erros ligados a segurança, essa afirmação permanece válida nesse contexto. Para sanar esse problema é essencial submeter o *software* a testes realizados por especialistas em segurança, visando, assim, garantir a segurança do mesmo (NICOLAYSEN et al., 2010). A experiência dos especialistas e o mindset de atacante permitem que estes encontrem falhas ignoradas ou inseridas durante o processo de desenvolvimento (KANNIAH; MAHRIN, 2016). Testes especializados incluem, mas não estão

limitados, a: *Penetration Testing*, *Red Team Testing*, *Risk Based Testing* e *Vulnerabilities Assessments*.

b) Como Aplicar

A aplicação de testes especializados, normalmente, requer a presença de um ou mais especialistas em segurança. Logo, uma das formas de aplicar essa abordagem é contratar uma empresa ou profissionais autônomos que ofereçam este serviço. Em cenários mais críticos, o próprio projeto pode contar com um time desses profissionais, para realizar esta tarefa. O método mais apropriado para testar o *software*, deve ser acordado com os especialistas, de modo que a tarefa seja a performada da forma mais eficiente possível, seguindo as características específicas de cada projeto.

c) Quando Aplicar

O momento ideal para aplicar esse tipo de teste seria coincidente com o lançamento de cada versão da aplicação. Contudo, nem sempre essa configuração ideal é possível, devido a restrições, explanadas a seguir. Uma abordagem possível é realizar esses testes antes de cada Major Release, ou seja, entregas que alteram muitos aspectos do *software* e incluem diversas novas funcionalidades (CHÓLIZ; VILAS; MOREIRA, 2015). De acordo com a opinião dos especialistas, é responsabilidade do time, em colaboração com o cliente, definir uma agenda para realização desses testes, de modo que o *software* seja testado da forma mais adequada possível.

d) Envolvidos

O time de desenvolvimento, o cliente e os especialistas em segurança devem estar presentes nessa prática.

e) Restrições

Realizar testes especializados é uma tarefa custosa, tanto do ponto de vista financeiro, quanto em relação ao tempo requerido (BEZNOSOV; KRUCHTEN, 2004). Desse modo, tempo e orçamento são restrições naturais desse processo. Além disso, testes especializados, muitas vezes, são baseados na documentação do projeto (BEZNOSOV; KRUCHTEN, 2004), como nos métodos ágeis, por vezes a documentação formal é desconsiderada, esta pode ser outra restrição aplicável.

f) Trabalhos Relacionados

(Oueslati; Rahman; Othmane, 2015), (KANNIAH; MAHRIN, 2016), (BACA; CARLSSON, 2011), (NICOLAYSEN et al., 2010), (Oyetoyan; Cruzes; Jaatun, 2016), (Bansal; Jolly, 2014), (CHÓLIZ; VILAS; MOREIRA, 2015),

(SINGHAL et al., 2012), (DAUD, 2010), (MCGRAW, 2004), (FRANQUEIRA et al., 2011) , (SEVEN..., 2019), (MICROSOFT..., 2019), (CLASP, 2019)

4.6.3 Desafios

A falta de documentação no mundo ágil, pode ser considerada um desafio para o processo de testes, por conseguinte, esta restrição se estende para esta política (Oueslati; Rahman; Othmane, 2015). Some-se a isso o fato de que, mesmo que sejam performados vários testes, cobrir todos os cenários passíveis de ameaça, é uma tarefa virtualmente impossível (Oueslati; Rahman; Othmane, 2015). Assim, balancear custo e tempo de desenvolvimento é um desafio devido à natureza limitada desses recursos (KANNIAH; MAHRIN, 2016). Além disso, segundo a visão dos especialistas, testes mal performados podem consumir recursos e não apresentar os resultados esperados.

4.6.4 Impacto

Testes especializados rigorosos e bem executados no sistema podem expor peças que não são suficientemente seguras. Todos os testes automatizados escritos, em teoria, devem provar que o itens de segurança, foram considerados e implementados. O uso de ferramentas de análise de código estático pode descobrir erros comuns de programação e possíveis problemas, ainda, durante o desenvolvimento (NICOLAYSEN et al., 2010).

5 VALIDAÇÃO

Este capítulo apresenta resultados, da validação das políticas propostas. São expostas as transcrições das entrevistas, assim como uma analise das opiniões obtidas.

5.1 ENTREVISTAS COM ESPECIALISTAS

Nesta seção, são apresentados os resultados de entrevistas, as quais foram realizadas com 10 profissionais, com os seguintes perfis: Especialista em Metodologias Ágeis, Desenvolvedores de *Software* de Times Ágeis com Experiência em Segurança e Especialistas em Segurança. Para cada uma das entrevistas, serão expostas a experiência de mercado de cada um dos entrevistados e suas opiniões sobre cada uma das políticas apresentadas.

5.1.1 Entrevista 1

1. Perfil do Profissional

   Engenheiro de *software*, a mais de 16 (dezesseis) anos, trabalhando com métodos ágeis a pelo menos 10 (dez) anos. A pouco mais de 3 (três) anos, passou a trabalhar como desenvolvedor, em uma empresa de segurança, adquirindo experiência na área desde então.

2. Ponto Levantados por Política

   a) Security Readiness Policy

      O primeiro entrevistado, apresenta uma visão positiva acerca da política e das práticas agregadas, para ele essa agregação faz sentido no contexto de segurança para times ágeis. De acordo com sua opinião, é plausível crer que elas podem ser aplicadas em times ágeis. Os principais métodos, propostos por ele, seriam a presença de uma equipe especializada ou uso de plataformas para realizar os treinamentos. Já quanto a pesquisa, conceder tempo para que os profissionais as realizem, pode ser o método mais eficiente. Treinamentos irão ajudar os desenvolvedores, a manter um nível mínimo de familiaridade com desenvolvimento seguro. Já as pesquisas, além de manter o profissional atualizado, podem prover insumo, para

que este apresente os resultados para o resto do time, disseminando conhecimento. Manter o foco das pesquisas, em temas de interesse do time é um desafio, fora isso nenhum outro foi levantado. O entrevistado não apontou nenhuma outra prática a ser adicionada na política.

b) Security Requirements Policy

Apresentar uma política relacionada a requisitos, assim como suas práticas, faz bastante sentido na visão do profissional. Apesar de as práticas poderem ser aplicadas em times ágeis, uma restrição de tempo é válida, uma vez que a adição dessas tarefas, irá consumir tempo do time, que, muitas vezes, já é limitado. Um desafio exposto, seria realizar essas atividades, de forma efetiva, em times com pouca experiência em segurança. O apoio de especialistas em segurança pode ser essencial para execução dessas práticas até que o time ganhe experiência suficiente para ter condições de aplicar a política. Ter clareza sobre quais requisitos de segurança devem ser considerados para o projeto, de acordo com as informações obtidas sobre o tema durante a execução dessa política, é o principal benefício dessa prática, sob pena de seguir o processo sem que haja contribuição real.

c) Security Design and Planning Policy

O entrevistado, confirma a coerência das políticas e das práticas agregadas, sendo, para ele, possível aplicar estas práticas. O *Protection Poker* se adapta de forma natural aos processos ágeis e o estabelecimento de design de segurança pode ser alcançado, por meio de padrões de projetos já existente. A análise de riscos pode ser realizada em conjunto com o *Protection Poker*, podendo, também, ser realizada em conjunto com outras cerimônias, como o *Backlog Grooming*. Duas restrições foram levantadas, sendo a primeira o nível de experiência dos desenvolvedores com relação a segurança e a segunda o fato de que mais atividades demandam mais tempo para a realização das cerimônias dos métodos ágeis. A entrevista aponta para essas práticas, como positivas em relação a adição de segurança ao projeto.

d) Security Implementation Policy

O entrevistado diz que essa política faz sentido, inclusive, que utiliza suas práticas em projetos que utilizam metodologias ágeis. A especificidade de segurança seria mais uma adição dentro do processo, mas poderia ser facilmente incorporada. Segundo ele um desafio a ser considerado é a dinamicidade das regras de desenvolvimento seguro, então deve haver

momentos para atualização dessas, podendo inclusive ter como insumo os aprendizados obtidos, por meio de outras práticas. O impacto é bastante positivo, pois durante essa fase, segundo ele, mais problemas de segurança surgem, podendo estes serem evitados aplicando as práticas dessa política, apesar, de deixar bastante claro, que só elas não são suficientes para garantir segurança.

e) Security Testing Policy

Na visão do entrevistado essa política faz todo sentido, sendo, inclusive, difícil dissociar suas práticas do processo de desenvolvimento seguro. *Security Specialized Testing*, apresenta restrições associadas, de custo e tempo, principalmente se houver necessidade de contratar um time externo. Segundo ele deve ficar claro, que apesar de todas essas atividades de teste, nunca há garantia total de segurança, porém, elas podem corroborar para afirmar racionalmente, que o *software* está seguro. Estas práticas cobrem, praticamente, todos os pontos de vista, relativos a verificação e validação, não sendo apontada nenhuma outra a ser adicionada.

5.1.2 Entrevista 2

1. Perfil do Profissional

   Engenheiro de *software* a aproximadamente 9 (anos), desde o início trabalhando com metodologias ágeis, teve mais contato com segurança, quando foi trabalhar em uma empresa da área, há aproximadamente 6 (seis) meses.

2. Ponto Levantados por Política

   a) Security Readiness Policy

   Segundo o profissional as duas práticas fazem sentido, quanto a desenvolvimento seguro e aos objetivos da política. Realizar treinamento é uma prática bem viável e na visão do entrevistado não demanda tanto tempo. Fomentar a participação em eventos, com ajuda de custo para tal, pode ser uma forma válida de incentivo a pesquisa. Idealmente realizar treinamentos deve ser parte do *onboarding* no time, para que novos profissionais, entendam mais rapidamente, como segurança é abordada no processo do time. Porém, alinhar a necessidade de realização das práticas e o investimento financeiro envolvido é um desafio, as vantagens obtidas devem estar bem claras. O impacto dessa política é positivo, levando

a uma diminuição efetiva das falhas de segurança, quando incorporada durante o processo de desenvolvimento.

b) Security Requirements Policy

De acordo com a entrevista a política e suas práticas tem bastante coerência. Na visão do entrevistado as práticas são factíveis, para times ágeis. Além disso, ele aponta que pensar em segurança iterativamente, pode contribuir para uma melhor avaliação de segurança, para cada requisito do *software*. Um desafio apresentado, especificamente para a escrita das *Evil User Stories*, é a expertise em segurança necessária para realizar tal tarefa. Contudo, isso não as tornam impraticáveis, ainda mais se a *Security Readiness Policy* for seguida. Outro desafio apresentado é negociar com cliente a inclusão dessas atividades, por causa do custo e tempo demandados. É necessário balancear esses contrapontos de forma racional, de modo a não deixar segurança de lado.

c) Security Design and Planning Policy

Segundo o entrevistado toda a política é coerente. Planejar e definir o desenvolvimento do *software* pensando em segurança, é de suma importância. Encontrar soluções palpáveis para alguns problemas evita retrabalho futuramente. Análise de risco, é muito importante, para conseguir negociar requisitos de segurança com o cliente. As práticas vão apresentar benefícios de segurança, durante o andamento do projeto. Na visão do participante, as práticas se incorporam em atividades do desenvolvimento ágil, provendo benefícios de segurança ao longo do projeto.

d) Security Implementation Policy

Segundo o profissional, essa política e suas práticas já estão presentes na maioria dos times ágeis. O *Pair Programming* foi visto como prática eficiente para agilizar a implementação de atividades de segurança complexas e disseminar conhecimento no time. O desafio principal é conscientizar os profissionais para incluir cuidados com segurança nessas práticas, necessidade que pode ser suprida pelos treinamentos já mencionados. Existe um impacto perceptível, na qualidade do código, evitando que erros de segurança mais básicos ocorram.

e) Security Testing Policy

Segundo o entrevistado, as práticas do processo de testes de segurança, estão bem contempladas dentro da política. Os testes automatizados vão enfrentar uma dificuldade de aderência, em times que não os fazem, tam-

bém, para os requisitos funcionais. Quanto aos testes especializados, foi enfatizada importância, porém, custo e tempo, necessários para realização, são uma restrição significativa em times ágeis. Dificilmente, segundo ele, uma equipe preparada para realização destes, vai estar disponível para o projeto, além disso, caso houvesse disponibilidade seria oneroso financeiramente. Entretanto, foi expresso que esse investimento vale a pena e que ela tende a diminuir com o tempo, em relação a todas as práticas.

5.1.3 Entrevista 3

1. Perfil do Profissional

   Engenheiro de *software* a pelo menos 15 (quinze) anos, vem trabalhando com metodologias ágeis a 6 (seis) anos, tendo pelo menos 10 (dez) anos de experiência com desenvolvimento seguro.

2. Ponto Levantados por Política

   a) Security Readiness Policy

   O entrevistado diz que esta política, faz todo sentido, dentro do processo ágil, sobretudo, a prática de treinamento. A práticas de pesquisa, segundo ele, trazem resultados menos palpáveis, mas, não deixam de ser importantes. Para serem eficazes, segundo o profissional, os treinamentos devem ter uma abordagem bastante prática. Não foi levantada nenhuma restrição para essa política, os treinamentos terão como principal benefício a expertise necessária para membros do time realizarem atividades de segurança, durante o processo de desenvolvimento.

   b) Security Requirements Policy

   De acordo com o profissional, o escopo de requisitos de segurança do *software*, deve ser bem definido, nesse contexto, a política e suas práticas, tem coerência. As práticas levantadas, foram vistas como plausíveis, para times ágeis. Segundo o entrevistado, o *Security Backlog* pode ser uma boa maneira de manter os desenvolvedores cientes do que será implementado, no que concerne a segurança. De modo afirmativo, o profissional aponta para a necessidade de o time se preocupar em propor atividades de segurança quando o cliente não estiver ciente dela. Os *Evil User Stories*, que nunca haviam sido vistas pelo profissional, lhe pareceram um método

interessante para facilitar a visão do time sobre como o *software* seria atacado. Ainda, segundo ele, essa política pode contribuir para a construção de *software* mais consistente e resiliente com relação a segurança.

c) Security Design and Planning Policy

Está política foi avaliada, como coerente pelo entrevistado. O *Protection Poker*, segundo ele, se adequaria naturalmente, trazendo alguns benefícios em relação a segurança, quanto a priorização de atividades. A definição de design de segurança foi considerada a mais importante das práticas, tendo um grande impacto na segurança do projeto. Esta prática, quando aplicada corretamente, evitaria diversos erros de projeto, assim como, sua perpetuação sucessiva nas iterações. Ele aponta que para minimizar as chances de erro, durante essa etapa, todo o time deve ser consultado, visando obter pontos de vista diferentes sobre o assunto. A opinião do entrevistado sobre análise de risco é que ela pode ser desafiadora no mundo ágil, principalmente, por causa, do tempo demandado para realizá-la.

d) Security Implementation Policy

O profissional destaca, que as três prática fazem sentido, assim como, a política que as engloba. Segundo ele, são atividades simples de aplicar em times ágeis, pelo fato de estes já estarem familiarizados com essas práticas. O desenvolvedor indica grandes desafios, mas, percebe grandes benefícios com relação à segurança, sendo que, os principais seriam: evitar erros simples de serem evitados durante a implementação e passar conhecimento para os integrantes de time, que detenham menos experiência em segurança.

e) Security Testing Policy

Para o profissional faz todo sentido essa política e suas práticas. Segundo o mesmo, implementar testes automatizados e realizar análise de código, para segurança, já impactam positivamente o desenvolvimento. O testes por parte de especialistas, seriam muito importantes, contudo colocar dentro do processo ágil, pode ser desafiador. Nem sempre, haverá disponibilidade de recurso para realizar esses testes. Além disso, ele indica, que pode haver uma certa resistência, por parte de times ágeis que ainda não implementam testes automatizados, dentro do seu processo.

5.1.4 Entrevista 4

1. Perfil do Profissional

   Engenheiro de *software* com 12 anos de experiência na área, todos com uso de metodologias ágeis, possui 7 anos de experiência com desenvolvimento seguro, atualmente, ocupa um cargo de gerência do time de engenharia em uma empresa de segurança.

2. Ponto Levantados por Política

   a) Security Readiness Policy

   Segundo o profissional, faz todo sentido, relacionar essa política e as suas práticas. Para termos bons resultados, o time deve ter um com conhecimento sobre segurança e treinamento pode prover isso. Pesquisa em segurança, pode ser mais difícil a princípio, mas com o tempo, se torna mais fácil de realizar. De acordo com a opinião do profissional, os treinamentos deveriam ser atualizados, pelo menos, uma vez ao ano. Essa pŕatica tem um benefício apontado como sendo de suma importância, que seria possibilitar a mudança na mentalidade de time em relação a segurança.

   b) Security Requirements Policy

   Segundo o entrevistado, faz sentindo a presença dessas práticas, nessa política. Porém ele, atenta, para a necessidade de um profissional, com experiência em segurança durante a realização das práticas. Geralmente, a mentalidade do desenvolvedor, está voltada para como se proteger, isso pode fazer com que muitos cenários de ataque ao *software*, passem despercebidos. Ele afirma, que em times ágeis, *Product Owner* ou *Scrum Master*, poderiam assumir essa tarefa, caso tenham expertise em segurança, caso contrário, deve haver um acompanhamento inicial, por parte de um especialista na área. Segundo o entrevistado, as *Evil User Stories* poderiam ser incluídas na próprias *User Stories*, facilitando sua inclusão nas cerimônias do mundo ágil. O principal impacto seria ter um bom investimento em segurança, desde o início do projeto, tendo em vista que se considerada tardiamente no projeto, poderá haver a geração de grandes problemas, podendo inclusive, colocar em risco o projeto todo.

   c) Security Design and Planning Policy

   De acordo com o entrevistado, faz sentido aplicar esta política e suas práticas. O *Protection Poker*, pode ser visto como mecanismo para dis-

cutir soluções de segurança e obter mais informações, a fim de priorizar como mais eficácia. Estabelecer um design seguro foi visto como uma atividade muito importante. Já a análise de risco, ele não tem certeza, se a aplicação seria feita corretamente, além disso, existe um risco, de ser deixada de lado conforme o *software* vai sendo desenvolvido. Ele aponta que existem outras práticas que poderiam ser adicionadas nesse processo, contudo, elas ocasionariam sobrecarga, diminuindo a agilidade.

d) Security Implementation Policy

O entrevistado salientou, que esta pode ser, uma das mais importantes políticas do processo, visto que, mesmo as outras práticas sendo implementadas, não havendo cuidado no processo de implementação, as falhas de segurança continuarão a ocorrer. No entanto, ele afirma que existe uma dependência quanto a experiência dos desenvolvedores com segurança, para que as práticas, sejam efetivamente aplicadas. Alguns desafios apontados são, monitorar a aplicação dessas práticas e manter a mentalidade do time em relação a segurança. Evitar retrabalho é o principal valor agregado desta política, de acordo com o profissional.

e) Security Testing Policy

A política de testes de segurança foi bem avaliada pelo entrevistado. Apenas para prática de testes especializados, foi apresentada uma restrição, relativa ao custo para realizá-los, mas, que poderiam ser reduzidos como planejamento para tal, uma método sugerido seria aplicá-los apenas em entregas mais críticas. Essa política é de suma importância para a segurança do projeto, por se tratar de uma última barreira, antes que o *software* vá para produção. Segundo o profissional, acrescentar outras práticas nessa política pode não trazer benefícios suficientes para cobrir o custo de realização.

5.1.5 Entrevista 5

1. Perfil do Profissional

   Analista de segurança com 5 (cinco) anos de experiência na área.

2. Ponto Levantados por Política

   a) Security Readiness Policy

   Proporcionar aos profissionais o básico na área de segurança é essencial para modelar a mentalidade dos desenvolvedores, contribuindo para que

estes tomem mais cuidados em diversas situações. Pesquisa, foi abordada pelo entrevistado, como sendo uma atividade muito dependente do perfil dos profissionais, mas, que traria vários benefícios de segurança. Foi apontado que, *software* totalmente seguro, é virtualmente impossível de alcançar, porém, preparar o time é a melhor forma começar a lidar com segurança.

b) Security Requirements Policy

Empenhar tempo durante o processo de levantamento de requisitos, em relação a segurança, faz todo sentido, na visão do profissional. Um time treinado, teria condições de apresentar requisitos básicos de segurança, comuns a maioria dos projetos de *software*. Discutir com o cliente sobre esses requisitos, irá ajudar o time a entender as necessidades do ponto de vista de negócio. As *Evil User Stories*, poderiam ser produzidas a partir de ataques já conhecidos. Todas as práticas foram tidas como benéficas para o processo de desenvolvimento de *software* seguro.

c) Security Design and Planning Policy

O entrevistado expressou que a política tem coerência. Análise de risco é uma atividade de suma importância para segurança de *software*. O *Protection Poker* ajuda o desenvolvedor a poupar tempo, priorizando atividades que envolvam maior risco de segurança. Definir design de segurança pode ser uma tarefa complexa, na visão do profissional, mas, é de suma importância para evitar falhas.

d) Security Implementation Policy

A política apresentada foi bem vista pelo profissional. Revisão de código pode encontrar diversas falhas de segurança, apesar de que algumas podem passar despercebidas, por se tratar de partes específicas do código, sem uma visão do todo. Todas as práticas são consideradas agregadoras de contribuições válidas para segurança, todavia, na visão do entrevistado, tempo pode ser um desafio.

e) Security Testing Policy

O profissional considera que a política de teste é imprescindível no concernente a segurança. Os testes automatizados, seriam a primeira barreira, evitando implementar código inseguro. Os testes especializados são essenciais, mas, são custosos financeiramente, além disso os desenvolvedores devem atentar para correção das vulnerabilidade levantadas, senão perde sentido fazer tal investimento.

5.1.6 Entrevista 6

1. Perfil do Profissional

   Atualmente analista de segurança, possui 21 (vinte e um) anos de experiência na área de computação, destes 9 (nove) são dedicados a área de segurança.

2. Ponto Levantados por Política

   a) Security Readiness Policy

   Segundo o entrevistado, aplicar essa política faz sentido, mas, é difícil, pois, a depender de como ela for implantada, os profissionais podem não levar o treinamento a sério. Pesquisa de segurança, pode não ser uma prática eficiente para desenvolvedores, pela falta de familiaridade com a área. Mesmo assim, a política é válida, como forma de conscientizar os times sobre a carência de segurança em projetos do *software*.

   b) Security Requirements Policy

   O entrevistado diz que a política pode até contribuir para segurança no projeto. No entanto, salienta que existe uma dificuldade para o desenvolvedor, entender de forma clara como expressar os requisitos de forma condizente com a realidade.

   c) Security Design and Planning Policy

   Segundo o entrevistado, essas práticas são de suma importância para segurança, porém apenas o treinamento de segurança, já abordado, pode não ser o suficiente para munir os desenvolvedores do conhecimento necessário para sua realização.

   d) Security Implementation Policy

   Segundo o profissional a política tem apenas uma vantagem, que seria conscientização, no mais as análises automatizadas poderiam trazer mais benefícios. O uso de ferramentas nesse processo é visto como essencial. O *Pair Programming* é apontado como tendo uma vantagem extra, qual seja, perpetuar conhecimento dentro do time.

   e) Security Testing Policy

   A política de teste, foi apontada, como sendo uma das mais positiva. O uso de ferramentas para análise de código, traz diversos benefícios, além de não tomarem muito tempo, por serem passíveis de automação. Os testes especializados são essenciais para encontrar cenários mais comple-

xos de ameaça, contudo, implicam em aumento de demanda de tempo e recursos para desenvolvimento do projeto.

5.1.7 Entrevista 7

1. Perfil do Profissional

   Engenheiro de *software* com mais de 8 (oito) anos de experiência em desenvolvimento seguro, todos estes trabalhando em times ágeis.

2. Ponto Levantados por Política

   a) Security Readiness Policy

   Para o entrevistado, esta política faz todo sentido, preparar o profissional é essencial para conseguir bons resultados. A prática de treinamento, é muito coerente, fazer com que o desenvolvedor aprenda a lidar com aspectos de segurança, é vital para desenvolvimento de *software* seguro. A parte de pesquisa, segundo o profissional, deve ter um foco bem específico, em novas formas de defesa, ferramentas de auxílio ao desenvolvimento, entre outras, caso não ocorra dessa forma, pode não trazer o resultado esperado. É importante, que quando realizada uma pesquisa, seus resultados devam ser apresentados para o restante do time. As duas práticas, são avaliadas como aplicáveis em times ágeis, nenhuma outra foi proposta para complementar a política.

   b) Security Requirements Policy

   Esta política, foi apontada como coerente pelo entrevistado. Segundo ele, segurança sempre tem um aspectos de requisito não funcional, sendo mais difícil, convencer o cliente a priorizá-lo. De acordo com o profissional, apesar das práticas adicionarem contribuição a segurança, aplicar *Threat Modeling*, pode ser bem mais eficaz em relação a segurança. Ele afirma que introduzir essa prática no mundo ágil é um grande desafio, dado a natureza complexa e densa, do modelo.

   c) Security Design and Planning Policy

   O entrevistado, vê muita coerência na política, principalmente, se associada com a política anterior. O profissional afirma que resultado das práticas anteriores vão servir de insumo para aplicar as práticas dessa política. Ele aponta que as atividades desta política podem evitar diversos problemas no futuro do projeto. Mesmo assim, ele argumenta que

Threat Modeling poderia prover melhores insumos para essas práticas, mas salienta a dificuldade de tal modelo ser incorporado no mundo ágil.

d) Security Implementation Policy

Todas as práticas fazem sentido no contexto de segurança, segundo o entrevistado. O *Secure Code Rules*, serve tanto para guiar o desenvolvedor, quanto como insumo para execução de ferramentas de análise estática. Sendo assim, pode ser aplicado como linha norteadora de desenvolvimento ou para guiar o processo de análise, via outra ferramenta, posteriormente. Um desafio exposto, diz respeito a experiência do time em relação a segurança, principalmente, no concernente a *Pair Programming* e *Code Review*. Todas a práticas possuem um impacto positivo, na segurança do projeto, sendo um ponto positivo o fato de que elas poderiam ser facilmente incorporadas ao processo ágil.

e) Security Testing Policy

Apresentar uma política de testes, faz todo sentido, na visão do entrevistado, principalmente, se ela estiver integrada à outras políticas apresentadas. Segundo ele, toda forma de teste traz benefícios, sejam eles, automatizados, ferramentais ou manuais, e todo time deveria usar esses recursos, principalmente os ágeis, para garantir uma entrega eficiente, isto é, com poucos problemas. Contudo, ele afirma que, tanto análise de código por ferramentas quanto por testes especializados, possui uma restrição financeira associada, pois, licenças de uso de boas ferramentas, assim como, a contratação de especialistas em segurança, normalmente, tem um custo elevado. Segundo ele, essa prática já cobre toda uma suíte de teste, adicionar outras só elevaria o custo desnecessariamente.

5.1.8 Entrevista 8

1. Perfil do Profissional

   Engenheiro de *software* a mais de (21) anos, especialista em ágil, possui 12 (doze) anos de experiência nessa área, passou a ganhar experiência em segurança a 2 (dois) anos, quando entrou para uma empresa da área.

2. Ponto Levantados por Política

   a) Security Readiness Policy

   A política faz sentido para o profissional, que considera a preparação do time um processo essencial para o desenvolvimento seguro. Realizar

pesquisa pode ser uma prática desafiadora, para o time, o material de segurança pode ser escasso, se comparado a outras áreas, mas, essa prática contribui para que os desenvolvedores e outros membros do time se mantenham atualizados. Encaixar pesquisa em times ágeis, também pode ser desafiador em relação ao tempo desprendido para realizá-las de forma eficaz. Já os treinamentos, devem ser realizados, independente do processo utilizado, no entanto, o entrevistado defende que um método de avaliação do conhecimento deveria ser aplicado. As duas práticas trazem benefícios para o processo de desenvolvimento seguro, desde que se tome os devidos cuidados para que elas sejam realizadas de maneira coerente, desse modo, essa deve ser a preocupação da organização e do time ágil.

b) Security Requirements Policy

O entrevistado julgou a política, como parte essencial do processo, pois, ter um mapeamento de requisitos de segurança, dá bastante visibilidade a estes, tanto para o time quanto para o cliente. Segundo ele, idealmente, essas práticas deveriam ser realizadas ou acompanhadas por um profissional com experiência em segurança, isto facilitaria as conversas com o cliente. Existe uma preocupação do entrevistado quanto a interferência da adição de requisitos de segurança em outros pontos do projeto, como usabilidade e performance, sendo, assim, necessário envolver membros do time, com essas expertises, para conciliar esses interesses e evitar problemas.

c) Security Design and Planning Policy

Segundo o entrevistado, a política apresenta bastante coerência com a necessidade de segurança. O *Protection Poker* agrega mais uma informação importante na hora de priorizar os requisitos, além disso, levanta discussões valiosas, em relação aos itens de segurança. Decisões de design de segurança, como arquitetura do *software*, são essenciais para evitar problemas futuros. A análise de risco é muito importante, porém, existe o desafio de manter a simplicidade, para que ela se adapte ao processos ágil. O entrevistado aponta para um novo documento simples, utilizado no seu time, no qual as decisões de design, são registradas, assim como, os motivos de serem implementadas ou não. Segundo ele, todas as práticas estão bem alinhadas com o objetivo da política, sendo, de grande valor para o desenvolvimento seguro.

d) Security Implementation Policy

De acordo com o profissional, essa política tem coerência, sendo plausível para times ágeis. Entretanto, *Secure Code Review* e *Pair Programing*, para serem efetivas, dependem da existência de membros do time com perfil de segurança, sendo que esse ponto pode ser atacado com treinamento. *Secure Code Rules* seria uma prática mais desafiadora, pelo fato de não ser tão simples definir as regras. Desse modo, ele entende que todas as práticas trazem benefícios para segurança no desenvolvimento de *software*, sendo que, pelo menos, *Secure Code Review* deve ser uma prática mandatória.

e) Security Testing Policy

Essa política e suas práticas fazem sentido, na visão do profissional, podendo fazer parte do processo de times ágeis. O *Security Automated Testing* e *Security Code Analysis*, são simples de serem postos em prática, devendo sempre serem aplicadas. Os *Security Specialized Testing*, são importantes, mas, contratar esses testes ou um profissional com essa expertise para o time, pode ser, financeiramente custoso, tornando estes, mais difíceis de serem incorporados, em time ágeis.

5.1.9 Entrevista 9

1. Perfil do Profissional

   Engenheiro de *software* a 14 (catorze) anos, possui 9 (nove) anos de experiência com metodologias ágeis, a 4 (quatro) anos vem trabalhando com desenvolvimento seguro.

2. Ponto Levantados por Política

   a) Security Readiness Policy

   A política como um todo, faz sentido, na visão do profissional, mas requer bastante planejamento, para ser incorporada ao mundo ágil, além disso, o tempo desprendido para as práticas deve ser devidamente alocado. A prática de pesquisa, dificilmente vai surgir de forma espontânea dentro do time, então, é importante que haja incentivos para tanto. Dedicar tempo do profissional para que ele estude as tecnologias que está usando traz benefícios independentemente da área. Treinamento é essencial, podendo ser menos custoso que realizar correções de segurança, posteriormente. Implantar essa prática, vai trazer mais resiliência ao *software* resultante do processo de desenvolvimento.

b) Security Requirements Policy

A política é coerente, apesar de existir uma dificuldade, apontada pelo entrevistado, em visualizar, como o time vai apresentar esses requisitos, para o cliente ou *Product Owner*. Apesar disso, estando o processo para realização dessas práticas bem definido, elas seriam praticáveis em times ágeis. Segundo o profissional, vale a pena uma adequação dos papéis envolvidos, incluindo a presença de um consultor ou membro do time, com expertise em segurança, pelo menos, até que o time tenha maturidade suficiente em relação a segurança, e possa dar seguimento a essa atividade de forma autônoma. Apesar dos contrapontos, segundo o entrevistado, essa é a melhor etapa do processo de desenvolvimento para se preocupar com segurança.

c) Security Design and Planning Policy

De acordo com o entrevistado, uma política visando aspectos de planejamento e design do projeto, relacionados a segurança, tem grande valor. É importante, que decisões de design e arquitetura, sejam tomadas previamente, pois, nem sempre o desenvolvedor, vai ter experiência suficiente para tomá-las durante a implementação, principalmente nos aspectos relativos à segurança. *Protection Poker* foi citada como sendo interessante, pois opiniões, de desenvolvedores com mais ou menos experiência, podem gerar discussões válidas sobre os requisitos, durante as sessões dessa atividade. Essa política traz benefícios para segurança nos projetos, de acordo com o entrevistado.

d) Security Implementation Policy

O entrevistado, afirma que as políticas e práticas fazem sentido, porém, o grande desafio está na maturidade do time, em relação a segurança. Pode ser necessário, que membros mais experientes do time, tomem para si responsabilidade sobre estas atividades, de modo que elas sejam bem performadas. Com o tempo, a maturidade do time, tende a evoluir, fazendo com que estas práticas tragam ainda mais benefícios de segurança para o projeto. Segundo ele, estas práticas têm total condição de serem aplicadas em times ágeis, visto que ele já as vê sendo praticadas com outros vieses.

e) Security Testing Policy

Para o profissional, a política tem coerência, um ponto interessante, é que ele, nunca havia atentado para escrita de testes automatizados de

segurança. Um desafio, segundo ele, é balancear o custo desses testes com outras necessidades do projeto, avaliando, quanto investimento deve ser aplicado nessas práticas. O profissional afirma, que essas práticas aumentam as chances de produzir *software* seguro. Segundo ele um planejamento específico para os testes especializados deve ser feito, considerando tempo e impacto financeiro, por se tratar de uma prática custosa em ambos pontos de vista. Uma opção apresentada seria realizá-los, apenas, em entregas principais ou críticas.

5.1.10 Entrevista 10

1. Perfil do Profissional

   Há 9 (nove) anos no mercado, sendo 2 (dois) anos como desenvolvedor e 7 (sete) como analista de segurança, está ocupando um cargo de gerência faz, aproximadamente, 1 (um) ano em uma equipe de consultoria em segurança.

2. Ponto Levantados por Política

   a) Security Readiness Policy

   Essa política faz todo sentido, de acordo com o entrevistado, incentivar a conscientização em segurança desde o início seria muito importante diante das ameaças vistas atualmente. Os treinamentos, na visão do profissional, podem ser realizados antes dos projetos, diminuindo o impacto de tempo. Ele inclusive elucida que esta prática é fundamental para qualquer time de desenvolvimento, dado a importância, de ter um time com treinamento em segurança. No caso da pesquisa, o entrevistado acredita que a melhor forma, é consumir material relacionado a segurança, diariamente, blog posts, listas de email, grupos de redes sociais e fóruns de discussão são apontados como possíveis fontes. Isso mantém o time atualizado, podendo, até, guiar as pesquisas futuras.

   b) Security Requirements Policy

   O entrevistado, considera muito importante uma política relacionada a requisitos, sendo esta bastante coerente. Sob a ótica de segurança, esta já deveria ser tratada como requisito funcional, destinando tempo e recursos para sua implementação. Todas as práticas fazem sentido em relação a segurança para o profissional. Pensando no mundo ágil, ele afirma que um dos desafios é tempo para realização das atividades. Outro desafio, especificamente relacionado às *Evil User Stories*, é que desenvolvedores,

geralmente, não possuem mentalidade de atacante, desse modo, os cenários, gerados por essas podem não estar condizentes com a realidade ou serem insuficientes. Incluir *Threat Modeling* nesse processo, poderia ser uma abordagem válida, mas, como outros entrevistados, ele afirma que esse processo, pode ser muito pesado para o mundo ágil. Mesmo assim, segundo ele, a políticas pode trazer benefícios para o desenvolvimento ágil de *software* seguro.

c) Security Design and Planning Policy

Esta política apresenta bastante coerência, de acordo com o entrevistado. Análise de risco é uma prática muito importante no mundo da segurança, todavia, leva bastante tempo para ser realizada, efetivamente, podendo não ser encaixar em metodologias ágeis. Porém se realizada, contribui amplamente, para o entendimento das necessidades de segurança do projeto. Estabelecer requisitos de design de segurança, no início do projeto, é fundamental, visto que eles são a base para que o *software* em desenvolvimento se mantenha seguro e para que requisitos de segurança levantados posteriormente sejam mais facilmente integrados. *Protection Poker* foi avaliado positivamente, pelo entrevistado, sendo a inclusão de risco como fator de priorização considerada como sendo de suma importância.

d) Security Implementation Policy

Segundo o entrevistado, essa política e suas práticas fazem sentido, para desenvolvimento seguro. Segundo ele, *Secure Code Review* evita que falhas simples passe despercebidas. Já *Pair Programming*, pode realmente, contribuir para difundir os conhecimentos dentro do time, além de ajudar no desenvolvimento de requisitos de segurança mais complexos. *Secure Coding Rules* devem ser definidas, podendo ser auxiliadas, por meio de ferramentas, como IDE's, que são preparadas para isso. A principal contribuição dessa política é diminuir o quantitativo de erros inseridos durante a implementação.

e) Security Testing Policy

Segundo o profissional, essa política é bastante coerente, sendo uma das mais importantes. No caso da escrita de testes automáticos, o entrevistado citou que no mínimo eles devem ser feitos para comprovar o funcionamento de requisitos de segurança, mesmo que não venham a cobrir em cenários reais de ataques. Análise de código, foi avaliada como sendo fácil de incorporar no processo ágil de desenvolvimento e complementar a prática de *Secure Code Review*, capturando falhas que passaram des-

percebidas na revisão. Deve-se considerar que ferramentas que realizam esse tipo de análise, geralmente, tem licenças caras e as opções gratuitas podem não performar como desejado. *Security Specialized Testing* pode ser aplicada, inserindo um profissional com esse perfil no time ou contratando uma empresa que o faça externamente. Salienta-se que ambas as abordagens, possuem um custo alto associado, desse modo, o investimento nesse tipo de teste, deve ser avaliado de acordo com a criticidade do *software* e com a quantidade de vezes que seriam necessárias para realizar esses testes e, assim, fornecer, uma cobertura satisfatória. No mais, é necessário tomar cuidado, ao realizar estes testes, pois nem sempre a quantidade de vulnerabilidades encontradas e corrigidas é uma boa métrica de avaliação, sendo necessário atentar para a qualidade dos testes realizados.

5.2 RESULTADOS

A partir das opiniões coletadas, durante as entrevistas, foi possível avaliar os pontos positivos e negativos de cada uma das políticas. Abaixo descrevemos para cada um deles.

5.2.1 Security Readiness Policy

Considerada como sendo uma das políticas mais válidas pelos entrevistados, esta política tem como principal ponto positivo, educar membros de times ágeis a lidarem com aspectos de segurança. De acordo com as opiniões coletadas, a falta de experiência é uma dos principais fatores que influencia na falta de segurança em projetos de desenvolvimento de *software*. Realizar treinamentos, além de não custar muito tempo da equipe, na visão da maioria do profissionais, apresenta bons resultados no que diz respeito a segurança.

Em relação à pesquisa em segurança, algumas ressalvas foram levantadas. O time pode não estar preparado para realizar essa atividade, mesmo após o treinamento. Pesquisa pode consumir tempo hábil do projeto, se não for bem direcionada, podendo se tornar um investimento falho. Existe uma dependência da vontade do profissional, em procurar realizar essa prática. Contudo foi demonstrado que existem benefícios associados e manter o time atualizado é um deles. O Entrevistado 10 afirmou, que consumir material sobre segurança no cotidiano do time, pode ajudar a engajar os profissionais nessa prática. No geral a política é vista como coerente,

podendo guiar times ágeis no processo de desenvolvimento seguro.

5.2.2 Security Requirements Policy

Para esta política, o resultado das entrevistas, congrega opiniões mais diversas. Muitos deles, afirmam que mesmo após treinamento, um time ágil não teria maturidade suficiente para aplicar suas práticas. Já outros afirmam, que seria plausível, para um time treinado seguir suas práticas. Umas das práticas mais apontadas, como desafiadoras, nessa política, foi o uso das *Evil User Stories*. Segundo a maioria dos entrevistados uma mentalidade de atacante é necessária para escrevê-las, característica esta, dificilmente, presente em desenvolvedores, os quais, em geral, possuem uma mentalidade focada na construção do *software*.

Por segurança, se tratar de um requisito não funcional, existe uma preocupação, sobre como seus requisitos seriam expostos ao cliente. Alguns dos profissionais afirmam que uma consultoria por especialistas ou um papel de segurança no time poderia mitigar esse problema. De modo geral, todos concordam que o processo de requisitos requer uma atenção no que concerne a segurança e observando as práticas dessa política poderíamos ter a visibilidade necessária para adicionar segurança no processo de desenvolvimento.

5.2.3 Security Design and Planning Policy

Em relação a esta política, alguns dos entrevistados relataram preocupações relacionadas a tempo e complexidade para com as práticas de análise de riscos e design de segurança. Mesmo assim, as duas foram vistas como sendo significativas para garantir segurança de *software*. A prática de *Protection Poker* apresentou poucas ressalvas, este fato pode estar relacionado a proximidade dela com uma pŕatica ágil, o *Planning Poker*. Mesmo com as ressalvas, não foi apontado nenhum impedimento definitivo para aplicar essas práticas no mundo ágil, mas, podemos perceber a necessidade de uma maior adequação de duas, das três práticas, dado a importância dos benefícios apresentados.

5.2.4 Security Implementation Policy

As práticas dessas políticas apresentaram o maior nível de aceitação, em relação a aplicabilidade em times ágeis. Vários dos entrevistados, indicaram a facilidade de incluí-las em times ágeis, por se tratarem de atividades já realizadas com outros propósitos. Apenas alguns dos profissionais, apontaram a falta de conhecimento

como sendo uma restrição, mas, que poderia ser mitigada com os treinamentos da Política 1 (Ver Subseção 4.2). Além dos poucos desafios expostos, a maioria dos entrevistados, concordaram que as práticas tem bastante contribuição no processo de desenvolvimento seguro.

5.2.5 Security Testing Policy

Algumas das entrevistas levam a crer, que essa política é uma das mais importantes, senão a mais importante. A prática de testes automatizados foi vista positivamente por praticamente todos os participantes. As outras duas práticas tiveram um bom resultado também, porém, para ambas uma restrição de recursos financeiros foi apresentada, para análise de código tem-se o custo de aquisição de licença para ferramenta e para testes especializados o custo de contratação de uma consultoria ou profissional externo ou a inclusão de outro membro ao time. Mesmo assim, alguns dos entrevistados apontam, para o uso de ferramentas gratuitas e para o planejamento prévio dos testes especializados, como forma de reduzir custos. Apesar das restrições, a política foi tida como viável para times ágeis, além disso, traz benefícios para segurança de projetos de desenvolvimento de software.

6 CONCLUSÃO

Como apresentado, durante o decorrer deste trabalho, a necessidade de integrar metodologias ágeis e requisitos de segurança ainda é um problema em aberto. Sendo este causado principalmente, pelo fato de que atividades de segurança, foram, em sua maioria, concebidas a partir do modelo tradicional de desenvolvimento de software. Isto leva a diversos desafios, os quais estão ligados, principalmente, às diferenças entres esses modelos. Tendo este trabalho, contribui como parte da resolução deste problema, apresentando políticas, que possibilitem uma melhor adequação de segurança no mundo ágil.

6.1 CONTRIBUIÇÕES

Nesta investigação, foram, estruturadas e organizadas, 5 (cinco) políticas, tendo como intuito facilitar o acesso a essas informações aos times ágeis, sendo elas: (I) *Security Readiness Policy*, (II) *Security Requirements Policy*, (III) *Security Design and Planning Policy*, (IV) *Security Implementation Policy* e (V) *Security Testing Policy*. Cada uma delas, agrega um subconjunto das 14 (catorze) práticas, selecionada por meio de uma avaliação preliminar com um grupo de especialistas. Uma vez propostas, essas políticas passaram, novamente, pela avaliação com especialistas, dessa vez, no formato de entrevista.

De acordo com os resultados obtidos, foi possível concluir que as políticas, *Security Readiness Policy*, *Security Implementation Policy* e *Security Testing Policy*, são mais coerentes e adequadas para times ágeis. Apesar de algumas ressalvas, suas práticas tiveram melhor aceitação por parte dos especialistas. Todas as opiniões convergem para confirmar que é plausível incorporá-las no processo de desenvolvimento ágil, de forma a suprir necessidades de segurança dos *softwares*, por meio deste desenvolvido.

As políticas *Security Requirements Policy* e *Security Design and Planning Policy*, ainda necessitam de ajustes, para serem incorporadas no mundo ágil. Apesar de algumas das práticas por elas agregadas, já estarem adequadas ao uso em time ágil. De modo geral, ainda, existem restrições aplicáveis, mesmo sendo visível, por parte dos especialistas a importância delas para o desenvolvimento seguro de *software*. Além disso, foi possível observar duas necessidades cruciais para desenvolvimento de *software* seguro, suprir necessidades de treinamento do time e disponibilizar tempo

do projeto para atividades relacionadas a segurança.

6.2 LIMITAÇÕES

O presente trabalho possui limitações, intenta-se, aqui expor as principais. Por se tratar de uma pesquisa exploratória, alguns artigos importantes podem ter passado despercebidos, isso pode levar a não completude das práticas levantadas. A diversidade de perfis e quantidade de entrevistados, pode não ser suficiente, para fazer uma análise completa das políticas propostas.

6.3 TRABALHOS FUTUROS

Como propostas de trabalhos futuros temos: realizar uma análise da prática de *Threat Modeling* e sua aplicação no mundo ágil; proceder uma revisão do cenário de segurança em times ágeis no Porto Digital na cidade de Recife-PE e avaliar restrições de tempo para inclusão de segurança em projetos ágeis.

Referências

ADELYAR, S. H.; NORTA, A. Towards a secure agile software development process. In: IEEE. *2016 10th International Conference on the Quality of Information and Communications Technology (QUATIC)*. [S.l.], 2016. p. 101–106. Citado 3 vezes nas páginas 16, 24 e 26.

AGILE Alliance. 2019. Https://www.agilealliance.org/. Accessed: 2019-11-28. Citado 2 vezes nas páginas 19 e 20.

ANIS, A. et al. Securing web applications with secure coding practices and integrity verification. In: IEEE. *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*. [S.l.], 2018. p. 618–625. Citado 2 vezes nas páginas 23 e 102.

AZHAM, Z.; GHANI, I.; ITHNIN, N. Security backlog in scrum security practices. In: IEEE. *2011 Malaysian Conference in Software Engineering*. [S.l.], 2011. p. 414–417. Citado 9 vezes nas páginas 27, 42, 43, 89, 92, 94, 95, 103 e 106.

BACA, D. *Developing Secure Software : in an Agile Process*. Tese (Doutorado) — , School of Computing, 2012. Citado na página 16.

BACA, D.; CARLSSON, B. Agile development with security engineering activities. In: *Proceedings of the 2011 International Conference on Software and Systems Process*. New York, NY, USA: ACM, 2011. (ICSSP '11), p. 149–158. ISBN 978-1-4503-0730-7. Disponível em: <http://doi.acm.org/10.1145/1987875.1987900>. Citado 32 vezes nas páginas 16, 17, 24, 25, 40, 41, 43, 46, 47, 48, 49, 51, 53, 54, 57, 58, 59, 90, 91, 92, 93, 94, 97, 98, 99, 100, 101, 102, 103, 104, 105 e 106.

BAIZE, E. Developing secure products in the age of advanced persistent threats. *IEEE Security Privacy*, v. 10, p. 88–92, 05 2012. Citado 2 vezes nas páginas 92 e 103.

Bansal, S. K.; Jolly, A. An encyclopedic approach for realization of security activities with agile methodologies. In: *2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence)*. [S.l.: s.n.], 2014. p. 767–772. Citado 16 vezes nas páginas 17, 25, 37, 42, 51, 58, 59, 89, 91, 92, 95, 97, 98, 99, 100 e 103.

Barbosa, D. A.; Sampaio, S. Guide to the support for the enhancement of security measures in agile projects. In: *2015 6th Brazilian Workshop on Agile Methods (WBMA)*. [S.l.: s.n.], 2015. p. 25–31. Citado 24 vezes nas páginas 16, 17, 21, 24, 25, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 58, 89, 92, 94, 95, 96, 97, 102 e 106.

BARTSCH, S. Practitioners' perspectives on security in agile development. In: IEEE. *2011 Sixth International Conference on Availability, Reliability and Security*. [S.l.], 2011. p. 479–484. Citado 14 vezes nas páginas 26, 37, 42, 44, 51, 56, 89, 91, 92, 95, 96, 99, 100 e 102.

BECK, K. *Extreme programming explained: embrace change*. [S.l.]: addison-wesley professional, 2000. Citado na página 20.

BERNHART, M.; MAUCZKA, A.; GRECHENIG, T. Adopting code reviews for agile software development. In: IEEE. *2010 Agile Conference*. [S.l.], 2010. p. 44–47. Citado na página 52.

BEZNOSOV, K.; KRUCHTEN, P. Towards agile security assurance. In: ACM. *Proceedings of the 2004 workshop on New security paradigms*. [S.l.], 2004. p. 47–54. Citado 3 vezes nas páginas 16, 28 e 59.

BODDEN, E. State of the systems security. In: *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings*. New York, NY, USA: ACM, 2018. (ICSE '18), p. 550–551. ISBN 978-1-4503-5663-3. Disponível em: <http://doi.acm.org/10.1145/3183440.3183462>. Citado 4 vezes nas páginas 15, 40, 58 e 97.

BOSTRÖM, G. et al. Extending xp practices to support security requirements engineering. In: ACM. *Proceedings of the 2006 international workshop on Software engineering for secure systems*. [S.l.], 2006. p. 11–18. Citado 15 vezes nas páginas 27, 42, 44, 49, 51, 58, 91, 92, 93, 96, 97, 99, 100, 101 e 104.

BOWEN, J. P. et al. Formality, agility, security, and evolution in software development. *Computer*, IEEE, n. 10, p. 86–89, 2014. Citado 15 vezes nas páginas 17, 23, 40, 41, 42, 44, 46, 54, 89, 91, 92, 96, 99, 101 e 103.

CHÓLIZ, J.; VILAS, J.; MOREIRA, J. Independent security testing on agile software development: a case study in a software company. In: IEEE. *2015 10th International Conference on Availability, Reliability and Security*. [S.l.], 2015. p. 522–531. Citado 13 vezes nas páginas 27, 44, 53, 57, 58, 59, 90, 92, 96, 97, 98, 100 e 103.

CHOUDHARY, B.; RAKESH, S. K. An approach using agile method for software development. In: IEEE. *2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH)*. [S.l.], 2016. p. 155–158. Citado 2 vezes nas páginas 20 e 21.

CLASP. 2019. Https://www.owasp.org/index.php/CLASP$_{C}oncepts.Accessed$: $2019 - 09 - 19.Citado18vezesnaspginas23, 37, 40, 41, 51, 58, 60, 89, 90, 91, 92, 93, 95, 97, 98, 99, 101e$ 103.

COCKBURN, A. *Crystal clear: a human-powered methodology for small teams*. [S.l.]: Pearson Education, 2004. Citado na página 20.

COMMON Criteria. 2019. Https://www.commoncriteriaportal.org/. Accessed: 2019-09-19. Citado 10 vezes nas páginas 23, 40, 41, 48, 91, 93, 94, 99, 101 e 105.

DAUD, M. I. Secure software development model: A guide for secure software life cycle. *Lecture Notes in Engineering and Computer Science*, v. 2180, 07 2010. Citado 16 vezes nas páginas 17, 27, 41, 42, 44, 51, 56, 60, 91, 92, 94, 96, 98, 99, 100 e 106.

DENZIN, N. K.; LINCOLN, Y. S. Introdução: a disciplina e a prática da pesquisa qualitativa. *O planejamento da pesquisa qualitativa: teorias e abordagens*, v. 2, p. 15–41, 2006. Citado na página 29.

ENTENDA o ciberataque que afetou mais de 200 mil PCs em 150 países. 2017. Https://olhardigital.com.br/especial/wannacry/. Accessed: 2019-11-23. Citado na página 15.

ESSAFI, M.; LABED, L.; GHEZALA, H. B. Towards a comprehensive view of secure software engineering. In: IEEE. *The International Conference on Emerging Security Information, Systems, and Technologies (SECUREWARE 2007)*. [S.l.], 2007. p. 181–186. Citado 3 vezes nas páginas 16, 23 e 44.

FRANQUEIRA, V. N. et al. Towards agile security risk management in re and beyond. In: IEEE. *Workshop on Empirical Requirements Engineering (EmpiRE 2011)*. [S.l.], 2011. p. 33–36. Citado 17 vezes nas páginas 38, 41, 42, 47, 48, 56, 58, 60, 89, 91, 94, 96, 97, 98, 99, 100 e 106.

FRIJNS, P.; BIERWOLF, R.; ZIJDERHAND, T. Reframing security in contemporary software development life cycle. In: IEEE. *2018 IEEE International Conference on Technology Management, Operations and Decisions (ICTMOD)*. [S.l.], 2018. p. 230–236. Citado na página 23.

GDPR. 2018. Https://g1.globo.com/economia/tecnologia/noticia/lei-da-uniao-europeia-que-protege-dados-pessoais-entra-em-vigor-e-atinge-todo-o-mundo-entenda.ghtml. Accessed: 2019-11-23. Citado na página 15.

GO Sec. 2001. Https://github.com/securego/gosec. Accessed: 2019-11-23. Citado na página 57.

HIGHSMITH, J.; COCKBURN, A. Agile software development: The business of innovation. *Computer*, IEEE, v. 34, n. 9, p. 120–127, 2001. Citado na página 15.

JAVA Secure Code Guide. 2001. Https://www.oracle.com/technetwork/java/seccodeguide-139067.html. Accessed: 2019-12-23. Citado na página 51.

KANNIAH, S. L.; MAHRIN, M. N. A review on factors influencing implementation of secure software development practices. *International Journal of Computer and Systems Engineering*, World Academy of Science, Engineering and Technology, v. 10, n. 8, p. 3032 – 3039, 2016. ISSN eISSN: 1307-6892. Disponível em: <https://publications.waset.org/vol/116>. Citado 25 vezes nas páginas 17, 23, 37, 39, 40, 41, 50, 51, 58, 59, 60, 89, 90, 91, 92, 93, 95, 97, 98, 99, 100, 101, 102, 103 e 104.

KERAMATI, H.; MIRIAN-HOSSEINABADI, S.-H. Integrating software development security activities with agile methodologies. In: IEEE. *2008 IEEE/ACS International Conference on Computer Systems and Applications*. [S.l.], 2008. p. 749–754. Citado 3 vezes nas páginas 23, 25 e 103.

KOTHARI, C. R. *Research methodology: Methods and techniques*. [S.l.]: New Age International, 2004. Citado na página 29.

LGPD - Lei Geral de Proteção a Dados. 2019. Https://politica.estadao.com.br/blogs/fausto-macedo/lgpd-entenda-o-que-e-a-lei-geral-de-protecao-de-dados-pessoais/. Accessed: 2019-11-23. Citado na página 15.

LIAQAT, I. et al. Review of secure software development in various agile models. *International Journal of Scientific and Engineering Research*, v. 6, p. 371–376, 09 2015. Citado 6 vezes nas páginas 91, 92, 94, 99, 103 e 106.

MAKING a pull request. 2001. Https://www.atlassian.com/git/tutorials/making-a-pull-request. Accessed: 2019-11-23. Citado na página 52.

MANIFESTO Ágil. 2001. Https://www.manifestoagil.com.br/. Accessed: 2019-10-18. Citado 3 vezes nas páginas 15, 39 e 41.

MARTIN, R. C. *Clean code: a handbook of agile software craftsmanship*. [S.l.]: Pearson Education, 2009. Citado na página 50.

MCGRAW, G. Software security. *IEEE Security & Privacy*, IEEE, v. 2, n. 2, p. 80–83, 2004. Citado 13 vezes nas páginas 22, 40, 42, 58, 60, 91, 93, 94, 97, 98, 99, 105 e 106.

MICROSOFT Secure Development Lifecycle. 2019. Https://www.microsoft.com/en-us/securityengineering/sdl/practices. Accessed: 2019-09-19. Citado 21 vezes nas páginas 23, 37, 41, 48, 49, 51, 57, 58, 60, 89, 90, 92, 93, 95, 97, 98, 99, 101, 102, 103 e 104.

MUNETOH, S.; YOSHIOKA, N. Railroadmap: An agile security testing framework for web-application development. In: IEEE. *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation*. [S.l.], 2013. p. 491–492. Citado 10 vezes nas páginas 42, 44, 51, 56, 58, 91, 96, 97, 99 e 100.

NICOLAYSEN, T. et al. Agile software development: The straight and narrow path to secure software? *IJSSE*, v. 1, p. 71–85, 01 2010. Citado 30 vezes nas páginas 16, 17, 23, 24, 26, 37, 40, 41, 42, 43, 44, 48, 49, 51, 55, 56, 58, 59, 60, 89, 90, 91, 92, 95, 96, 97, 98, 99, 100 e 101.

OTHMANE, L. ben; ANGIN, P.; BHARGAVA, B. Using assurance cases to develop iteratively security features using scrum. In: IEEE. *2014 Ninth International Conference on Availability, Reliability and Security*. [S.l.], 2014. p. 490–497. Citado 6 vezes nas páginas 41, 42, 44, 91, 96 e 99.

Oueslati, H.; Rahman, M. M.; Othmane, L. b. Literature review of the challenges of developing secure software using the agile approach. In: *2015 10th International Conference on Availability, Reliability and Security*. [S.l.: s.n.], 2015. p. 540–547. Citado 24 vezes nas páginas 16, 22, 23, 24, 25, 36, 37, 38, 39, 44, 53, 58, 59, 60, 89, 91, 92, 95, 97, 98, 100, 101, 102 e 106.

OWASP - Open Web Application Security Project. 2019. Https://www.owasp.org/index.php/Main$_{P}age.Accessed$: $2019-11-23.Citadonapgina15$.

Oyetoyan, T. D.; Cruzes, D. S.; Jaatun, M. G. An empirical study on the relationship between software security skills, usage and training needs in agile settings. In: *2016 11th International Conference on Availability, Reliability and Security (ARES)*. [S.l.: s.n.], 2016. p. 548–555. Citado 26 vezes nas páginas 23, 26, 37, 42, 49, 51, 53, 54, 58, 59, 89, 91, 92, 93, 94, 95, 97, 98, 99, 100, 101, 102, 103, 104, 105 e 106.

PALMER, S. R.; FELSING, M. *A practical guide to feature-driven development*. [S.l.]: Pearson Education, 2001. Citado na página 20.

PAULE, C.; DÜLLMANN, T. F.; HOORN, A. V. Vulnerabilities in continuous delivery pipelines? a case study. In: IEEE. *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*. [S.l.], 2019. p. 102–108. Citado na página 58.

PROJECT Zero. 2019. Https://googleprojectzero.blogspot.com/. Accessed: 2019-11-23. Citado na página 15.

SAFECODE Fundamental Practices for Secure Software Development. 2018. Https://safecode.org/wp-content/uploads/2018/03/SAFECode$_{F}undamental_{P}ractices_{f}or_{S}ecure_{S}oftware_{D}evelopment_{M}arch$ $2019-11-23.Citado2vezesnaspginas22e$ 23.

SEVEN Security Touchpoints. 2019. Http://www.swsec.com/resources/touchpoints/. Accessed: 2019-09-19. Citado 18 vezes nas páginas 23, 40, 41, 43, 48, 57, 58, 60, 90, 91, 93, 94, 97, 98, 99, 101, 105 e 106.

SINGHAL, A. et al. Integration analysis of security activities from the perspective of agility. In: IEEE. *2012 Agile India*. [S.l.], 2012. p. 40–47. Citado 16 vezes nas páginas 25, 37, 42, 56, 58, 60, 89, 91, 92, 95, 97, 98, 99, 100, 102 e 103.

SIPONEN, M.; BASKERVILLE, R.; KUIVALAINEN, T. Integrating security into agile development methods. In: IEEE. *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*. [S.l.], 2005. p. 185a–185a. Citado 12 vezes nas páginas 16, 23, 27, 40, 41, 42, 44, 91, 93, 96, 99 e 104.

Sodanil, M. et al. A knowledge transfer framework for secure coding practices. In: *2015 12th International Joint Conference on Computer Science and Software Engineering (JCSSE)*. [S.l.: s.n.], 2015. p. 120–125. Citado 23 vezes nas páginas

23, 24, 37, 39, 41, 49, 51, 58, 89, 90, 91, 92, 93, 94, 95, 97, 99, 100, 101, 102, 103, 104 e 106.

SOLMS, R. V.; NIEKERK, J. V. From information security to cyber security. *computers & security*, Elsevier, v. 38, p. 97–102, 2013. Citado na página 21.

'SPOOFING': como foi a invasão do celular de Sérgio Moro, segundo a decisão judicial que mandou prender 4 suspeitos. 2019. Https://g1.globo.com/economia/tecnologia/noticia/2019/07/24/spoofing-como-foi-a-invasao-do-celular-de-sergio-moro-segundo-a-decisao-judicial-que-mandou-prender-4-suspeitos.ghtml. Accessed: 2019-11-23. Citado na página 15.

STOICA, M.; MIRCEA, M.; GHILIC-MICU, B. Software development: Agile vs. traditional. *Informatica Economica*, v. 17, n. 4, 2013. Citado 2 vezes nas páginas 15 e 20.

SUTHERLAND, J.; SCHWABER, K. The scrum guide. *The definitive guide to scrum: The rules of the game. Scrum. org*, v. 268, 2013. Citado na página 20.

Terpstra, E.; Daneva, M.; Wang, C. Agile practitioners' understanding of security requirements: Insights from a grounded theory analysis. In: *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*. [S.l.: s.n.], 2017. p. 439–442. Citado 17 vezes nas páginas 16, 17, 24, 26, 40, 41, 42, 48, 53, 90, 91, 92, 94, 100, 101, 102 e 106.

VIEIRA, M. M. F.; ZOUAIN, D. M. *Pesquisa qualitativa em administração-Teoria e prática*. [S.l.]: FGV Editora, 2005. Citado na página 29.

Villamizar, H. et al. A systematic mapping study on security in agile requirements engineering. In: *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. [S.l.: s.n.], 2018. p. 454–461. Citado 22 vezes nas páginas 17, 23, 24, 26, 37, 38, 39, 40, 41, 43, 44, 46, 53, 89, 91, 92, 95, 96, 99, 100, 101 e 102.

VIVO admite brecha de segurança; milhões de clientes estariam expostos. 2019. Https://www.techtudo.com.br/noticias/2019/11/vivo-admite-vazamento-de-dados-de-clientes.ghtml. Accessed: 2019-11-23. Citado na página 15.

VULNERABILITY Scanning Tools. 2001. $Https://www.owasp.org/index.php/Category:Vulnerability_{S}canning_{T}ools.Accessed: 2019-11-23.Citadonapagina57.$

WANG, W.; GUPTA, A.; NIU, N. Mining security requirements from common vulnerabilities and exposures for agile projects. In: IEEE. *2018 IEEE 1st International Workshop on Quality Requirements in Agile Projects (QuaRAP)*. [S.l.], 2018. p. 6–9. Citado 9 vezes nas páginas 28, 38, 40, 41, 42, 89, 91, 96 e 99.

WILLIAMS, L.; MENEELY, A.; SHIPLEY, G. Protection poker: The new software security"game". *IEEE Security & Privacy*, IEEE, v. 8, n. 3, p. 14–20, 2010. Citado 8 vezes nas páginas 27, 44, 45, 46, 89, 92, 96 e 103.

A CONJUNTO INICIAL DE PRÁTICAS

Este apêndice apresenta, o conjunto inicial de práticas, levantadas na literatura consultadas, incluindo práticas duplicada e ou similares.

Prática	**Referencia**
Security Training	(Oueslati; Rahman; Othmane, 2015) (KANNIAH; MAHRIN, 2016)
Provide Training	(MICROSOFT..., 2019)
Core Security Training	(Sodanil et al., 2015)
Provide Security Training	(Villamizar et al., 2018)
Security Training	(NICOLAYSEN et al., 2010)
Security Awareness Training	(Barbosa; Sampaio, 2015)
Education	(Oyetoyan; Cruzes; Jaatun, 2016)
Initial Education and training	(Bansal; Jolly, 2014)
Security Education & awareness	(SINGHAL et al., 2012)
Security training	(BARTSCH, 2011)
Institute Awareness Programs	(KANNIAH; MAHRIN, 2016) (CLASP, 2019)
Security Backlog	(Barbosa; Sampaio, 2015), (Villamizar et al., 2018), (NICOLAYSEN et al., 2010), (AZHAM; GHANI; ITHNIN, 2011)
Evil User Stories	(Barbosa; Sampaio, 2015)
Abuser Stories	(Villamizar et al., 2018)
Misuse Stories	(NICOLAYSEN et al., 2010)
Research Vulnerabilities	(Barbosa; Sampaio, 2015), (Villamizar et al., 2018), (WANG; GUPTA; NIU, 2018), (FRANQUEIRA et al., 2011)
Protection Poker	(Barbosa; Sampaio, 2015), (Villamizar et al., 2018), (BOWEN et al., 2014), (WILLIAMS; MENEELY; SHIPLEY, 2010)
Security Verifications	(Barbosa; Sampaio, 2015)
Perform Static Analysis Security Testing (SAST)	(MICROSOFT..., 2019)

Perform Dynamic Analysis Security Testing (DAST)	(MICROSOFT..., 2019)
Perform Static Analysis	(Sodanil et al., 2015)
Perform Dynamic Analysis	(Sodanil et al., 2015)
Perform Fuzz Testing	(Sodanil et al., 2015)
Static Code Analysis	(BACA; CARLSSON, 2011), (SEVEN..., 2019)
Dynamic Code Analysis	(BACA; CARLSSON, 2011), (MICROSOFT..., 2019)
Fuzzy Testing	(BACA; CARLSSON, 2011), (MICROSOFT..., 2019)
Static Code Analyses	(NICOLAYSEN et al., 2010)
Automating Security Checks	(Terpstra; Daneva; Wang, 2017), (CHÓLIZ; VILAS; MOREIRA, 2015)
Perform application assessments	(KANNIAH; MAHRIN, 2016) (CLASP, 2019)
Penetration Testing	(KANNIAH; MAHRIN, 2016), (BACA; CARLSSON, 2011), (SEVEN..., 2019), (NICOLAYSEN et al., 2010)
Perform Penetration Testing	(MICROSOFT..., 2019)
Red Team Testing	(BACA; CARLSSON, 2011), (SEVEN..., 2019)
Risk Based Testing	(BACA; CARLSSON, 2011), (SEVEN..., 2019)
Capture security requirements	(KANNIAH; MAHRIN, 2016) (CLASP, 2019)
Define Security Requirements	(MICROSOFT..., 2019)
Establish Security Requirements	(Sodanil et al., 2015)

Security Requirements	(BACA; CARLSSON, 2011) (SEVEN…, 2019), (COMMON…, 2019), (NICOLAYSEN et al., 2010), (Villamizar et al., 2018), (Oyetoyan; Cruzes; Jaatun, 2016), (Bansal; Jolly, 2014), (BOSTRÖM et al., 2006), (BOWEN et al., 2014), (SIPONEN; BASKERVILLE; KUIVALAINEN, 2005), (WANG; GUPTA; NIU, 2018), (BARTSCH, 2011), (LIAQAT et al., 2015), (DAUD, 2010), (MCGRAW, 2004), (FRANQUEIRA et al., 2011), (OTHMANE; ANGIN; BHARGAVA, 2014)
Security Requirements Analysis	(SINGHAL et al., 2012)
Assess Security Requirements	(MUNETOH; YOSHIOKA, 2013)
Code Review	(BACA; CARLSSON, 2011)
Diff Review	(BACA; CARLSSON, 2011)
Reviewing Security Code	(Terpstra; Daneva; Wang, 2017)
Security-oriented tdd: Security tests	(NICOLAYSEN et al., 2010)
Security Unit Testing	(SINGHAL et al., 2012), (BARTSCH, 2011), (MUNETOH; YOSHIOKA, 2013), (DAUD, 2010), (FRANQUEIRA et al., 2011),
Secure Code	(NICOLAYSEN et al., 2010)
Deprecate Unsafe Functions	(Sodanil et al., 2015)
Coding Rules	(BACA; CARLSSON, 2011)
Implement secure development practices	(KANNIAH; MAHRIN, 2016) (CLASP, 2019)
Risk Analysis	(BACA; CARLSSON, 2011), (SEVEN…, 2019), (COMMON…, 2019), (NICOLAYSEN et al., 2010)
Assess Security Risk	(Oueslati; Rahman; Othmane, 2015), (Terpstra; Daneva; Wang, 2017)
Pair Programming	(BACA; CARLSSON, 2011), (Oyetoyan; Cruzes; Jaatun, 2016), (BOWEN et al., 2014)

Establish Design Requirements	(MICROSOFT..., 2019) (Sodanil et al., 2015)
Design Requirements	(BACA; CARLSSON, 2011) (MICROSOFT..., 2019)
Secure Design	(Oyetoyan; Cruzes; Jaatun, 2016)
Security Design	(NICOLAYSEN et al., 2010), (BOSTRÖM et al., 2006)
Identify Vulnerabilities	(Oueslati; Rahman; Othmane, 2015)
Vulnerability Analysis	(Villamizar et al., 2018)
Verify and Validate the Security	(Oueslati; Rahman; Othmane, 2015)
Build Vulnerability Remediation Procedures	(KANNIAH; MAHRIN, 2016) (CLASP, 2019)
Define and Use Cryptography Standards	(MICROSOFT..., 2019)
Use Approved Tools	(MICROSOFT..., 2019) (Sodanil et al., 2015)
Security Tools	(BACA; CARLSSON, 2011)
Prioritizing Security Risk	(Terpstra; Daneva; Wang, 2017), (Villamizar et al., 2018), (BARTSCH, 2011)
Incident Plan	(Barbosa; Sampaio, 2015)
Establish a Standard Incident Response Process	(MICROSOFT..., 2019)
Create an Incident Response Plan	(Sodanil et al., 2015)
Execute Incident Response Plan	(Sodanil et al., 2015)
Incident Response Plan	(BACA; CARLSSON, 2011), (Oyetoyan; Cruzes; Jaatun, 2016), (SINGHAL et al., 2012)
Specifie Security Policies	(Oueslati; Rahman; Othmane, 2015)
Security Specification	(KANNIAH; MAHRIN, 2016)
Threat Modelling	(KANNIAH; MAHRIN, 2016), (BACA; CARLSSON, 2011), (Bansal; Jolly, 2014), (BAIZE, 2012), (BOWEN et al., 2014), (CHÓLIZ; VILAS; MOREIRA, 2015), [17], (SINGHAL et al., 2012), (WILLIAMS; MENEELY; SHIPLEY, 2010), (LIAQAT et al., 2015), (DAUD, 2010), (AZHAM; GHANI; ITHNIN, 2011)

Perform Threat Modeling	(MICROSOFT..., 2019)
Use Threat Modeling	(Sodanil et al., 2015)
Define and Monitor Metrics	(KANNIAH; MAHRIN, 2016) (CLASP, 2019)
Define Metrics and Compliance Reporting	(MICROSOFT..., 2019)
Create Quality Gates/Bugs Bars	(Sodanil et al., 2015)
Quality Gates	(BACA; CARLSSON, 2011), (Oyetoyan; Cruzes; Jaatun, 2016)
Publish Operational Security Guidelines	(KANNIAH; MAHRIN, 2016) (CLASP, 2019)
Manage the Security Risk of Using Third-Party Components	(MICROSOFT..., 2019)
Role Matrix	(BACA; CARLSSON, 2011)
Countermeasures Graphs	(BACA; CARLSSON, 2011), (Oyetoyan; Cruzes; Jaatun, 2016), (BOSTRÖM et al., 2006), (SIPONEN; BASKERVILLE; KUIVALAINEN, 2005)
Review of Security Specification	(KANNIAH; MAHRIN, 2016)
Perform Security and Policy Risk Assessment	(Sodanil et al., 2015)
Conduct Attack Surface Review	(Sodanil et al., 2015)
Perform Attack Surface Analysis/Reduction	(Sodanil et al., 2015)
Attack Surface Reduction	(BACA; CARLSSON, 2011)
Attack Surface Analysis	(Oyetoyan; Cruzes; Jaatun, 2016)
Certify Release and Archive	(Sodanil et al., 2015)
Assumption Documentation	(BACA; CARLSSON, 2011), (SEVEN..., 2019), (COMMON..., 2019), (Oyetoyan; Cruzes; Jaatun, 2016)
External Review	(BACA; CARLSSON, 2011), (SEVEN..., 2019), (Oyetoyan; Cruzes; Jaatun, 2016), (MCGRAW, 2004)
Agree on definitions	(BACA; CARLSSON, 2011), (COMMON..., 2019)
Critical Assets	(BACA; CARLSSON, 2011), (COMMON..., 2019), (Oyetoyan; Cruzes; Jaatun, 2016)

UML Sec	(BACA; CARLSSON, 2011), (COMMON..., 2019), (Oyetoyan; Cruzes; Jaatun, 2016)
Requirements Inspection	(BACA; CARLSSON, 2011), (COMMON..., 2019), (Oyetoyan; Cruzes; Jaatun, 2016)
Repository Improvement	(BACA; CARLSSON, 2011), (COMMON..., 2019) (Oyetoyan; Cruzes; Jaatun, 2016)
Cost Analyses	(BACA; CARLSSON, 2011) (Oyetoyan; Cruzes; Jaatun, 2016)
Final Secure Review	(BACA; CARLSSON, 2011)
Conduct Final Security Reviews	(Sodanil et al., 2015)
Abuse Cases	(BACA; CARLSSON, 2011), (SEVEN..., 2019), (Oyetoyan; Cruzes; Jaatun, 2016), (DAUD, 2010), (MCGRAW, 2004), (FRANQUEIRA et al., 2011)
Security Master	(Barbosa; Sampaio, 2015), (Terpstra; Daneva; Wang, 2017), (LIAQAT et al., 2015), (AZHAM; GHANI; ITHNIN, 2011)

B CATALOGO DE PRÁTICAS

Este apêndice apresenta o catalogo, de todas as práticas, consideradas durante essa pesquisa.

Prática	Nomes na Literatura	Metodologia	Referencias
Provide Security Training	*Security Training, Provide Training, Core Security Training, Provide Security Training, Security Training, Security Awareness Training, Education, Initial Education and training, Security Education awareness, Security training, Institute Awareness Programs*	Ágil e Clássica	(Oueslati; Rahman; Othmane, 2015), (KANNIAH; MAHRIN, 2016), (MICROSOFT..., 2019), (Sodanil et al., 2015), (Villamizar et al., 2018), (NICOLAYSEN et al., 2010), (Barbosa; Sampaio, 2015), (Oyetoyan; Cruzes; Jaatun, 2016), (Bansal; Jolly, 2014), (SINGHAL et al., 2012), (BARTSCH, 2011), (CLASP, 2019)
Security Backlog	*Security Backlog*	Ágil	(Barbosa; Sampaio, 2015), (Villamizar et al., 2018), (NICOLAYSEN et al., 2010), (AZHAM; GHANI; ITHNIN, 2011)

Evil User Stories	*Evil User Stories, Abuser Stories, Misuse Storires*	Ágil	(Barbosa; Sampaio, 2015), (Villamizar et al., 2018), (NICOLAYSEN et al., 2010), (BOSTRÖM et al., 2006), (BOWEN et al., 2014), (CHÓLIZ; VILAS; MOREIRA, 2015), (SIPONEN; BASKERVILLE; KUIVALAINEN, 2005), (BARTSCH, 2011), (WILLIAMS; MENEELY; SHIPLEY, 2010), (MUNETOH; YOSHIOKA, 2013), (DAUD, 2010), (OTHMANE; ANGIN; BHARGAVA, 2014)
Protection Poker	*Protection Poker*	Ágil	(Barbosa; Sampaio, 2015), (Villamizar et al., 2018), (BOWEN et al., 2014), (WILLIAMS; MENEELY; SHIPLEY, 2010)
Incentive Security Research	*Research Vulnerabilities*	Ágil e Clássica	(Barbosa; Sampaio, 2015), (Villamizar et al., 2018), (WANG; GUPTA; NIU, 2018), (FRANQUEIRA et al., 2011)

Security Code Analysis	*Security Verifications, Perform Static Analysis Security Testing (SAST), Perform Dynamic Analysis Security Testing (DAST), Perform Static Analysis (MSDL Based), Perform Dynamic Analysis (MSDL Based), Perform Fuzz Testing, Static Code Analysis, Dynamic Code Analysis, Fuzzy Testing, Static Code Analyses, Automating Security Checks, Perform application assessments.*	Ágil e Clássica	(Barbosa; Sampaio, 2015), (MICROSOFT..., 2019), (Sodanil et al., 2015), (BACA; CARLSSON, 2011), (NICOLAYSEN et al., 2010), (Oueslati; Rahman; Othmane, 2015), (SEVEN..., 2019), (KANNIAH; MAHRIN, 2016), (Oyetoyan; Cruzes; Jaatun, 2016), (Bansal; Jolly, 2014), (BOSTRÖM et al., 2006), (CHÓLIZ; VILAS; MOREIRA, 2015), (SINGHAL et al., 2012), (MUNETOH; YOSHIOKA, 2013), (MCGRAW, 2004), (BODDEN, 2018), (FRANQUEIRA et al., 2011), (KANNIAH; MAHRIN, 2016), (CLASP, 2019)

Secutiry Specialized Testing	*Penetration Testing, Perform Penetration Testing, Red Team Testing, Risk Based Testing, Security Testing*	Ágil e Clássica	(Oueslati; Rahman; Othmane, 2015), (KANNIAH; MAHRIN, 2016), (BACA; CARLSSON, 2011), (NICOLAYSEN et al., 2010), (Oyetoyan; Cruzes; Jaatun, 2016), (Bansal; Jolly, 2014), (CHÓLIZ; VILAS; MOREIRA, 2015), (SINGHAL et al., 2012), (DAUD, 2010), (MCGRAW, 2004), (FRANQUEIRA et al., 2011) , (SEVEN..., 2019), (MICROSOFT..., 2019), (CLASP, 2019)

Define Security Requirements	*Capture security requirements, Define Security Requirements, Establish Security Requirements, Security Requirements, Security Requirements Analysis, Assess Security Requirements*	Ágil e Clássica	(KANNIAH; MAHRIN, 2016), (CLASP, 2019), (MICROSOFT..., 2019), (Sodanil et al., 2015), (BACA; CARLSSON, 2011), (SEVEN..., 2019), (COMMON..., 2019), (Villamizar et al., 2018), (NICOLAYSEN et al., 2010), (Oyetoyan; Cruzes; Jaatun, 2016), (Bansal; Jolly, 2014), (BOSTRÖM et al., 2006), (BOWEN et al., 2014), (SIPONEN; BASKERVILLE; KUIVALAINEN, 2005), (SINGHAL et al., 2012), (WANG; GUPTA; NIU, 2018), (BARTSCH, 2011), (MUNETOH; YOSHIOKA, 2013), (LIAQAT et al., 2015), (DAUD, 2010), (MCGRAW, 2004), (FRANQUEIRA et al., 2011), (OTHMANE; ANGIN; BHARGAVA, 2014)

Security Code Review	*Code Review, Diff Review, Reviewing Security Code*	Ágil e Clássica	(BACA; CARLSSON, 2011), (Oueslati; Rahman; Othmane, 2015), (Terpstra; Daneva; Wang, 2017), (Villamizar et al., 2018), (Oyetoyan; Cruzes; Jaatun, 2016), (CHÓLIZ; VILAS; MOREIRA, 2015)
Security Automated Testing	*Security-oriented tdd: Security tests, Security Unit Testing*	Ágil	(NICOLAYSEN et al., 2010), (SINGHAL et al., 2012), (BARTSCH, 2011), (MUNETOH; YOSHIOKA, 2013), (DAUD, 2010), (FRANQUEIRA et al., 2011)
Security Coding Rules	*Secure Code, Deprecate Unsafe Functions, Coding Rules, Implement secure development practices*	Ágil e Clássica	(NICOLAYSEN et al., 2010), (Sodanil et al., 2015), (BACA; CARLSSON, 2011), CLASP, (NICOLAYSEN et al., 2010), (Oyetoyan; Cruzes; Jaatun, 2016), (Bansal; Jolly, 2014), (BOSTRÖM et al., 2006), (BARTSCH, 2011), (MUNETOH; YOSHIOKA, 2013), (DAUD, 2010), (KANNIAH; MAHRIN, 2016)

Perform Risk Analysis	*Risk Analyses, Assess Security Risk*	Ágil e Clássica	(BACA; CARLSSON, 2011), (SEVEN..., 2019), (COMMON..., 2019), (NICOLAYSEN et al., 2010), (Terpstra; Daneva; Wang, 2017)
Pair Programming	*Pair Programming*	Ágil	(BACA; CARLSSON, 2011), (Oyetoyan; Cruzes; Jaatun, 2016), (BOWEN et al., 2014)
Establish Security Design Requirements	*Establish Design Requirements, Design Requirements, Secure Design, Security Design*	Ágil e Clássica	(MICROSOFT..., 2019), (Sodanil et al., 2015), (BACA; CARLSSON, 2011), (Oyetoyan; Cruzes; Jaatun, 2016), (NICOLAYSEN et al., 2010), (BOSTRÖM et al., 2006)
Identify Vulnerabilities	*Identify Vulnerabilities, Vulnerability Analysis*	Ágil	(Oueslati; Rahman; Othmane, 2015), (Villamizar et al., 2018)
Verify and Validate the Security	*Verify and Validate the Security*	Ágil	(Oueslati; Rahman; Othmane, 2015)
Build Vulnerability Remediation Procedures	*Build Vulnerability Remediation Procedures*	Clássico	(KANNIAH; MAHRIN, 2016), (CLASP, 2019)
Define and Use Cryptography Standards	*Define and Use Cryptography Standards*	Ágil e Clássica	(MICROSOFT..., 2019)

Use Approved Tools	*Use Approved Tools, Security Tools*	Ágil e Clássica	(MICROSOFT..., 2019), (Sodanil et al., 2015), (BACA; CARLSSON, 2011)
Prioritizing Security Risk	*Prioritizing Security Risk*	Ágil	(Terpstra; Daneva; Wang, 2017), (Villamizar et al., 2018), (BARTSCH, 2011)
Incident Plan	*Incident Plan, Establish a Standard Incident Response Process, Create an Incident Response Plan, Execute Incident Response Plan, Incident Response Plan*	Ágil e Clássica	(Barbosa; Sampaio, 2015), (MICROSOFT..., 2019), (Sodanil et al., 2015), (BACA; CARLSSON, 2011), (Oyetoyan; Cruzes; Jaatun, 2016), (SINGHAL et al., 2012)
Specifie Security Policies	*Specifie Security Policies, Security Specification*	Ágil e Clássica	(Oueslati; Rahman; Othmane, 2015), (KANNIAH; MAHRIN, 2016), (ANIS et al., 2018)

Threat Modelling	*Threat Modelling, Perform Threat Modeling, Use Threat Modeling*	Ágil e Clássica	(KANNIAH; MAHRIN, 2016), (MICROSOFT..., 2019), (Sodanil et al., 2015), (BACA; CARLSSON, 2011), (Bansal; Jolly, 2014), (BAIZE, 2012), (BOWEN et al., 2014), (CHÓLIZ; VILAS; MOREIRA, 2015), (KERAMATI; MIRIANHOSSEINABADI, 2008), (SINGHAL et al., 2012), (WILLIAMS; MENEELY; SHIPLEY, 2010), (LIAQAT et al., 2015), (AZHAM; GHANI; ITHNIN, 2011)
Define and Monitor Metrics	*Define and Monitor Metrics, Define Metrics and Compliance Reporting, Create Quality Gates/Bugs Bars, Quality Gates*	Ágil e Clássica	(KANNIAH; MAHRIN, 2016), (CLASP, 2019), (Sodanil et al., 2015), (BACA; CARLSSON, 2011), (Oyetoyan; Cruzes; Jaatun, 2016), (MICROSOFT..., 2019)
Publish Operational Security Guidelines	*Publish Operational Security Guidelines*	Ágil e Clássica	(KANNIAH; MAHRIN, 2016), (CLASP, 2019)

Manage the Security Risk of Using Third-Party Components	*Manage the Security Risk of Using Third-Party Components*	Ágil e Clássica	(MICROSOFT…, 2019)
Role Matrix	*Role Matrix*	Ágil e Clássica	(BACA; CARLSSON, 2011)
Countermeasure Graphs	*Countermeasures Graphs*	Ágil	(BACA; CARLSSON, 2011), (Oyetoyan; Cruzes; Jaatun, 2016), (BOSTRÖM et al., 2006), (SIPONEN; BASKERVILLE; KUIVALAINEN, 2005)
Review of Security Specification	*Review of Security Specification*	Clássica	(KANNIAH; MAHRIN, 2016)
Perform Security and Policy Risk Assessment	*Perform Security and Policy Risk Assessment*	Clássica	(Sodanil et al., 2015)
Perform Attack Surface Analysis/Reduction	*Perform Attack Surface Analysis/Reduction, Conduct Attack Surface Review, Attack Surface Reduction, Attack Surface Analysis*	Clássico	(Sodanil et al., 2015), (BACA; CARLSSON, 2011), (Oyetoyan; Cruzes; Jaatun, 2016)
Certify Release and Archive	*Certify Release and Archive*	Clássico	(Sodanil et al., 2015)

Assumption Documentation	*Assumption Documentation*	Clássico	(BACA; CARLSSON, 2011), (SEVEN..., 2019), (COMMON..., 2019), (Oyetoyan; Cruzes; Jaatun, 2016)
External Review	*External Review*	Clássico	(BACA; CARLSSON, 2011), (SEVEN..., 2019), (Oyetoyan; Cruzes; Jaatun, 2016), (MCGRAW, 2004)
Agree on definitions	*Agree on definitions*	Clássico	(BACA; CARLSSON, 2011), (COMMON..., 2019)
Critical Assets	*Critical Assets*	Clássico	(BACA; CARLSSON, 2011), (COMMON..., 2019), (Oyetoyan; Cruzes; Jaatun, 2016)
UML Sec	*UML Sec*	Clássico	(BACA; CARLSSON, 2011), (COMMON..., 2019), (Oyetoyan; Cruzes; Jaatun, 2016)
Requirements Inspection	*Requirements Inspection*	Clássico	(BACA; CARLSSON, 2011), (COMMON..., 2019), (Oyetoyan; Cruzes; Jaatun, 2016)
Repository Improvment	*Repository Improvement*	Clássico	(BACA; CARLSSON, 2011), (COMMON..., 2019), (Oyetoyan; Cruzes; Jaatun, 2016)

Cost Analyses	*Cost Analyses*	Ágil e Clássica	(BACA; CARLSSON, 2011), (Oyetoyan; Cruzes; Jaatun, 2016)
Final Secutiry Review	*Final Secure Review, Conduct Final Security Reviews*	Clássico	(BACA; CARLSSON, 2011), (Oueslati; Rahman; Othmane, 2015), (Sodanil et al., 2015), (Oyetoyan; Cruzes; Jaatun, 2016)
Abuse Cases	*Abuse Cases*	Clássica	(BACA; CARLSSON, 2011), (SEVEN..., 2019), (Oyetoyan; Cruzes; Jaatun, 2016), (DAUD, 2010), (MCGRAW, 2004), (FRANQUEIRA et al., 2011)
Security Master	*Security Master*	Ágil	(Barbosa; Sampaio, 2015), (Terpstra; Daneva; Wang, 2017), (LIAQAT et al., 2015), (AZHAM; GHANI; ITHNIN, 2011)

C SELEÇÃO DE PRÁTICAS JUNTO AO TIME ÁGIL

Este apêndice apresenta os resultados obtidos para os questionamentos abaixo, durante a seção de avaliação das práticas, junto a um time de desenvolvimento ágil.

1. Essa práticas podem ser relacionadas com metodologias ágeis?
2. É possível agrupar 1 ou mais dessas práticas?
3. Qual a importância da aplicação da prática em relação a desenvolvimento de software seguro?
4. Em qual fase do desenvolvimento esta práticas deve estar incorporada?

Prática	Q1	Q2	Q3	Q4
Risk Analysis	Sim	Não	Alta	Planejamento
Pair Programming	Sim	Não	Média	Implementação
Establish Security Design Requirements	Sim	Não	Alta	Planejamento
Use Approved Tools	SIm	Não	Baixa	Planejamento, Implementação
Prioritizing Security Risk	Sim	Incorporavél em Risk Analysis e ProtectionPoker	Baixa	Requisitos e Planejamento
Incident Plan	Sim	Não	Baixa	Entrega
Specifie Security Policies	Não	Não	Baixa	-
Threat Modelling	Não	Não	Alta	Planejamento
Define and Monitor Metrics	Sim	Não	Baixa	Planejamento
Countermeasure Graphs	Sim	Não	Baixa	Planejamento e Requisitos

Provide Security Training	Sim	Não	Alta	Treinamento
Security Backlog	Sim	Não	Média	Requisitos
Evil User Stories	Sim	Não	Média	Requisitos
Protection Poker	Sim	Não	Média	Planejamento
Incentive Security Research	Sim	Não	Média	Treinamento
Security Code Analysis	Sim	Não	Alta	Implementação e Verificação
Secutiry Specialized Testing	Sim	Não	Alta	Verificação
Define Security Requirements	Sim	Não	Alta	Requisitos
Security Code Review	Sim	Não	Alta	Implementação e Verificação
Security Automated Testing	Sim	Não	Alta	Implementação e Verificação
Security Coding Rules	Sim	Não	Alta	Implementação e Verificação

D PRÁTICAS NÃO INCLUSAS NAS POLÍTICAS

Este Apêndice apresenta as práticas que não foram inclusas nas políticas propostas e motivação para tal.

Prática	**Motivo da remoção**
Identify Vulnerabilities	Referências Insuficientes.
Verify and Validate the Security	Referências Insuficientes.
Build Vulnerability Remediation Procedures	Referências insuficientes e Prática exclusiva das metodologias clássicas..
Define and Use Cryptography Standards	Referências Insuficientes.
Use Approved Tools	De acordo com a análise, do time de desenvolvimento consultado, pode não trazer benefícios suficientes para sua inclusão.
Prioritizing Security Risk	De acordo com a análise, do time de desenvolvimento consultado, as práticas*Protection Poker* e *Risk Analysis* podem cobrir a ausência dessa prática.
Incident Plan	De acordo com a análise, do time de desenvolvimento consultado, não depende exclusivamente do time ágil, além disso pode não trazer tantos benefícios assim.
Specifie Security Policies	De acordo com a análise, do time de desenvolvimento consultado, o foco em documentação, dessa prática, é um empecilho para times ágeis, além disso, seu contexto pode ser mais institucional e não diretamente ligado ao processo de desenvolvimento.

Threat Modelling	De acordo com a análise, do time de desenvolvimento consultado, existem diversos benefícios associados, porém, todos apresentam a visão de que essa práticas precisa de muitos ajustes para se adequar ao mundo ágil. Além disso, esses ajustes, podem vir a causar uma redução desses benefícios.
Define and Monitor Metrics	De acordo com a análise, do time de desenvolvimento consultado, pode não trazer benefícios suficientes para sua inclusão. Além disso, a maioria dos envolvidos não apresentou nenhum método para implementação da mesma.
Publish Operational Security Guidelines	Referências Insuficientes.
Manage the Security Risk of Using Third-Party Components	Referências Insuficientes.
Role Matrix	Referências Insuficientes.
Countermeasure Graphs	De acordo com a análise, do time de desenvolvimento consultado, pode não trazer benefícios suficientes para sua inclusão.
Review of Security Specification	Referências insuficientes e Prática exclusiva das metodologias clássicas..
Perform Security and Policy Risk Assessment	Referências insuficientes e Prática exclusiva das metodologias clássicas..
Perform Attack Surface Analysis/Reduction	Prática exclusiva das metodologias clássicas..
Certify Release and Archive	Referências insuficientes e Prática exclusiva das metodologias clássicas..
Assumption Documentation	Prática exclusiva das metodologias clássicas..
External Review	Prática exclusiva das metodologias clássicas..
Agree on definitions	Referências insuficientes e Prática exclusiva das metodologias clássicas..

Critical Assets	Prática exclusiva das metodologias clássicas..
UML Sec	Prática exclusiva das metodologias clássicas..
Requirements Inspection	Prática exclusiva das metodologias clássicas..
Repository Improvment	Prática exclusiva das metodologias clássicas..
Cost Analyses	Referências insuficientes e Prática exclusiva das metodologias clássicas.
Final Security Review	Prática exclusiva das metodologias clássicas..
Abuse Cases	Prática exclusiva das metodologias clássicas..
Security Master	Trata-se de um novo papel, não de uma Pratica