INSTITUTO FEDERAL DE SÃO PAULO - IFSP

ÁREA DE INFORMÁTICA

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - ADS

RENÊ ESTEVES MARIA

LUIZ ANTONIO RODRIGUES JUNIOR

TÉCNICAS DE SEGURANÇA APLICADAS NO SCRUM

TRABALHO DE CONCLUSÃO DE CURSO - TCC

CARAGUATATUBA

2013

RENÊ ESTEVES MARIA

LUIZ ANTONIO RODRIGUES JUNIOR

TÉCNICAS DE SEGURANÇA APLICADAS NO SCRUM

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Tecnólogo, da Área de Informática, do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo.

Orientador:

Prof. M.e Nelson Alves Pinto

CARAGUATATUBA

2013

M332t MARIA, Renê Esteves.

Técnicas de Segurança Aplicadas no Scrum

/ Renê Esteves Maria; Luiz Antonio Rodrigues Junior. Caraguatatuba, SP: 2013.

62f.

Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas)–Instituto Federal de São Paulo, Caraguatatuba, 2013.

1. Desenvolvimento de software. I. Rodrigues Junior, Luiz Antonio. II. Título.

CDD - 005:1

TERMO DE APROVAÇÃO

TÉCNICAS DE SEGURANÇA APLICADAS NO SCRUM

por

RENÊ ESTEVES MARIA
LUIZ ANTONIO RODRIGUES JUNIOR

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 17 de Dezembro de 2013 como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas (ADS). Os candidatos foram arguidos pela Banca Examinadora composta pelos professores abaixo assinados, a qual após deliberação, considerou o trabalho aprovado.

_____
Prof. M.e Nelson Alves Pinto
Prof. Orientador

_____
Prof. Dr. Lineu Fernando Stege Mialaret
Presidente

_____
Prof. Esp. Renan Cavichi de Freitas
Membro

Este trabalho é especialmente dedicado à memória do meu avô Manoel Maria, pois todo seu sacrifício valeu a pena.

Rene Esteves Maria

AGRADECIMENTOS

Será difícil agradecer todas as pessoas que fizeram parte dessa importante fase de nossas vidas. Por isso, pedimos desculpas á todos que não estão presentes entre essas palavras, mas podem ter a certeza que fazem parte da nossa história e por isso sempre seremos gratos.

Agradecemos ao orientador professor Nelson Alves Pinto, por aceitar e ter paciência em nos guiar por todos os caminhos que levaram a construção deste trabalho.

É importante registrar o apoio de nossas famílias durante o decorrer do curso, pois acreditamos que sem o apoio deles seria muito difícil enfrentar todos os desafios encontrados.

Aos professores de todas as áreas que nos acompanharam durante toda graduação, e especialmente aos professores que compõem a área de informática.

Por fim, a todos os que por algum motivo contribuíram para a realização desta pesquisa.

AGRADECIMENTOS – RENE ESTEVES MARIA

Primeiramente a Deus, por todas as bençõe recebidas e por iluminar sempre meu caminho.

À minha mãe, Adriana Esteves Maria, por ser sempre amável, carinhosa e compreensível comigo. Também, por me ensinar, com muita paciência, a cuidar da casa toda.

Ao meu pai, Reinaldo Alfredo Maria, posso dizer que sempre será um exemplo a ser seguido por mim, por tudo que ele é. Sempre contei com seu apoio e seus conselhos durante minha vida.

À minha vó, Maria de Lourdes, uma das principais incentivadoras da minha vida acadêmica, que sempre me tratou com muito carinho.

Aos meus irmãos, Renato Esteves Maria e Angélica Esteves Maria, que sempre foram meus melhores amigos e sempre contei com nossa união para ganhar forças.

Aos meus familiares, que sempre torceram por mim e acreditaram que era possível alcançar este sonho.

Ao meu orientador, Nelson Alves Pinto, pela paciência na orientação e oportunidades no decorrer do curso, que foram fundamentais para minha formação.

Ao meu parceiro, Luiz Antonio Rodrigues Junior, que topou encarar esta jornada comigo, suas contribuições tornaram possível à construção dessa pesquisa.

Finalmente, quero registrar meus agradecimentos a todas as pessoas que passaram pela minha vida, neste período, cada uma teve seu papel e sabe o quanto foi importante para mim, graças a todos foi possível concluir essa pesquisa. Obrigado!

Uma vida sem desafios não vale a pena
ser vivida.

Sócrates

RESUMO

A utilização do método ágil Scrum para desenvolvimento de produtos oferece como principais benefícios aos seus utilizadores o aceleramento do processo e recursos para lidar com a instabilidade do ambiente tecnológico. O ato de privilegiar a rapidez e suportar requisitos voláteis resulta em um produto com maior valor agregado, entretanto pode levar o time a não lidar de forma adequada com um aspecto crítico de todo sistema que é a segurança da informação. Considerando que os ataques tem se tornado mais sofisticados e que mesmo sistemas mais simples podem ser alvos em potencial, se torna imprescindível que a segurança do *software* seja tratada de forma mais elaborada dentro da própria metodologia, de modo a fazer parte do processo. Com o intuito de aumentar a qualidade e confiabilidade dos sistemas é proposto um incremento ao método Scrum, denominado ScrumS, que agrega técnicas específicas de segurança do processo de Análise de Riscos e Testes de Segurança ao ciclo de vida para auxiliar na identificação e correção de vulnerabilidades.

Palavras-chave: Scrum. Segurança. Análise de Riscos. Desenvolvimento Ágil.

ABSTRACT

The agile method Scrum when used for software development offers as major benefits to its users the process acceleration and means to deal with the instability of the technological environment. The practice of prioritize the speed and support volatile requirements results in a product with higher value but at the same time can lead the team to not deal adequately with a critical aspect of all systems that is the security of information. Whereas the attacks have become more sophisticated and even simpler systems can be potential targets, becomes indispensable that the software security be treated in a more elaborated way in the methodology itself, as a process part. In order to increase systems quality and reliability is proposed an increment to the method Scrum, called ScrumS, which adds particular security techniques derived from the Risk Analysis and Security Testing to the lifecycle in order to assist the identification and correction of vulnerabilities.

Keywords: Scrum. Security. Risk Analysis. Agile Development

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS, SIGLAS E ACRÔNIMOS

AES	*Advanced Encryption Standard*
BC	*Burndown Chart*
CPA	Comissão Própria de Avaliação
DOD	*Definition of Done*
IFSP	Instituto Federal de São Paulo
PB	*Product Backlog*
PG	*Pregame*
PO	*Product Owner*
SB	*Sprint Backlog*
PO	*Product Owner*
SDL	*Secure Development Lifecycle*
SINAES	Sistema Nacional de Avaliação da Educação Superior
SO	Sistema Operacional
US	*User Story*
USs	*User Stories*
V&M	Viega e McGraw

SUMÁRIO

INTRODUÇÃO

O Scrum se tornou uma dos métodos mais utilizados no desenvolvimento ágil de *software* [21]. A simplicidade de seu funcionamento contribuiu para isso, mas o ponto principal está relacionado aos benefícios que costuma trazer aos seus praticantes. O método considera o cliente como um personagem fundamental, buscando reduzir o tempo do produto até o mercado e também oferece uma estrutura capaz de auxiliar o time a lidar com requisitos voláteis, comuns em métodos iterativos e incrementais [14].

No Scrum o cliente possui um representante que é responsável por conhecer o sistema e entender as suas regras de negócio, além de documentar todos os requisitos para que posteriormente possam ser codificados pelos desenvolvedores. Contudo, alguns requisitos do *software* não são facilmente detectados, tais como requisitos de segurança e confiabilidade [1]. Este foco nas funcionalidades do sistema é transferido aos desenvolvedores após a fase de documentação sem que haja um evento intermediário para extração de requisitos não-funcionais.

A fragilidade citada causa a exposição de vulnerabilidades no sistema que poderão ser exploradas por atacantes quando em ambiente de produção. Somado a isso tem-se que os métodos para ganho de acesso estão cada vez mais sofisticados e podem ser realizados de forma massiva sem que se possa determinar a motivação do ataque [12]. Assim, se faz necessário uma preocupação diferencial com os aspectos de segurança do *software* e esta deve ser considerada desde o início do processo de desenvolvimento, para abranger todo o ciclo de vida do *software* é necessário um planejamento avançado e cauteloso [11].

O trabalho em questão propõe um modelo denominado ScrumS onde se pretende que seja utilizado para desenvolvimento de sistemas em que a segurança da informação é um diferencial importante. Ele agrega aos eventos do Scrum à fase de Projeto Seguro proposta por Viega e Mcgraw (V&G) [22] e a Análise de Riscos de acordo com o Modelo de Processos para Testes de Segurança com Abordagem Orientada a Riscos [11].

A aplicação do Projeto Seguro dentro do Scrum é realizado durante a fase de *Pregame* (PG), que originalmente tem como propósito solucionar todos os problemas relacionados ao projeto antes dele começar [16]. Com o Projeto Seguro será possível definir todas as diretrizes relacionadas às tecnologias, processos e políticas de segurança.

A Análise de Riscos é trabalhada com o intuito de criar um inventário com todos os riscos que cada funcionalidade do sistema poderá se deparar e com base neles propor controles de segurança que devem ser codificados durante a etapa de desenvolvimento, mitigando ou extinguindo eventuais falhas.

É essencial que os esforços desprendidos para criação da documentação não comprometam o fluxo de trabalho original do Scrum. Também se espera que esta documentação possa ser refinada e atualizada, além de ser possível reaproveitá-la em casos específicos.

Pretende-se obter com a utilização do modelo proposto aplicações com um diferencial em segurança e qualidade, guiando o desenvolvimento pelas funcionalidades e as possíveis vulnerabilidades que as mesmas podem vir a possuir.

1.1 CONTEXTO DA PESQUISA

O método Scrum (assim como outros métodos ágeis) é criticado por negligenciar práticas de segurança no desenvolvimento de *software* [8]. O efeito disso são vulnerabilidades que podem ser exploradas por um atacante [1].

Este trabalho tem como objetivo agregar técnicas e processos específicos de segurança ao método Scrum, de modo que o método passe a oferecer um diferencial no aspecto de segurança do sistema sem que haja grande comprometimento dos princípios ágeis.

1.2 OBJETIVOS

O principal objetivo deste trabalho é acrescentar ao método Scrum artefatos relacionados aos aspectos da segurança do *software*, para que seja possível realizar uma investigação com relação à possibilidade de usá-los sem comprometer o ciclo original do método.

Como ponto inicial toma-se por base o Modelo de Processos para Testes de Segurança com Abordagem Orientado a Riscos [11], cuja proposta é a extração de Testes de Segurança a partir dos riscos detectados. Contudo, o trabalho não oferece especificações para utilização em métodos ágeis, tornando necessário o mapeamento de suas fases mais importantes da Análise de Riscos para os eventos do Scrum.

A documentação de requisitos em metodologias tradicionais costuma ser feita com Diagramas de Caso de Uso. Uma das técnicas utilizadas para descoberta de ameaças é elaboração de Casos de Mal uso, que adicionam novas ações e atores com o intuito de perpetuar um ataque. É comum em projetos Scrum que a documentação de requisitos seja feita com *User Stories* (USs) e para que a descoberta das ameaças se mantenha ágil serão propostas Histórias do Usuário de Segurança.

Um dos artefatos do Scrum conhecidos como DOD (Definição do Pronto), propõe a validação do que foi desenvolvido durante uma iteração, porém, com a nova documentação esta etapa deixa de ser trivial, visto que todos os controles de segurança precisam ser testados e aprovados. A criação de um mecanismo automatizado de validação viria a reduzir a propensão a erros neste evento.

Com a utilização do processo de Análise de Riscos no Scrum, pretende-se detectar antecipadamente possíveis vulnerabilidades que as funcionalidades possam vir a expor. Espera-se obter como resultado tarefas específicas de segurança para cada funcionalidade, ou seja, controles de segurança para cada risco descoberto.

1.3 ORGANIZAÇÃO DO TEXTO

No Capítulo 2 serão apresentados os conceitos referentes ao método de desenvolvimento ágil Scrum, à Análise de Riscos e aos ciclos de desenvolvimento seguro de *software*. No Capítulo 3 é apresentado o incremento ao Scrum, denominado ScrumS, para a melhoria da qualidade e segurança final do *software*. No Capítulo 4 é apresentado um estudo de caso onde foi aplicado o modelo desenvolvido. No Capítulo 5 são apresentados os resultados obtidos com o estudo de caso. No Capítulo 6 são apresentadas as conclusões finais, bem como limitações e trabalhos futuros.

CONTEXTUALIZAÇÃO

Nesse capítulo são apresentados os conceitos envolvidos no processo de desenvolvimento utilizando o método Scrum e os conceitos da área de segurança, tais como ciclos de desenvolvimento seguros, Análise de Riscos, processo de modelagem dos Casos de Mal Uso e a extração das Árvores de Ataques. Ao final do capítulo é realizada uma breve análise do Modelo de Processos para Testes de Segurança com Abordagem Orientado a Riscos, e dos métodos ágeis seguros *S-Scrum*, *Security Backlog* e *Agile SDL* que representam o estado atual da pesquisa.

2.1 MÉTODOS ÁGEIS

Métodos de desenvolvimento ágeis buscam acelerar o processo de desenvolvimento de sistemas. São desenvolvidas versões mínimas de um sistema (divisão do problema em partes menores) de forma a realizar entregas regulares do produto [15]. As funcionalidades são integradas através de um processo iterativo baseado na interação com o cliente ao longo do ciclo de desenvolvimento.

A origem deste tipo de método está diretamente ligada à instabilidade do ambiente tecnológico, onde integrantes podem deixar de fazer parte da equipe de desenvolvimento, novas versões de ferramentas podem ser lançadas corrigindo *bugs* ou vulnerabilidades das versões anteriores, requisitos podem ser mal compreendidos e outras inúmeras variáveis exclusivas de cada projeto. A principal vantagem da utilização destes métodos está relacionada à administração dos requisitos voláteis [5].

Em um método ágil o cliente pode ser considerado o piloto do projeto e obtém rapidamente uma primeira versão do seu *software* e assim pode auxiliar os envolvidos a compreender o que ele precisa e informar sobre requisitos mal definidos.

2.2 SCRUM

O Scrum é um método de desenvolvimento ágil para gestão de projetos amplamente utilizada [21]. Com sua utilização é possível manter produtos complexos onde indivíduos trabalham em conjunto para resolver problemas.

De forma produtiva são entregues produtos de alto valor através de uma abordagem cíclica em que partes pequenas são desenvolvidas e entregues ao cliente que por sua vez orienta o time em relação as suas perspectivas, o que aprimora a previsibilidade e controle de riscos [15].

O processo de desenvolvimento é iniciado com o levantamento e documentação, junto ao cliente, das funcionalidades do sistema, comumente armazenadas no formato de *User Story* (US) que em conjunto compõem o *Product Backlog* (PB). Devido ao Scrum permitir volatibilidade de requisitos é possível adicionar, alterar ou remover estes desejos (funcionalidades do sistema) de acordo com as necessidades do cliente [5].

Posteriormente são separados alguns itens do PB para compor o *Sprint Backlog* (SB), que é a lista das funcionalidades que serão desenvolvidas durante uma iteração. Esta iteração pode durar de 2 a 4 semanas e é conhecida como *Sprint*. O final de cada *Sprint* é marcado com a entrega de uma parte do *software* em funcionamento [14].

2.2.1 VISÃO GERAL

O método Scrum consiste em times associados a papéis, eventos e artefatos que administram a relação e interação entre os seus praticantes. Pregando que todos os envolvidos devem estar unidos para atingir um objetivo em comum, sempre somando os esforços para o sucesso do projeto [15].

2.2.1.1 Fases do Scrum: Pregame

De acordo com Schwaber [16], o método Scrum segue três fases na sua aplicação, sendo as seguintes: *Pregame* (PG); *Game*; *PostGame.*

A fase de PG é responsável por a partir de um PB conhecido, definir todas as diretrizes de estrutura do time em relação ao projeto e planejar seus entregáveis. A fase de *Game* consiste no desenvolvimento das funcionalidades em *Sprints.* E durante a fase de *Postgame é* disponibilizado o entregável, junto a sua documentação final.

2.2.2 Teoria

O método se fundamenta em teorias empíricas de controle de processo. O empirismo afirma que o conhecimento vem da experiência e a partir disso são tomadas as decisões com base no que é conhecido [15].

O controle de processo empírico acontece dentro dos três pilares do Scrum: transparência, inspeção e adaptação. A transparência auxilia o time a sincronizar o entendimento do projeto, a inspeção por sua vez garante a manutenção dos valores dos artefatos Scrum e a adaptação prevê que o time deve estar sempre pronto a responder as mudanças ocorridas dentro do projeto [15].

2.2.3 Papéis

O método Scrum dispõe de três papéis para seus utilizadores. Estes papéis são definidos como, *Product Owner* (PO), Time de Desenvolvimento e *Scrum Master* (SM). As próximas três seções descrevem cada papel.

2.2.3.1 PRODUCT OWNER

O *Product Owner*, também conhecido por PO, deve representar os objetivos de todos os interessados no projeto. Sua tarefa inicial é realizar o levantamento das funcionalidades gerais do *software* e logo em seguida criar o PB. Deve acompanhar todo processo de desenvolvimento a fim de priorizar sempre a funcionalidade que agrega maior valor ao cliente [14].

2.2.3.2 TIME DE DESENVOLVIMENTO

O time de desenvolvimento ou equipe é formado por no mínimo três e no máximo nove integrantes [15]. Este time deve ser auto-gerenciável, auto-organizável e possuir membros multi-disciplinares. Ao longo do processo o time de desenvolvimento deve transformar os requisitos levantados em um incremento funcional ao final de cada iteração [14].

2.2.3.3 SCRUM MASTER

O *Scrum Master* (SM) é responsável por garantir o processo do Scrum, fazendo com que todos sigam as boas práticas e não quebrem os princípios. Além disso, ele também é o responsável por remover quaisquer impedimentos que dificultem a conclusão dos objetivos [14].

2.2.4 ARTEFATOS

Existem cinco artefatos que compõem o método Scrum a fim de maximizar a transparência em todo o processo [15]. Estes artefatos são descritos nas próximas cinco seções.

2.2.4.1 PRODUCT BACKLOG

O *Product Backlog* (PB) é o artefato que reúne a lista de todos os desejos do cliente, ou seja, funcionalidades do sistema. É essencial que as funcionalidades

ainda não implementadas sejam priorizadas de acordo com o seu valor para o cliente, desta forma podem ser avaliadas mais cedo já que estarão nos primeiros entregáveis [14].

Os incrementos entregues geram uma realimentação do cliente, podendo causar alterações durante todo o decorrer do projeto no PB [14].

2.2.4.2 SPRINT BACKLOG

O SB é um subconjunto de itens do PB. É composto por USs selecionadas para estar na *Sprint*, juntamente com o plano para entregar o incremento do produto e atingir o seu objetivo. O SB representa qual é a previsão do time de desenvolvimento sobre qual funcionalidade estará no próximo incremento [15].

2.2.4.3 BURNDOWN CHART

O *Burndown Chart* (BC) é um artefato utilizado para monitorar a quantidade de trabalho restante. Com ele é possível analisar se uma *Sprint* corre o risco de fracasso ou se o time está adiantado. O BC deve ser transparente para todas as partes interessadas [15].

Caso o SM julgue que não será possível obter sucesso na *Sprint*, ele pode junto ao time de desenvolvimento e ao PO cancelar a *Sprint*. Neste cenário as USs são devolvidas ao PB e as razões para o fracasso são discutidas [15].

No cenário em que uma *Sprint* é concluída antes do tempo estimado, são discutidas as possíveis causas e caso seja diagnosticado que o time está com uma fluência alta, é possível aumentar os pontos de USs que podem ser selecionadas (velocidade do time) [15].

2.2.4.4 INCREMENTO

Um incremento é a implementação de todos os itens do SB somados aos itens concluídos nas *Sprints* anteriores. Ao final da *Sprint* um incremento deve ser entregue, o que significa que deve estar na condição de ser utilizado, atender ao DOD e agregar novas funcionalidades em relação à *Sprint* anterior [15].

2.2.4.5 DEFINIÇÃO DE PRONTO

O DOD (*Definition Of Done*) é a orientação sobre o que pode ser considerado como um entregável ao final da *Sprint.* Embora isso varie significativamente para cada time Scrum, os integrantes devem ter um entendimento compartilhado do que significa o trabalho estar completo, assegurando a transparência [15].

2.2.5 EVENTOS

Existem cinco eventos que compõem o método Scrum a fim de criar uma rotina que sempre inspecione e adapte algo relacionado ao projeto [15]. Estes eventos são descritos nas próximas cinco seções.

2.2.5.1 SPRINT

A *Sprint* é um intervalo de dois até quatro semanas, durante o qual é desenvolvida uma versão incremental potencialmente utilizável do produto. Uma nova *Sprint* inicia imediatamente após a conclusão da Sprint anterior [15].

As *Sprints* são compostas por uma reunião de planejamento da *Sprint*, reuniões diárias, o trabalho de desenvolvimento, uma revisão da *Sprint* e a retrospectiva da *Sprint*. Durante a *Sprint* não são feitas mudanças que possam por em perigo o seu objetivo, as metas de qualidade não são alteradas e o escopo pode ser clarificado e renegociado entre o PO e a equipe [15].

2.2.5.2 Reunião de Planejamento da Sprint

O trabalho a ser realizado na *Sprint* é planejado na reunião de planejamento da *Sprint*. Este plano é criado com o trabalho colaborativo de todo o time Scrum e gera o SP [15]. As reuniões possuem uma métrica de oito horas para uma *Sprint* com duração de um mês. Durante a reunião é importante responder três perguntas:

- Qual é o objetivo da *Sprint*?
- O que pode ser entregue como resultado do incremento da próxima *Sprint*?
- Como o trabalho necessário para entregar o incremento será realizado?

2.2.5.3 Reunião Diária

A Reunião Diária do Scrum é um evento com duração de quinze minutos, para que a equipe possa sincronizar as atividades e criar um plano para as próximas vinte e quatro horas. Esta reunião é feita para inspecionar o trabalho desde a última reunião diária, e prever o trabalho que deverá ser feito antes da próxima reunião diária [15]. Essencial para manutenção da inspeção e transparência do processo, assim como a reunião de planejamento, existe três perguntas que devem ser feitas todos os dias:

- O que eu fiz ontem?
- O que eu farei hoje?
- Algum obstáculo me impediu de atingir o objetivo?

2.2.5.4 Revisão da Sprint

Ao terminar uma *Sprint* é realizada sua revisão, para inspecionar o incremento e adaptar o PB se necessário. Esta é uma reunião informal e a apresentação do incremento destina-se a obter aprovação do PO e obter seus comentários, a fim de otimizar o próximo valor entregável [15].

2.2.5.5 Retrospectiva da Sprint

A retrospectiva da *Sprint* é uma oportunidade para a equipe inspecionar a si própria e criar um plano para melhorias a serem aplicadas na próxima *Sprint*. A retrospectiva da *Sprint* ocorre depois da revisão da *Sprint* e antes da reunião de planejamento da próxima *Sprint* [15]. Esta é uma reunião que tem duração de três horas para uma *Sprint* de um mês.

2.3 Fase de Projeto Seguro

Uma atividade sugerida em ciclos de vida seguros [22] trata da escolha de linguagens, ferramentas, processos e políticas de segurança utilizadas para desenvolvimento do projeto. Com isso, se determinam diversas restrições e assertivas que o projeto terá de lidar. A tarefa consiste em se escolher todas as ferramentas e processos envolvidos no desenvolvimento, testes e ambiente de produção do *software* [11].

2.4 Análise e Gerenciamento de Riscos

A partir de um conjunto de ativos é possível analisar, medir e determinar todos os riscos do sistema num processo conhecido por Análise e Gerenciamento de Riscos [2]. Esse processo pode ser executado de forma paralela ao desenvolvimento do *software*, sendo assim é possível utilizá-lo desde o levantamento de requisitos até a desativação final do sistema [11].

2.4.1 Elementos de um Risco

Para compreender de forma clara o que é um risco é necessário a definição separada de cada um dos seus elementos, listados nas próximas quatro seções.

2.4.1.1 Ativo

Ativo é considerado toda parte de um sistema que possui um valor para qualquer parte interessada que venha a interagir com o sistema [9]. Os ativos são elencados como programas de *software*, informações e até mesmo o pessoal que acessa e mantém esses recursos [11].

2.4.1.2 Vulnerabilidade

Vulnerabilidade é a fraqueza ou ausência de controles de segurança que são utilizados por uma ameaça contra um ativo específico, a fim de subtrair seu valor [9].

2.4.1.3 Fonte de Ameaça

Toda pessoa ou processo iniciado por uma pessoa que possui a intenção de explorar vulnerabilidades do sistema visando perpetuação de um ataque é considerado como fonte de ameaça [9].

2.4.1.4 Ameaça

Quando se é formado um conjunto referente a cenários que tem o potencial comprometedor contra qualquer objetivo que fundamenta a segurança do sistema, entende-se como ameaça [9].

2.4.2 Extração dos Elementos de um Risco

A extração dos elementos de um risco é feita através dos ativos conhecidos do sistema [11]. Para obter um sucesso significativo na extração é recomendado usar uma lista previamente elaborada, facilitando assim a identificação de cada um dos elementos.

2.4.3 Parâmetros de um Risco

As características de um risco são definidas por intermédio de três parâmetros, o primeiro é grau de impacto que o risco apresenta ao sistema. O segundo é grau de probabilidade que o risco tem de ser concretizado. O terceiro é o grau que o risco apresenta ao sistema, sendo derivado dos dois anteriores [11].

2.4.4 Controles de Segurança

Após a atribuição de valores aos parâmetros de cada risco é iniciada a tarefa de identificar, analisar, implementar, monitorar e avaliar os controles de segurança [19]. Os controles de segurança quando implementados ajudam a mitigar os riscos levantados e devem ser monitorados para que haja sua manutenção [11].

2.5 CASOS DE MAL USO

Para haja um melhor entendimento entre todas as partes interessadas do projeto, é necessário utilizar uma forma clara de documentação. Em projetos tradicionais utiliza-se a modelagem através dos Casos de Uso [11]. Uma técnica aplicável como forma de ilustrar a um cenário onde um atacante possui o desejo de explorar o sistema, é utilizar Casos de Mal Uso, para extração e documentação de requisitos não-funcionais. Casos de Mal Uso serão demonstrados na seção 3.4.4.

2.6 ÁRVORES DE ATAQUES

As Árvores de Ataques foram concebidas das Árvores de Falhas, com a intenção de utilizar-se de uma técnica mais metódica e formal [12]. Serve para descrição dos possíveis ataques que uma parte ou todo o sistema pode enfrentar. Sua representação é construída a partir de uma raiz que representa o objetivo do atacante, que possui nodos que simulam a criatividade do ataque [11]. Árvores de ataques serão apresentadas na seção 3.4.5.

2.7 TRABALHOS RELACIONADOS

Uma pesquisa propõe um modelo de segurança para ciclos iterativos de desenvolvimento. Alguns trabalhos identificaram as fraquezas de segurança no método Scrum, essas propostas sugeridas são descritas nas próximas quatro seções.

2.7.1 MODELO DE PROCESSOS PARA TESTES DE SEGURANÇA COM ABORDAGEM ORIENTADA A RISCOS

Durante a pesquisa realizada foi encontrado um modelo de processos [11], que possui um ciclo de vida em espiral e oferece uma sequência de técnicas de segurança que garante um *software* mais seguro, gerando uma documentação robusta. O modelo utiliza todas as técnicas citadas anteriormente além de oferecer uma nova abordagem para Árvores de Ataques. Sua principal proposta é interligar a Análise e Gerenciamento de Riscos com os Testes de Segurança.

2.7.2 SECURITY BACKLOG IN SCRUM SECURITY PRACTICES

A proposta, *Security Backlog in Scrum Security Practices* [1], sugere criar um *Security Backlog*, facilitando a documentação e compreensão do time sobre os requisitos de segurança. Para tal é designado um novo papel dentro do método, chamado de *Security Master* que seria responsável por identificar as características dos riscos de segurança, documentar e testar os recursos selecionados. Contudo, as mudanças sugeridas criam um artefato que se mistura ao ciclo original, aumentando para o time a complexidade do Scrum, e como o papel de *Security Master* é algo novo, não são previstas certificações pelas instituições oficiais, o que dificulta a profissionalização.

2.7.3 S-SCRUM

A proposta *S-Scrum* [4] sugere a incorporação das atividades de análise e design de segurança ao Scrum, visando a melhorar a segurança em uma aplicação mais robusta. O método proposto modifica o Scrum para acomodar documentação de ativos. Também trata as novas exigências de segurança ao decorrer do projeto e durante o desenvolvimento propõem uma gramática livre para a descrição formal do processo. Entretanto, a proposta sugere criar algumas novas etapas, que mesmo ocorrendo em um ramo separado, acabam por misturar-se ao o ciclo original, aumentando a complexidade do Scrum para todo o time.

2.7.4 SDL MICROSOFT

A proposta *Agile SDL* [20] é adaptação do processo SDL utilizado pela *Microsoft* para melhorar a segurança no desenvolvimento de *softwares* com um nível maior de complexidade, como o Windows, *Microsoft Office* e *SQL Server*. Como o ciclo de vida do SDL é utilizado no desenvolvimento de *softwares* em longo prazo, conta com um grande número de requisitos durante o curso do cronograma. Como os métodos ágeis possuem ciclos menores de desenvolvimento, a equipe SDL sugeriu uma adaptação dividindo os requisitos em duas categorias, sendo uma Todos os *Sprints* e outra a Integração. Os requisitos que não se enquadram em nenhuma delas são colocados em outras três subcategorias de requisitos: verificação de segurança, revisão de *design* e planos de resposta. Contudo, a *Agile SDL* não define um método ágil específica e quando utilizada com Scrum, se torna complexa pela dificuldade em identificar e classificar os requisitos.

SCRUMS

Neste capítulo é apresentado o modelo proposto ScrumS, que tem por finalidade tornar mais eficiente a segurança no desenvolvimento de *software* utilizando o método Scrum, além de buscar alternativas para as limitações apontadas nas pesquisas descritas anteriormente.

Inicialmente será apresentado o delineamento da pesquisa, bem como seus objetivos. Também serão descritos todos os processos de como aplicar o ScrumS na prática.

3.1 DELINEAMENTO

Por definição um time Scrum deve ser multidisciplinar, ou seja, possuir integrantes capazes de atuar em diversas áreas de um projeto. Com isso é essencial que todos sejam conscientizados a respeito da segurança da informação para o projeto em questão, tornando os integrantes mais cuidadosos e atentos aos ativos que precisam ser protegidos. Além disso, as etapas convencionais passam por leves mudanças que devem ser esclarecidas para não quebrar o princípio da transparência.

Os eventos da nova estrutura são realizados de forma paralela aos eventos originais do Scrum. A responsabilidade com relação aos requisitos de segurança deve ser delegada aos membros do time ou ser designada a um time de segurança. Não é sugerido um número específico de pessoas para este trabalho nem mesmo um nível mínimo de conhecimento. Porém, apesar do processo ser facilmente compreendido a proteção acrescentada ao *software* irá depender diretamente da experiência dos envolvidos.

Um fator importante para os métodos ágeis é que precisam de documentação simples [1] para que haja fácil entendimento, contribuindo com a agilidade. Os riscos

precisam da mesma clareza para ser compreendidos, visto que podem ser reaproveitados.

É proposta uma nova abordagem para os casos de Mal Uso, utilizados para descoberta de ameaças e fontes de ameaças. Eles serão mapeados para um formato semelhante às USs. As Árvores de Ataques são uma excelente forma de extrair vulnerabilidades, já que permitem que seja simulada a inteligência do atacante. Em conjunto, as ameaças, fonte de ameaças e as vulnerabilidades formam o Inventário de Riscos.

O Inventário de Riscos é sugerido para que o time tenha seu próprio repositório de riscos, tornando possível seu reaproveitamento em outros projetos e guiando sempre as melhores práticas de segurança, já que é possível rever esse repositório e amadurecer os controles de segurança constantemente. A equipe deve usar o DOD para definir os testes de segurança de cada controle sugerido para as iterações, os testes devem ser automatizados e assim como o Inventário de Riscos, podem ser armazenados em um repositório e melhorados no decorrer do tempo.

3.2 MODELO PROPOSTO

ScrumS é um modelo composto por um conjunto de artefatos de segurança que servem para identificar e mitigar riscos em um produto, agregando um maior valor a sua segurança, para tal é sugerido à utilização de dois processos, sendo o primeiro a criação do projeto seguro durante a fase de PG, e o segundo a criação de um Inventário de Riscos derivado dos artefatos de segurança, tornando-a parte de todo o ciclo de vida do produto.

O projeto seguro é essencial para o sucesso do ScrumS, já que é responsável por estabelecer as tecnologias e ferramentas utilizadas no projeto. Como durante o desenvolvimento com Scrum existe a possibilidade de utilizar uma fase para essa função [16], chamada de PG, o modelo proposto sugere que seja feito o mapeamento de projeto seguro [22] dentro desta fase, consolidando assim sua

existência e reafirmando sua importância no que representa as diretrizes de segurança do projeto.

A proposta de Análise de Riscos do ScrumS é responsável por gerar o Inventário de Riscos, e atribuir os controles de segurança que deverão ser implementos para mitigar os riscos encontrados, a figura 1 ilustra o fluxo da Análise de Riscos do ScrumS.

Figura 1 - Análise de Riscos ScrumS

A seguir será apresentado um breve resumo sobre a utilização de cada artefato no processo de Análise de Riscos:

- Ativos: Identificar nas funcionalidades do *Product Backlog* as informações que possuem valor às partes interessadas, e que devem ser protegidas;

- História do Usuário de Segurança: Identificar desejos dos atacantes, com relação aos ativos do sistema;

- Análise de Vulnerabilidades: Criar Árvores de Ataques que possibilitem a simulação da inteligência do atacante ao realizar investidas contra o sistema;

- Inventário de Riscos: É composto por ativos, ameaças, fonte de ameaças e vulnerabilidades que formam os riscos do sistema. A partir deles é possível gerar os controles de segurança necessários.

Importante lembrar que a relação entre os artefatos, sempre está ligada na proporção em que para cada algo encontrado em uma fase anterior podem existir muitos outros na próxima fase. Como o Scrum é suscetível a mudanças, acaba gerando um problema na fase de Análise de Riscos, pois uma funcionalidade pode ser acrescentada ou retirada durante o desenvolvimento. Para que o ScrumS acompanhe o desenvolvimento do sistema contemplando todas as funcionalidades, é necessário analisá-las no PB sempre na ordem pela qual foram priorizadas. Assim, espera-se que as funcionalidades sejam analisadas antes de entrarem no SB. A figura 2 apresenta uma ilustração do fluxo no qual o modelo se aplica.

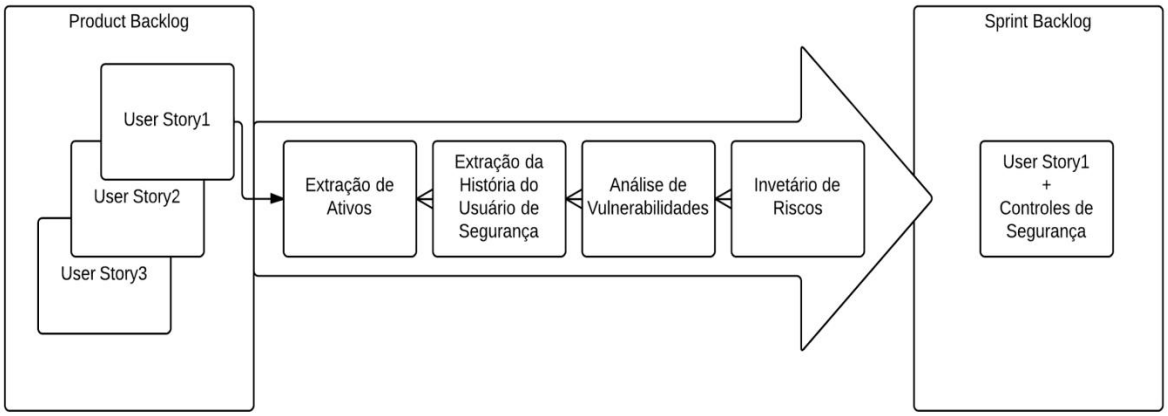

Figura 2 – Representação do Ciclo de Vida ScrumS

3.3 MAPEAMENTO DO MODELO NO SCRUM

A segurança deve ser considerada a partir do início do processo de desenvolvimento, porque assim como a confiabilidade, é uma propriedade que requer planejamento antecipado e um projeto cuidadoso [1].

Com a intenção de abranger todo o ciclo de vida do produto e reduzir o nível de complexidade que a segurança gera aos projetos, propõe-se que o ScumS ocorra de forma paralela. O propósito geral é que todos do time tenham como objetivo zelar pela segurança da informação, de forma que esta seja considerada como parte natural do processo.

A apresentação da figura 3 demonstra o mapeamento da Análise de Riscos ScrumS, descrita na seção 3.2.

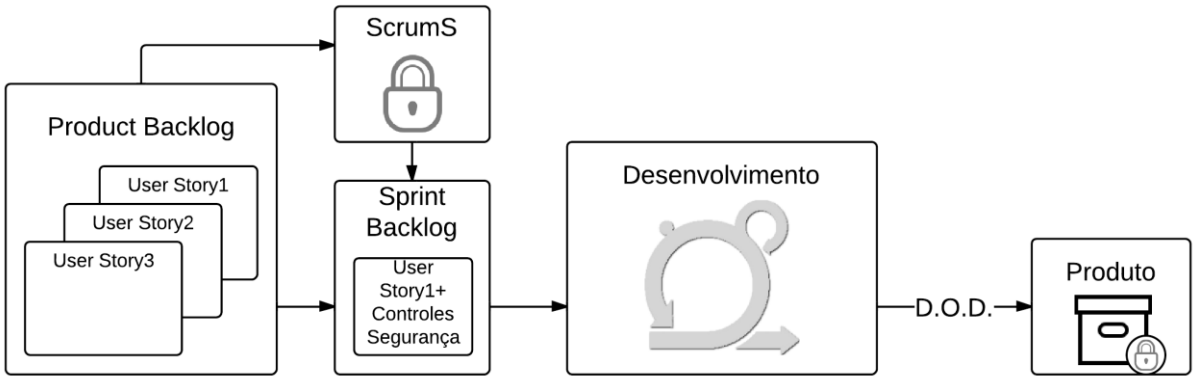

Figura 3 - Mapeamento das Fases do Scrum para o ScrumS

3.4 DESENVOLVIMENTO DE UM PROJETO COM SCRUMS

As próximas seções são responsáveis por explicar e detalhar todo o processo do modelo proposto dentro do ciclo Scrum.

3.4.1 PROJETO SEGURO

Dentro do modelo proposto do ScrumS deve ser agregado ao desenvolvimento o conceito de Projeto Seguro, conforme proposto por V&M [22]. Trata-se de uma fase de Análise onde são escolhidas linguagens, ferramentas e tecnologias, para tal as escolhas se baseiam no fato de que o PG ocorre antes de qualquer iteração do sistema, tendo foco nos desejos do cliente, respeitando as

funcionalidades do sistema a fim de definir a infraestrutura do projeto e preparar o ambiente de desenvolvimento [16]. A próxima seção descreve tal mapeamento.

3.4.1.1 Criando um Projeto Seguro no Pregame

O método Scrum define a fase de PG que não tem como objetivo ter algo potencialmente entregável ao seu término, mas sim realizar configurações no projeto antes de iniciar o desenvolvimento [16]. Sugere-se que esta atividade seja feita logo após a montagem inicial do PB, de modo que o Projeto Seguro não altere a sequência dos eventos do Scrum. Por se tratar de um conceito que tem se tornado popular não causará impacto aos ambientados com o Scrum e proporcionará um ambiente mais seguro, definindo todas as tecnologias, processos envolvidos com o projeto e as políticas de segurança dos ambientes de desenvolvimento, teste e produção.

3.4.2 Análise de Riscos

Logo após a conclusão do PG inicia-se a fase de Análise de Riscos, onde são extraídos os principais riscos pertencentes ao sistema. Suas fases são descritas nas próximas seções. É importante ressaltar que esta fase deve ocorrer sempre de forma paralela ao ciclo original do Scrum e pode ser iniciada assim que o PB estiver priorizado.

3.4.3 Extraindo Ativos

Uma das primeiras tarefas dentro do método Scrum é a de análise dos requisitos do sistema, que é feita através de reuniões e conversas com o cliente. Neste momento, podem ser direcionadas perguntas relacionadas as atribuições dos possíveis usuários e suas interações com o sistema. Desta forma é criado um esboço inicial dos tipos de usuários e quais são suas atribuições ou funções no sistema. Se tratando de uma metodologia ágil, o resultado dessas reuniões costuma ser documentado no formato de US, que em conjunto compõem o PB [15].

Posteriormente estas USs são priorizadas de acordo com o seu valor para o cliente, tarefa que é de extrema importância para fluência do modelo proposto. Com as funcionalidades priorizadas o time sabe qual será a próxima funcionalidade a ser desenvolvida, habilitando o inicio da extração de ativos do sistema [11]. Uma sugestão da pesquisa é de utilizar as reuniões para esboçar os ativos de cada funcionalidade, reduzindo esforço e aumentado agilidade.

A tarefa é realizada a partir das USs que irão para o próximo SB, que é observada com o intuito de tentar descobrir quais ativos estão relacionados à mesma, conforme mostra a figura 4.

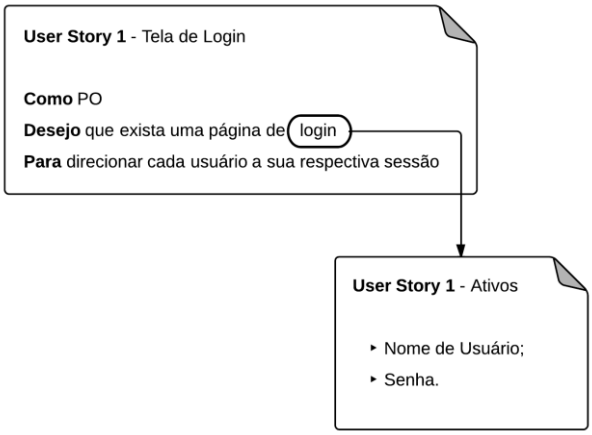

Figura 4 - Extração dos Ativos de Uma User Story

3.4.4 HISTÓRIA DO USUÁRIO DE SEGURANÇA

Em ciclos de vida que não se baseiam em métodos ágeis é comum a utilização de casos de uso para extrair e documentar os requisitos do sistema [11]. Uma abordagem comum para ciclos de vida seguros consiste em acrescentar aos casos de uso os chamados Casos de Mal Uso, descritos por Firesmith [17].

Um Caso de Mal Uso acrescenta atores mal intencionados aos cenários. Tais atores têm como objetivo subverter ou corromper casos de uso do sistema. Como resultado, os Casos de Mal Uso elucidam e descrevem as possíveis ameaças ao sistema [17].

A abordagem proposta no ScrumS consiste em explorar similaridades entre os conceitos de Casos de Uso e USs, utilizadas para documentar requisitos dentro de métodos ágeis, conforme descrito na seção 2.5.

A fim de não acrescentar complexidade á documentação o número de atores foi reduzido para dois, chamados de Atacante Interno e Atacante Externo. Assim, a abordagem apresentada para ScrumS utiliza Histórias do Usuário de Segurança para descrever ameaças (descritas na forma de desejo) do sistema. A nova abordagem é ilustrada na figura 5.

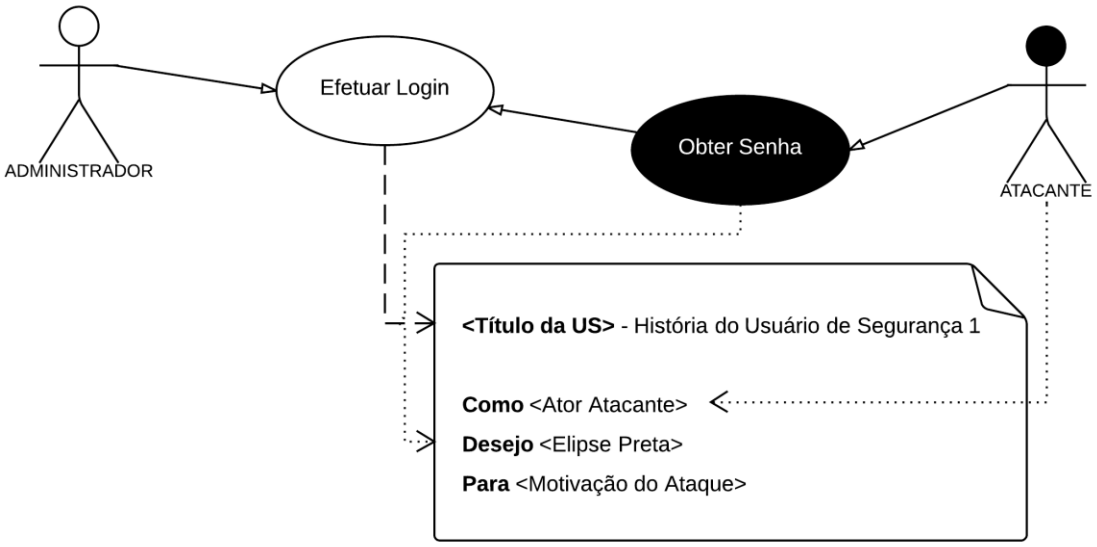

Figura 5 - De Mal Uso para História do Usuário de Segurança

A História do Usuário de Segurança segue o modelo tradicional de US, porém, como atores devem ser identificados os possíveis atacantes do sistema. Cada US pode possuir diversas Histórias do Usuário de Segurança, visto que um mesmo atacante pode representar mais de uma ameaça ao sistema.

3.4.4.1 Extraindo as Histórias do Usuário de Segurança

Logo após a identificação dos ativos de uma US é iniciada a extração das Histórias do Usuário de Segurança. O principal objetivo dessa fase é apontar as potenciais ameaças a sistema.

As Histórias do Usuário de Segurança sempre estão vinculadas as USs a serem implementadas, a figura 6 apresenta um exemplo que demonstra a extração da História do Usuário de Segurança referente à figura 5.

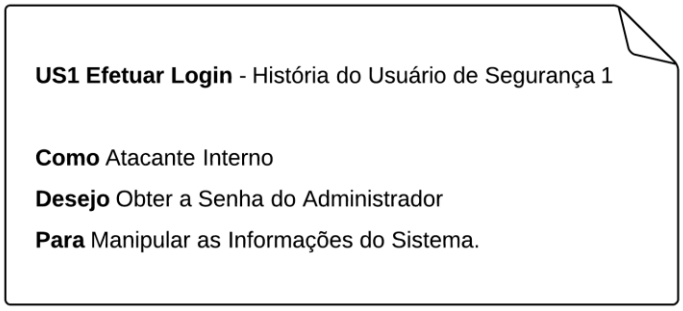

Figura 6 - Representação de História do Usuário de Segurança

3.4.5 Utilizando Árvores de Ataques para explorar uma História do Usuário de Segurança

Após a extração de ameaças ao sistema, a próxima tarefa deve envolver a descrição das vulnerabilidades que podem permitir um ataque ao sistema. A técnica mais comum para descrever ataques ao sistema é conhecida por Árvores de Ataques, conforme descrito por Schneier [12]. Uma Árvore de Ataques simula a inteligência de um atacante, descrevendo o caminho percorrido para que se atinja o objetivo de perpetuar a ameaça. Cada passo necessário para o ataque é visto como potencial vulnerabilidade do sistema. Outra característica importante das Árvores de Ataques é que estas podem ser reutilizadas, Com isso, o time acaba acumulando conhecimento sobre determinada ameaça.

A abordagem proposta para ScrumS utiliza a técnica descrita no modelo de processos para testes de segurança [11], onde cada ameaça extraída através dos casos de Mal Uso serve como ponto de partida para uma Árvore de Ataques. No caso, serão utilizadas as Histórias do Usuário de Segurança.

A figura 7 demonstra uma identificação de vulnerabilidade a partir das Histórias do Usuário de Segurança extraídas anteriormente.

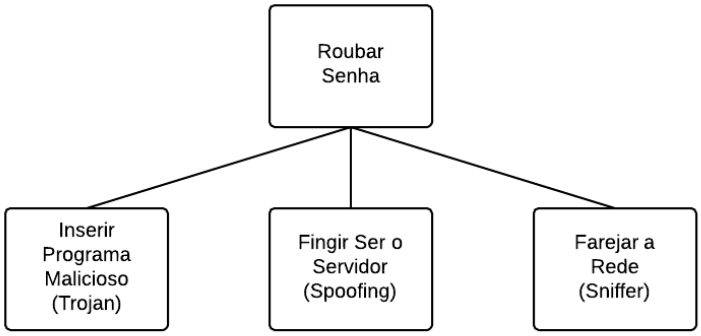

Figura 7 - Demonstração de uma Árvore de Ataques

3.4.6 EXTRAINDO RISCOS

Com base nas extrações de ativos, ameaças e vulnerabilidades descritas anteriormente obtém-se o Inventário de Riscos, documento que descreve os riscos do sistema. Através dele, o time pode identificar todos os controles de segurança necessários ao sistema. Um controle de segurança descreve quais são as providências que devem ser tomadas para que um risco seja eliminado ou mitigado.

A tabela 1 ilustra a identificação dos controles de segurança referentes às vulnerabilidades extraídas anteriormente.

Tabela 1 - Inventário de Riscos

Ativo	Senha	Senha	Senha
Fonte de Ameaça	Atacante Interno	Atacante Interno	Atacante Interno
Ameaça	Roubar Senha	Roubar Senha	Roubar Senha
Vulnerabilidade	Inserir um programa malicioso (*Trojan*)	Fingir ser o servidor utilizando (*Spoofing*)	Senha enviada ao servidor em texto plano
Risco	Uma pessoa interna pode inserir um programa malicioso	Uma pessoa pode fingir ser o servidor (*Spoofing*)	Uma pessoa externa pode farejar a rede (*Sniffer*)
Controle de Segurança	Instalar solução e *anti-malware* nos micros que utilizarão o sistema	Implementar comunicação segura *Open SSL*	Cifrar a senha

3.4.7 Analisando Riscos

Embora documentados os riscos também precisam ser priorizados, pois o tempo da *Sprint* pode não ser o suficiente para implementar os controles de segurança necessários. A priorização é feita determinando o impacto e probabilidade de cada risco. Uma tabela com três níveis pode ser considerada eficiente na maioria dos casos [11], onde tanto o impacto quanto a probabilidade são estimados em alto, baixo ou médio. Os riscos que obtiverem níveis mais altos demandam maior atenção, ou seja, devem ser discutidos as possíveis soluções e os custos que gerariam para o projeto, a figura 8 demonstra um exemplo com os controles de segurança identificados no Inventário de Riscos da seção 3.4.6.

A tabela 2 apresenta uma análise feita sobre os riscos descritos na tabela 1.

Tabela 2 - Determinando o Nível do Risco

RISCO	Baixo	Médio	Alto
Um atacante interno pode inserir um programa malicioso.		X	
Um atacante interno pode fingir ser o servidor (*Spoofing*)			X
Um atacante interno pode farejar a rede (*Sniffer*).			X

3.4.8 Criando Sprint Backlog com Controles de Segurança Selecionados

O processo de Análise de Riscos do ScrumS tem como objetivo adicionar a funcionalidade que irá entrar em uma *Sprint,* vinculando a ela uma lista com os controles de segurança previamente discutidos. Tais funcionalidades recebem o nome de Tarefas de Segurança. A figura 8 demonstra um controle de segurança da US1.

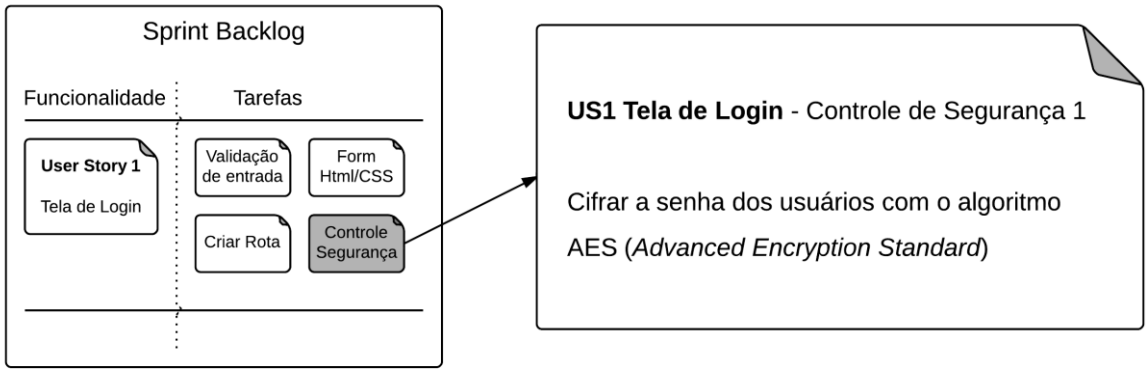

Figura 8 – Representação do Controle de Segurança no SB

3.4.9 DEFININDO O DOD

No trabalho proposto sugere-se que durante a reunião de planejamento da próxima *Sprint* o time, tendo consciência dos controles de segurança selecionados e do esforço gasto para atingir o objetivo, utilize o DOD para definir quais os testes de segurança devem ser realizados.

Podem ser exigidos diversos testes de Segurança que são geralmente divididos em Testes de Unidade, Integração, Aceitação e Penetração. Primando pelo principio ágil [13] o modelo proposto indica que os testes devem ser automatizados e fazer parte do repositório do time, existindo a manutenção e reaproveitamento dos mesmos.

3.5 RESUMO DO SCRUMS

Para tornar possível a obtenção dos riscos do sistema dentro do método Scrum a pesquisa se baseou nas técnicas de segurança apresentadas nas seção de contextualização do trabalho, que são resumidamente descritas abaixo.

Para consolidar a importância do PG foi utilizado o conceito de Projeto Seguro [22], orientado assim o time Scrum a definir previamente todas as diretrizes do projeto. O PG é fundamental para direcionar políticas, tecnologias e processos para todos os ambientes do *software*.

O processo de Análise de Riscos foi extraído da proposta [11] cujo modelo exigia muita documentação e não orientava como proceder quando se trata de um método ágil. Com o uso simplificado desse modelo foi possível eleger as técnicas necessárias que possibilitam a identificação dos riscos dentro do método Scrum de forma menos complexa.

A extração de ativos permite a criação de uma lista para cada funcionalidade com os bens que precisam ser protegidos.

As Histórias do Usuário de Segurança apresentam as ameaças e suas fontes, tendo como base os ativos.

Árvores de Ataques exploram as vulnerabilidades do sistema, simulando a inteligência do atacante, a partir do desejo de uma História do Usuário de Segurança.

E por último o Inventário de Riscos documenta e prioriza os riscos, além de auxiliar na proposta de controles de segurança, que são analisados e disponibilizados no SB.

Em resumo, com ScrumS é possível utilizar uma nova forma de abordar segurança no método Scrum, sendo possível contemplar todas as funcionalidades do *software*, mesmo quando disponibilizadas posteriormente à criação do PB.

É importante destacar que o ScrumS gera documentação mais complexa do que o habitual para métodos ágeis. Contudo, essa documentação auxilia o time a entender melhor as vulnerabilidades do *software* e descentraliza o conhecimento sobre segurança, guiando as práticas mais seguras que promovem um produto melhor, além de proporcionar a maturação e reaproveitamento dos controles de segurança encontrados em outros projetos.

ESTUDO DE CASO

É apresentado neste capítulo um projeto onde o ScrumS foi aplicado, com o objetivo de observar o seu uso no desenvolvimento de uma aplicação prática expondo suas vantagens e limitações.

4.1 APRESENTAÇÃO DO PROJETO CPA

A Comissão Própria de Avaliação (CPA) é um processo de qualidade de grande importância para as instituições acadêmicas, sendo inclusive uma exigência do SINAES (Sistema Nacional de Avaliação da Educação Superior), órgão responsável por fiscalizar os cursos nacionais de nível superior [7].

O CPA tem como objetivo avaliar a qualidade das várias competências da instituição, desde sua infraestrutura até o corpo docente, bem como coordenadores e diretor, a fim de identificar pontos fortes e trabalhar os pontos fracos da instituição.

Esta avaliação é feita por meio de um questionário anual, que é respondido pelas entidades do campus (discentes, docentes e técnicos administrativos) anonimamente por meio de perguntas direcionadas pela própria comissão. Essas perguntas têm por base as dimensões avaliadas pelo SINAES.

O projeto surge de uma necessidade do campus Caraguatatuba do Instituto Federal de São Paulo (IFSP), não possuir um sistema que permitisse à comissão local criar questionários e visualizar graficamente os resultados.

4.1.1 IMPORTÂNCIA DO CPA PARA A INSTITUIÇÃO

A grande importância do sistema para a instituição é a garantia de um canal aberto com os funcionários e usuários dos serviços oferecidos. Se tratando de uma votação anônima e não obrigatória é possível determinar quais são os pontos fortes e fracos da Instituição.

Os pontos fortes uma vez descobertos devem ter as suas causas analisadas, de modo que tenham o seu estado mantido. Já os pontos fracos devem ser trabalhados para que possam ser mitigados, visando a melhoria da Instituição para toda a comunidade.

4.1.2 Aspectos do Projeto CPA

O projeto apresentou como requisitos primários:

- O código fonte deve ser aberto, visando que o produto seja mantido em constante evolução pelos alunos do Campus;
- Trabalhar em plataforma *Web* e móvel, sendo acessível à maioria dos dispositivos com conexão a internet;
- A votação deve ocorrer de forma anônima para preservar a identidade do usuário;
- O docente precisa ser identificado quando avaliar sua própria disciplina;
- O banco de dados precisa ser protegido de acessos indevidos.

4.1.3 Asserção do ScrumS no Projeto CPA

O projeto CPA apresentou necessidades ligadas a segurança da informação, pois a maioria dos usuários participam de forma anônima, sendo importante a preservação da identidade destes usuários. Em contrapartida o sistema deve garantir que a integridade dos votos não seja violada, agregando credibilidade e fidelidade aos votos registrados.

4.2 DESENVOLVIMENTO DO PROJETO CPA

Esta seção do trabalho descreve todo o desenvolvimento do projeto CPA utilizando o ScrumS, o orientador da pesquisa foi definido como PO por conhecer as

regras de negócio, o time de desenvolvimento foi composto pelos autores da pesquisa, assim como dividiram o papel de SM.

4.2.1 Levantamento de Requisitos e Visão Arquitetural

Ocorreu uma reunião com os interessados do projeto para coletar requisitos e posteriormente foi apresentada uma visão do produto aos mesmos. Assim que aprovado, iniciou-se a criação do PB com as USs iniciais. Seguindo a orientação dos autores Pham e Phuong-Van [10] foi utilizado no PB o processo de visão arquitetural que segmenta as funcionalidades levantadas em módulos. No projeto do CPA foram identificados três módulos.

O primeiro módulo serve como base para que possam ser realizadas as votações. A partir dele o coordenador do CPA, ou um usuário designado por ele, deverá configurar o programa para que represente o estado da instituição no momento (semestre atual). Esta representação faz referência aos cursos, suas disciplinas e turmas, as áreas, seus professores e funcionários e as perguntas direcionadas que serão respondidas pelos alunos, professores e técnicos administrativos. Ao término da modelagem do novo questionário o sistema deve iniciar a votação gerando os usuários anônimos e senhas para todos os usuários do sistema.

No segundo módulo deve ocorrer a votação. É necessário que seja oferecido um mecanismo de autenticação de sessão anônimo que identifique quais são as perguntas que o usuário deverá responder e uma vez que cada pergunta é respondida, não poderá ser alterada. Ao se tratar de um aluno, devem ser carregadas perguntas sobre todas as disciplinas do curso que está matriculado e perguntas gerais sobre a instituição.

Quando um professor realiza o acesso ao sistema é necessário que sejam carregadas perguntas sobre todas as disciplinas que ministra. Técnicos administrativos por sua vez, respondem perguntas gerais sobre o instituto e sobre seu setor.

No terceiro módulo entram regras de negócio sobre o que foi armazenado no banco de dados na etapa anterior, ou seja, é a fase de análise das respostas que ocorreram no período de votação. Estes resultados precisam ser exibidos no formato de gráficos e deve ser exportados para criação do relatório do questionário. O sistema deve identificar os pontos fortes e fracos da instituição e de cada curso. Assim que identificado e autorizado pelo coordenador do CPA, devem ser enviados e-mails automáticos para cada responsável pelos pontos fracos. Também devem ser disponibilizados na internet os gráficos de cada votação, exceto os relacionados ao corpo docente.

Após o projeto ter sido segmentado em módulos foi realizada a priorização e pontuação das USs. A priorização foi feita de acordo com a ordem dos módulos e a pontuação de acordo com o tempo estimado sugerido pelos desenvolvedores. O total dos pontos da soma das USs dos dois primeiros módulos totalizou três *Sprints*. Diante disso foi definido que apenas os dois primeiros módulos seriam desenvolvidos para avaliar a utilização do ScrumS.

4.2.2 PREGAME E PROJETO SEGURO

Compreendidas as exigências o time se reuniu para estabelecer quais seriam as tecnologias utilizadas no projeto, que foram definidas conforme a tabela:

Tabela 3 - Tecnologias Definidas Durante o Projeto Seguro
(continua)

Requisitos	Definições
Linguagem de Programação	Ruby 2.0.0
Web Framework	Ruby on Rails 4.0.0
Framework para Design Responsivo	Bootstrap 3.0.0
Sistema Gerenciador de Banco de Dados	PostgreSQL 9.2
Controlador de Versão	Git 1.8.3.2

Tabela 3 - Tecnologias Definidas Durante o Projeto Seguro (conclusão)

Requisitos	Definições
Repositório do Projeto	http://github.com/ralrodrigues/cpa.git
Servidor de Aplicação	Phusion Passenger 4.0.0 (Standalone)
Licença do Software	GPL v2
SO do Ambiente de Desenvolvimento	Linux Ubuntu 13.04

As escolhas das tecnologias foram feitas de acordo com os requisitos iniciais, que faziam referência à necessidade do sistema rodar na plataforma *Web* (não precisar ser instalado) e poder ser executado a partir de qualquer dispositivo conectado a internet.

O *framework Web Ruby on Rails* foi considerada a melhor opção para o projeto por oferecer um conjunto de ferramentas para construção de sites, além de possuir nativamente implementadas defesas para os ataques mais comuns à páginas dinâmicas.

O *framework* para design responsivo *Bootstrap* foi selecionado por ter sido recentemente atualizado, oferecendo um visual agradável e suporte a maioria dos navegadores e versões disponíveis no mercado.

Quanto ao sistema gerenciador de banco de dados, o *PostgreSQL* foi utilizado pela integração com a maioria dos recursos do *Ruby on Rails*, além de ser robusto e escalável.

Para o controle de versão foi selecionado o *Git* pela simplicidade e eficácia, além de possuir o *Github*, uma plataforma na internet que permite o armazenamento do código fonte gratuitamente. Havendo a necessidade é possível pagar o armazenamento privado.

Também foram realizados testes em servidores de aplicação e dada preferência ao *Phusion Passenger* por possuir uma versão *standalone* (que não necessita de outro servidor de aplicação sendo executado na máquina).

A licença GPL v2 foi escolhida por ser *Copyleft* e obrigar a todos que distribuirem esse código o manterem sob os mesmos termos, garantido assim o trabalho dos desenvolvedores desse projeto.

O Ubuntu 13.04 possui uma excelente compatibilidade com todas as tecnologias selecionadas, além de ser um sistema operacional desenvolvido de forma restritiva.

A politica de *backup* de todo o sistema foi definida para ocorrer duas vezes por semana e foram criados dois usuários desenvolvedores com acesso total ao sistema e dois usuários administradores que tomam decisões importantes sobre o funcionamento do sistema.

4.2.3 SPRINT UM

A *Sprint* Um é apresentada nas próximas duas seções, onde a primeira seção é dedicada para a apresentação do planejamento dessa *Sprint* e a segunda seção descreve como ocorreu sua implementação.

4.2.3.1 PLANEJAMENTO DA SPRINT

Após as USs estarem priorizadas o processo de Análise de Riscos do ScrumS foi iniciado para acrescentar ao SB os controles de segurança necessários. Foram definas as três primeiras USs para serem desenvolvidas na primeira *Sprint*. O processo de Análise de Riscos identificou dois ativos, a senha do administrador e o *backup* da base de dados, o que resultou em oito controles de segurança a serem desenvolvidos durante essa *Sprint*.

Ao realizar a análise dos riscos, sugerido na seção 3.4.7, dois controles foram retirados do SB, pois o time identificou que essas implementações iriam tomar um maior tempo do que a *Sprint* possuía de duração.

4.2.3.2 DESENVOLVIMENTO

A implementação dos requisitos do primeiro módulo implica no gerenciamento de questionários que foi dividido em três estados: em preparação, em votação e encerrado. Ao inicio do desenvolvimento os programadores e o PO já sabiam quais os controles que deveriam ser implementados, sendo assim, para proteger o ativo de Backup da Base de Dados o repositório foi totalmente cifrado e definiu-se uma senha forte para o Banco de Dados. Restringiu-se o acesso do usuário criado pelo PostgreSQL, sendo cifrada sua senha de administrador.

4.2.4 SPRINT DOIS

A *Sprint* Dois é apresentada nas próximas duas seções, onde a primeira seção é dedicada para a apresentação do planejamento dessa *Sprint* e a segunda seção descreve como ocorreu sua implementação.

4.2.4.1 PLANEJAMENTO DA SPRINT

Na metade do desenvolvimento da *Sprint* um, iniciou-se a análise dos ativos das USs que iriam fazer parte da *Sprint* Dois. Assim ao montar o SB Dois foi possível identificar os controles de segurança necessários para implementação da segunda parte do primeiro módulo que havia sido selecionada para compor essa *Sprint*. No caso, foram identificados os ativos, votos e senhas dos usuários, que resultaram em nove controles de segurança.

4.2.4.2 DESENVOLVIMENTO

As USs desta *Sprint* contavam com as regras de negócio mais importantes para todo o sistema, pois o entregável deveria gerar as senhas aleatórias e carregar as perguntas que cada usuário iria responder. Houve a mesma proposta pela Análise de Riscos do ScrumS referente a senha do administrador com um acréscimo relacionado ao treinamento dos usuários, a fim de evitar ataques de engenharia social.

Para isso houve o reaproveitamento dos controles de segurança da *Sprint* Um e foi disponibilizado o material que orienta o coordenador do CPA sobre possíveis formas de fraude. Coube ao coordenador CPA orientar o pessoal responsável pela gestão da votação. Os votos tiveram como controle de segurança a cifragem da coluna no banco de dados onde é armazenado.

4.2.5 SPRINT TRÊS

A *Sprint* Três é apresentada nas próximas duas seções, onde a primeira seção é dedicada para a apresentação do planejamento dessa *Sprint* e a segunda seção descreve como ocorreu sua implementação.

4.2.5.1 PLANEJAMENTO DA SPRINT

Foi projetado para essa *Sprint* a implementação do segundo módulo do projeto que consiste na visualização do questionário pelos usuários e controle de votos. A *Sprint* não teve Análise de Riscos, pois o time preferiu dedicá-la para a coleta dos resultados obtidos com ScrumS, aplicando alguns testes de segurança.

4.2.5.2 DESENVOLVIMENTO

Ocorreu à implantação de uma página principal do sistema, com o material disponibilizado pela coordenadora do CPA, também foram criados os questionários responsivos para serem acessados por qualquer dispositivo. Para validar os

controles de segurança implementados, foram aplicados testes de verificação com ferramentas existentes no BackTrack5 R3.

RESULTADOS OBTIDOS

Nesse capítulo serão apresentados os resultados da utilização do modelo proposto no projeto CPA.

5.1 DEFINIÇÃO DO PROJETO SEGURO

Como o mapeamento do projeto seguro foi realizado dentro do PG, não houve dificuldades por parte dos desenvolvedores em definir todas as plataformas, tecnologias e processos, porém exigiu do time um conhecimento sobre vocabulário mais específico de segurança, que teve de ser aprendido.

O ponto positivo deste processo é que desde o início do projeto, todos os envolvidos possuíam o conhecimento acerca das políticas de segurança adotadas para realização e manipulação dos *backups* gerados da base de dados e identificação dos usuários com maior privilégio no sistema.

5.2 IDENTIFICAÇÃO DE ATIVOS DAS USER STORIES

Os ativos de cada funcionalidade do sistema foram identificados e documentados antes do início da *Sprint* responsável pelo seu desenvolvimento, ou seja, o processo de Análise de Riscos ScrumS ocorreu antes do início de cada *Sprint*.

Para haver uma melhor identificação de cada ativo, na maioria das reuniões com o PO houve o acompanhamento do time de desenvolvimento. Esse acompanhamento facilitou a identificação dos ativos de cada funcionalidade desejada, orientando o time sobre quais os dados de maior importância para os interessados.

5.3 IDENTIFICAÇÃO DE ATACANTES E AMEAÇAS COM AS HISTÓRIAS DO USUÁRIO DE SEGURANÇA

Seguindo a abordagem proposta pela pesquisa, os atacantes foram mantidos como externos ou internos. Para identificar os atacantes e suas ameaças, foi utilizado o formato de Histórias do Usuário de Segurança, definindo assim os cenários de ataque ligados aos ativos definidos na fase anterior do processo.

A técnica proposta pela pesquisa ajudou a gerar um fácil entendimento e visualização das formas de ataque, identificando assim as possíveis motivações que poderiam resultar na exploração de cada funcionalidade.

5.4 IDENTIFICAÇÃO DE VULNERABILIDADES COM AS ÁRVORES DE ATAQUES

Para simular a inteligência do atacante foram criadas Árvores de Ataques a partir das Histórias do Usuário de Segurança. Na figura 9 pode ser visualizada uma dessas Árvores.

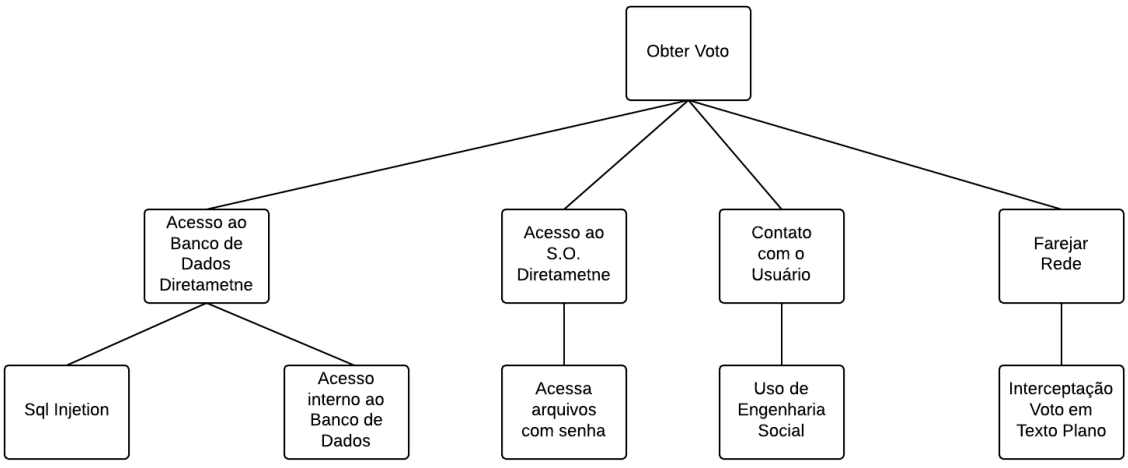

Figura 9 - Demonstração de uma Árvore de Ataques

A utilização desta técnica proporcionou ao time de desenvolvimento uma forma simples de visualizar os caminhos que um atacante pode percorrer para atingir

seu objetivo. Apesar de que a elaboração das Árvores de Ataques exigirem um bom conhecimento sobre técnicas de ataque a sistemas, sua criação acaba gerando árvores que podem compor um repositório, facilitando assim sua maturação.

5.5 CRIAÇÃO DO INVENTÁRIO DE RISCOS

O Inventário de Riscos foi construído em forma de tabela, onde são agrupados por linha as USs; ativos; fonte de ameaças; ameaças e vulnerabilidades.

A última coluna da tabela foi preenchida com os controles de segurança que mitigariam os riscos levantados. Para isso o time de desenvolvimento discutiu quais seriam os melhores controles de segurança para cada vulnerabilidade encontrada.

Está foi a atividade proposta pela pesquisa que exigiu um maior esforço do time de desenvolvimento, porém o ganho encontrado é de que com o passar do tempo a equipe adquiriu uma maior capacidade de mitigar os erros, pois esses controles geraram um repositório que pode ser constantemente atualizado e consequentemente reaproveitado.

5.6 DEFINIÇÃO DOS CONTROLES DE SEGURANÇA

Deve-se destacar que as *Sprints* tem um tempo curto de duração. Ocorre que alguns controles de segurança não podem ser implementados no mesmo tempo de duração da *Sprint*. Assim, foi utilizada a proposta de Análise de Riscos, descrita na seção 3.4.7, para auxiliar a reunião de planejamento da *Sprint*, elencando quais os controles de segurança deveriam ser implementados, deixando outros para uma próxima *Sprint*.

5.7 RESUMO DOS RESULTADOS OBTIDOS

O modelo proposto conciliou técnicas de segurança com o método Scrum no desenvolvimento do projeto, que além de proporcionar a entrega incremental de um produto mais seguro, disponibilizou um Inventário de Riscos ao sistema.

O mapeamento realizado sobre Projeto Seguro no PG garantiu as diretrizes em que o projeto ocorreu e não aumentou a complexidade do Scrum aos utilizadores.

A Análise de Riscos ScrumS, foi utilizada de forma paralela ao ciclo original, tendo apenas os controles de segurança acrescidos nas USs, antes das mesmas entrarem no SB.

Neste projeto todos os membros do time dedicaram-se ao ScrumS, o que gerou um atraso para iniciar a *Sprint* Um, principalmente por não existir um repositório de Riscos e seus respectivos Controles de Segurança.

Toda a documentação gerada pela utilização das técnicas sugeridas foi de grande importância não só para criar um repositório como para manter o time conscientizado dos riscos. Isso consequentemente melhorou os próprios conceitos do time sobre a segurança. Entende-se que a segurança tornou-se parte do ciclo de vida do *software*.

CONCLUSÕES FINAIS

Esse capítulo é dedicado a apresentação das conclusões finais e indicações dos possíveis trabalhos futuros.

6.1 ATENDIMENTO AOS OBJETIVOS PROPOSTOS

Foram investigadas maneiras de melhorar o processo de desenvolvimento de *software* utilizando o método Scrum priorizando o requisito de segurança da informação.

A Análise de Riscos ScrumS, permitiu que cada funcionalidade do sistema tenha seus riscos controlados e documentados no Inventário de Riscos. O inventário possui um conjunto de Controles de Segurança, que são anexados as USs e entregue ao time de desenvolvimento.

As fases foram extraídas e mapeadas aos eventos do Scrum, que se manteve fluente, pelo fato de que são realizadas em paralelo ao seu fluxo original, causando um menor impacto e gerando uma complexidade menor.

Uma nova abordagem foi elaborada para extrair ameaças e suas fontes. Em substituição aos Casos de Mal Uso, foram propostas as Histórias do Usuário de Segurança, visto que não são utilizados casos de uso no Scrum, já que o sistema não é previamente planejado e sim incrementado através de *Sprints*.

Entendeu-se que os controles de segurança possuem a finalidade de tornar a documentação objetiva, para que o time tenha conhecimento sobre quais serão implementados na *Sprint*, que opcionalmente podem não ser implementado devido ao pouco tempo da *Sprint*. Essa decisão geralmente é tomada durante o planejamento da *Sprint*, onde cada controle é elencado por impacto e probabilidade de ocorrer.

O último objetivo, relacionado a automatização e validação dos controles de segurança, ficou a cargo do time escolher qual o DOD referente aos testes dos controles implementados. A pesquisa demonstrou que existe a necessidade de elaborar planos de teste de segurança. Sendo necessário seu mapeamento para o método Scrum.

6.2 CONTRIBUIÇÕES

O Scrum possui como característica a facilidade de entendimento [1]. Este benefício se manteve no ScrumS, pois o modelo proposto busca trabalhar em paralelo ao ciclo original sem aumentar a complexidade do método Scrum.

Para utilizar o ScrumS é necessário possuir um nível de conhecimento prévio sobre conceitos da segurança da informação, assim como termos da área Entretanto, não é necessário que o projeto possua um analista de segurança, pois ao dominar esses conhecimentos, o próprio time pode dividir as responsabilidades do modelo proposto, sendo capaz de executá-lo.

A fixação do PG com o Projeto Seguro dentro do método Scrum é vital para que o time saiba quais as políticas de segurança, tecnologias e processos utilizados. Desta forma é possível mitigar riscos relacionados à parte de infra estrutura do projeto, assim como riscos que podem estar relacionados a tecnologia.

A utilização do modelo proposto agrega ao time de desenvolvimento um repositório com riscos, que podem ser reutilizados em outros projetos, bem como mantê-los constantemente em processo de melhoria.

O mapeamento do caso de Mal Uso descrito por Firesmith [17], em história do usuário de segurança pode ser considerada como uma contribuição importante na tentativa de diminuir o nível de complexidade que habitualmente a documentação na parte de segurança possui nos projetos.

A julgar pelo nível do risco, fica a critério do time (que pode levar em conta o prazo de entrega do produto) implementar imediatamente (na *Sprint* atual) ou manter o risco documentado e controlado, para que possa ser mitigado posteriormente.

6.3 LIMITAÇÕES

Não há uma proporção entre membros do time Scrum para membros que ficaram a cargo de executar o ScrumS, visto que depende da necessidade do projeto. O nível de segurança que será agregado ao mesmo, está diretamente relacionado ao conhecimento sobre segurança que o time possui.

Outra limitação é que sobre certos requisitos de segurança levantados podem ser complexos e até mesmo inéditos, exigindo um conhecimento além do que os integrantes possuem. Adiar de forma controlada a implementação pode não ser a melhor opção, visto que o atacante sempre tentará o ganho de acesso pela parte mais vulnerável.

A principal limitação encontrada na pesquisa está relacionada à primeira implantação do ScrumS em um projeto Scrum, pelo fato do time não estar acostumado a lidar com a segurança e ter que iniciar a criação dos repositórios de risco e testes de segurança. Isso que gera um atraso no início da primeira *Sprint*.

Por se tratar de um processo crítico a fase de Análise de Riscos pode exigir mais documentação do que habitualmente se encontra num projeto Scrum [1]. A fim de não carregar o projeto com documentação extensa e laboriosa sugere-se o uso de algum repositório com Análise de Riscos pronta para uso e que possam ser aplicadas em qualquer projeto.

6.4 TRABALHOS FUTUROS

Ausente no modelo proposto, uma fase de extrema importância é a de testes de segurança. A elaboração de um plano de testes de segurança e casos de testes de segurança viria a validar os requisitos levantados.

Recomenda-se que seja desenvolvido um *framework* para simplificar a Análise de Riscos e outro *framework* para automatizar os Testes de Segurança. Ambos os *frameworks* estariam de acordo com o modelo proposto e proporcionariam maneiras de reaproveitar o trabalho realizado em outros projetos e aumentar a velocidade com que é feita a primeira utilização do ScrumS.

REFERÊNCIAS

[1] AZHAM, Z.; GHANI, I.; ITHNIN, N. **Security backlog in Scrum security practices**. Software Engineering (MySEC), 2011.

[2] BISHOP, M. **Computer security**: art ad science. New York: Addison-Wesley. 2002. 1136 p.

[3] BENZEL, T. V. et al. Design principles and guidelines for security. 2007

[4] FIRESMITH, D. G. **Security Use Cases**. Journal of Objetct Technology. v.2, n.3. p.55-64, 2003.

[5] HIGHSMITH, J. **Agile Project Management – Creating Innovative Products**. Pearson Education, 2004.

[6] MCGRAW, G. **Software security**. IEEE Security & Privacy, v. 2, n. 2, p. 80–83, doi:10.1109/MSECP.2004.1281254, 2004.

[7] MINISTÉRIO DA EDUCAÇÃO. **Instrumento de Avaliação de Cursos de Graduação presencial e a distância.** 2012. Disponível em: <http://download.inep.gov.br/educacao_superior/avaliacao_cursos_graduacao/instrumentos/2012/instrumento_com_alteracoes_maio_12.pdf>. Acesso em: 12 nov. 2013.

[8] MOUGOUEI, D.; SANI, N.; ALMASI, M. **S-Scrum: a Secure Methodology for Agile Development of Web Services**. World of Computer Science, 2013.

[9] PFEEGLER, C.; PFEEGLER, S. L. **Security in computing.** [S. I.]: Prentice Hall, 2006.

[10] PHAN, A.; PHAN, P. **Scrum in Action: Agile Software Project Management and Development**. Novatec, 2011. 287p.

[11] PINTO, Nelson Alves. **Um Modelo de Processos para Testes de Segurança com abordagem orientado a riscos**. 2008. 111f. Tese de Mestrado – Instituto Tecnológico de Aeronáutica, São José dos Campos.

[12] SCHNEIER, B. **Secrets and lies**. [S. l.]: John Wiley & Sons, 2000, 412 p.

[13] SCHWABER, K. et al. **Principles behind the Agile Manifest**. Disponível em: <http://agilemanifesto.org/principles.html>. Acesso em: 21 jun. 2013.

[14] SCHWABER, K. **Agile Project Management with SCRUM**. Microsoft Press, 2004.

[15] SCHWABER, K. ; SUTHERLAND, J. **Scrum Guide**. Scrum Alliance, v. 19, n. 6, p. 21, 2009. Disponível em: <http://www.scrum.org/scrumguides/>.

[16] SCHWABER, K. **Scrum development process**. Business Object Design and Implementation, , n. April 1987, p. 10–19, 1997. Disponível em: <http://link.springer.com/chapter/10.1007/978-1-4471-0947-1_11>. Acesso em: 19 dez. 2013.

[17] SINDRE, G., OPDAHL, A.L. Elicting security requirements by misuse cases. **Requirements Engineering**. v. 10, n. 1, p. 34-44.

[18] STEFFEN, J. **O que são essas tais de metodologias Ágeis ?** (O Mundo depende de Software), Janeiro 2012. Disponível em: <https://www.ibm.com/developerworks/community/blogs/rationalbrasil/entry/mas_o_que_s_c3_a3o_essas_tais_de_metodologias__c3_a1geis?lang=en>. Acesso em: 5 nov. 2013.

[19] STONEBURNER G.; GOGUEN A.; FERINGA, A. **Risk management guide for information technology systems**. Gaithersburgh, 55 p., 2002.

[20] SULLIVAN, B. Agile SDL: Streamline Security Practices For Agile Development. **Msdn Magazine**: The Microsoft Journal for Developers, Estados Unidos, n. , p.52-58,

nov. 2008. Mensal. Disponível em: <http://msdn.microsoft.com/en-us/magazine/dd153756.aspx>. Acesso em: 07 nov. 2013.

[21] VERSION ONE. **State of Agile Development Survey Results**: Version One, 5° edição, 2010.

[22] VIEGA, J.; MCGRAW, G. **Building secure software**. Boston: Addison-Wesley, 2001. 412p.