S2D2: Security & Safety Driven Development

an approach for agile services development process

Carlo Marcelo Revoredo da Silva
Universidade Federal de Pernambuco (UFPE)
Av. Jorn. Aníbal Fernandes, s/n, Cidade Universitária
Recife, Pernambuco 50740-560
cmrs@cin.ufpe.br

Vinícius Cardoso Garcia
Universidade Federal de Pernambuco (UFPE)
Av. Jorn. Aníbal Fernandes, s/n, Cidade Universitária
Recife, Pernambuco 50740-560
vcg@cin.ufpe.br

Eduardo Luzeiro Feitosa
Universidade Federal do Amazonas (UFAM)
Av. Rodrigo Otávio, 6200, Japiim
Manaus, Amazonas 69077-000
efeitosa@icomp.ufam.edu.br

ABSTRACT

The purpose of this paper is to present a methodology to guide a service secure development for agile development teams. The proposal is guided by a Reference Architecture (RA) based on a threat and risk modeling. The methodology predicts emphasis on behaviors that consider threats, attacks, vectors, risks, actors, devices, and assets. Into the agile concerns, the RA concerned with presenting a lightweight process focused on the efficiency and reduction efforts for good practices for secure development, considering design and coding, as well as acceptance tests. In its current state, the proposal was validated in two ways, namely: through ten "top threats" catalogs, such as [15, 18]. In addition, applying the proposal as an agile tool for threat predictability and risk control over the source-code. As a result, it was possible to characterizing as a mechanism that provides a guided methodology for threat and risk driven-development.

KEYWORDS

Secure Programming, Agile Development, Reference Architecture, Threat Modeling, Risk Modeling

1 CARACTERIZAÇÃO DO PROBLEMA

Atualmente a *Web* se tornou a plataforma mais empregada para disponibilização de serviços, atingindo desde o segmento financeiro até o monitoramento de infraestruturas críticas. Em linhas gerais, pode-se afirmar que a *Web* é um ecossistema onde pessoas e empresas, através de dispositivos e serviços, trocam dados (sensíveis ou não) - considerados ativos de informação, que podem ou não conter ou representar algum valor agregado ao seu proprietário.

É neste cenário que diversas organizações (privadas ou governamentais), antes tradicionalmente orientadas a construção de produtos, agora apresentam uma concisa tendência em adotar seus modelos de negócio focados no provimento de serviços [14]. Assim, essas organizações, orientadas em **serviços**, acabam por disponibilizar seus recursos através de ***entrypoint*** distintos, a exemplo de um navegador *Web*, serviço de terceiros ou dispositivos inteligentes, como *smartphone* e *smartTV*, possibilitando fornecer a informação ao seu proprietário de acordo com a demanda.

Por outro lado, essa emergente exposição faz com o ecossistema *Web* sofra um número cada vez maior de ameaças[1] e riscos[2] à segurança. De acordo com a Symantec [26], o número de ataques a serviços na *Web* cresceu mais de 30% entre 2015 e 2016. A WhiteHat [29] estima que, em média, 50% dos *sites* possuem vulnerabilidades a serem exploradas.

Esse grande número de ameaças muitas vezes é ocasionado pelo uso indiscriminado de *framework* vulneráveis ou padrões de codificação inseguros por parte dos desenvolvedores do serviço. Em outras palavras, o processo de desenvolvimento do serviço muitas vezes é conduzido sem um devido aprofundamento sobre às ameaças existentes no ecossistema [7]. Além disso, não se pode descartar os problemas ligados ao navegador *Web* ou demais dispositivos de consumo, que são mecanismos preparados para oferecer tudo que o usuário precisa para acessar os dados do serviço, porém, desenvolvidos por terceiros e, em relação ao serviço em questão, sem garantias efetivas de segurança. Outro viés é o fator humano (usuários, administradores do serviço e terceiros), que é naturalmente propenso a erros e descuidos [9].

Uma das formas de combater tais ameaças é utilizar uma Modelagem de Ameaças (*Threat Modeling* - TM), que se traduz como um modelo que auxilia na identificação, quantificação e avaliação de ameaças através de uma classificação bem definida [19]. Não obstante, não é incomum combinar o uso da TM com uma Modelagem de Risco (*Risk Modeling* - RM), que visa mitigar e mensurar riscos resultantes de uma ou mais ameaças [19]. A aplicação de tais modelos tem sido tratada no âmbito do desenvolvimento seguro de *software* uma vez que os fornecedores de serviço na *Web* são cada vez mais pressionados a adotarem um desenvolvimento eficiente em relação a segurança dos dados tidos como sensíveis [7].

No entanto, apesar de existirem várias propostas na literatura [1, 23, 28], a aplicabilidade desses modelos tem seus entraves que dificultam a sinergia com as equipes de desenvolvimento de *software*, sendo possível destacar três problemáticas. A primeira é sobre o **domínio de atuação** do serviço. De fato, é interessante considerar um certo nível de abstração na modelagem de ameaças e riscos a fim

In:XVII Workshop de Teses de Dissertações (WTD 2017), Gramado, Brasil. Anais do XXIII Simpósio Brasileiro de Sistemas Multimídia e Web: Workshops e Pôsteres. Porto Alegre: Sociedade Brasileira de Computação, 2017.

ISBN 978-85-7669-380-2.

[1] A norma ISO/IEC 27000:2013 define ameaça como a causa potencial de um incidente indesejado, que pode resultar em dano para um sistema ou para a organização.
[2] A norma ISO/IEC 27000:2013 define risco como a chance de uma ameaça se consolidar (probabilidade), resultando em um evento indesejado e possíveis consequências para um sistema ou para a organização.

de contemplar serviços em segmentos distintos. Contudo, em uma única ameaça existem muitos aspectos a serem considerados, como atores, dispositivos e comportamentos envolvidos, o que sugere uma moderação na abstração.

O problema é que certas modelagens tem sua abordagem voltada para o propósito geral [23, 28] com alta abstração e baixa profundidade, o que põe em cheque sua aplicabilidade sobre ameaças e riscos intrínsecos da *Web*, já que seu nível de abstração dificulta a identificação de características como a semântica e a similaridades de ataques que partem de uma ameaça. As similaridades nas variações dos ataques podem ser observadas em ataques do tipo *Buffer Overflow* (BoF) [15], que atuam em diferentes estruturas de dados, mas compartilham o mesmo vetor de ataque e quando explorados reproduzem impactos similares nos ativos.

A segunda problemática diz respeito à **adesão aos modelos** em metodologias de desenvolvimento. As modelagens, em sua maioria, são direcionadas ao processo convencional [23, 28] como o modelo em cascata [20], onde para avançar a uma nova atividade é necessário que a antecessora deva estar totalmente desenvolvida. Diante disso, a utilização dos modelos fica restrita à atividades como *projeto* ou codificação, já que o desenvolvedor terá em mãos requisitos com especificações imutáveis, sugerindo um processo de mão única. Consequentemente, não sugere práticas nas demais fases, apesar de também serem suscetíveis as ameaças e riscos, tornando a eficiência dos modelos questionável.

O problema ocorre também nas metodologias ágeis, devido a natureza pouco burocrática e enxuta dos artefatos dessas equipes [25]. Além disso, neste tipo de desenvolvimento, as mudanças são esperadas a todo o momento, o que implica dizer que as fases antecessoras e sucessoras trabalham continuamente de forma iterativa e incremental [24], gerando resistência as modelagens disponíveis.

Por fim, a terceira problemática remete a **avaliação de impacto** sobre os ativos. É de conhecimento geral que o impacto provocado por um ataque em sistemas que manipulam dados comuns não pode ter um peso equivalente ao de um sistema com dados sensíveis. Porém, o que chama atenção é que na literatura alguns modelos de mitigação de ameaça e risco tratam o ataque com um peso predefinido de forma generalizada [16, 18], indiferente do domínio do serviço. Tal contexto deveria permitir que a equipe de desenvolvimento definisse a criticidade do ataque com base nos comportamentos do seu serviço e analisasse o prejuízo considerando não apenas a confidencialidade dos dados, mas também o fator humano privacidade e os aspectos técnicos e legais que possam trazer consequências ao serviço.

É neste contexto que a proposta objetiva elaborar uma arquitetura de referência que faz uso de uma TM e RM específica para os serviços atuantes no ecossistema Web. Diferente dos modelos convencionais, que são centrados no atacante, nos ativos ou no *software*, a arquitetura possui uma ótica hibrida, englobando essas três visões, resultando em uma abordagem centrada em comportamentos. Também tem o intuito de ser um modelo de documentação leve, para garantir maior aderência no processo de desenvolvimento em equipes de natureza ágil. Por fim, por não ser de propósito geral, é capaz de considerar comportamentos dos atores e recursos intrínsecos da *Web*, além de fatores como engenharia social, propagação das ameaças, similaridades dos ataques e impactos aos ativos.

2 FUNDAMENTAÇÃO TEÓRICA E TRABALHOS RELACIONADOS

Nesta seção, serão descritos trabalhos da literatura que possuem soluções correlatas à proposta deste estudo. Uma das motivações para o desenvolvimento da proposta foi minimizar a carência de soluções dessa natureza considerando a Web como um ecossistema. Além disso, é pretendido considerar três problemáticas, conforme citado na seção anterior: (i) **domínio de atuação**, (ii) **adesão aos modelos** e (iii) **avaliação de impacto**.

No que tange o **domínio de atuação**, os trabalhos propõem soluções generalizadas, como falhas em sistemas de informação [11] ou operacionais [4]. Este tipo de proposta tem benefícios e serve como base para correlatos - taxonomias para sistema operacional UNIX [3, 4], engenharia social [9], computação em nuvem [2] e Internet das Coisas [17]. Entretanto, tais abordagens se enviesam no fator conceitual, sem identificar comportamentos intrínsecos da ameaça.

Na mesma linha, outros trabalhos apresentam identificação e propagação do ataque e comportamentos em comum com ameaças distintas, a exemplo da capacidade em descrever o como, quando e onde uma ameaça é atuante [13], do método utilizado [10], de investigar implementações inseguras em padrões de falhas na codificação [1, 27], e ainda, abordar sobre a perspectiva do atacante [5]. Contudo, apesar de cobrir a propagação, essas abordagens não relacionam similaridades compartilhadas entre ataques distintos. Na proposta da tese, esse compartilhamento permite auxiliar na prevenção de ameaças, pois facilita a distribuição das responsabilidades em identificar diversos vetores oriundos de uma única ameaça, mitigando vulnerabilidades através de uma abordagem guiada por aspectos semânticos.

Quanto a **adesão aos modelos**, descreve a adoção de modelagens de ameças e riscos no desenvolvimento de *software*. Modelos de TM e RM são tópicos bastante explorados, a exemplo da STRIDE [23] que é uma modelagem centrada em *software* ou o *Common Attack Pattern Enumeration and Classification* (CAPEC) [16] que auxilia na quantificação da ameaça. O grande entrave é que modelagens desta natureza são fundamentadas pelo modelo cascata, o que dificulta a adoção de um modelo incremental que favoreça a relação entre papéis distintos nas fases do desenvolvimento. Outro fator é que embora haja uma considerável cobertura quanto ao número de ataques, tais modelos induzem alta abstração e baixa granularidade em determinadas variações de ataques.

Alguns trabalhos trazem apoio ao desenvolvedor, descrevendo o como, quando e onde uma ameaça é atuante [13] ou buscam investigar implementações inseguras baseando-se em padrões de falhas na codificação [1, 27]. Contudo, apesar de cobrir a propagação, essas abordagens não relacionam similaridades compartilhadas entre ataques distintos. O principal benefício disso seria auxiliar na prevenção, pois facilita na distribuição das responsabilidades em identificar diversos vetores oriundos de uma única ameaça.

Por fim, em **avaliação de impacto** relata a capacidade de analisar riscos e impactos aos ativos. Alguns trabalhos propõem soluções baseadas em cenários de codificação de *software* [27] ou aspectos entre os computadores e a rede [8]. O principal diferencial da proposta é que sugerimos maior granularidade no modelo de classificação, através da ramificação em vetores. Por estabelecer um agrupamento

semântico, oferece maior precisão sobre a propagação da exploração e a identificação dos atributos de segurança comprometidos, mensurando o impacto aos ativos de forma qualitativa e quantitativa, com base nas ameaças.

Propostas de RM também são tópicos bem debatidos na literatura, a exemplo da modelagem DREAD [23], que estima o risco fazendo perguntas em cada uma das categorias, com base na ótica do atacante. Contudo, o DREAD, no molde inicialmente proposto, em 2010 foi desaconselhado por seus autores [21] devido falta de precisão em determinadas situações. Contudo, o mesmo recebeu melhorias quanto ao nível de riscos com base na causa e efeito [22]. Outro exemplo é a modelagem PASTA [28] que é focada em ativos e provê um processo que considera possíveis cenários de ataque e vulnerabilidades e ataques, mitigando níveis de risco e impacto. Contudo, ambas abordagens são direcionadas ao desenvolvimento convencional, sem uma abordagem com maior consonância ao desenvolvimento ágil.

3 CONTRIBUIÇÕES ESPERADAS

A proposta pretende oferecer uma arquitetura de referência centrada em comportamentos de ameaças e riscos, que consolida-se em um processo guiado com enfase no desenvolvimento seguro de comportamentos **(development)**, potencialmente candidatos, à serem incorporados ao serviço. Ela visa nortear **(driven)** equipe ágeis nas atividades associadas a especificação, implementação, verificação e validação. Conforme a Figura 1, a construção da proposta envolve diversos artefatos, em que a arquitetura de referência apresenta-se como um artefato macro, composto por sub-artefatos (TM e RM), além de atividades que investigam evidências na literatura para as decisões da proposta.

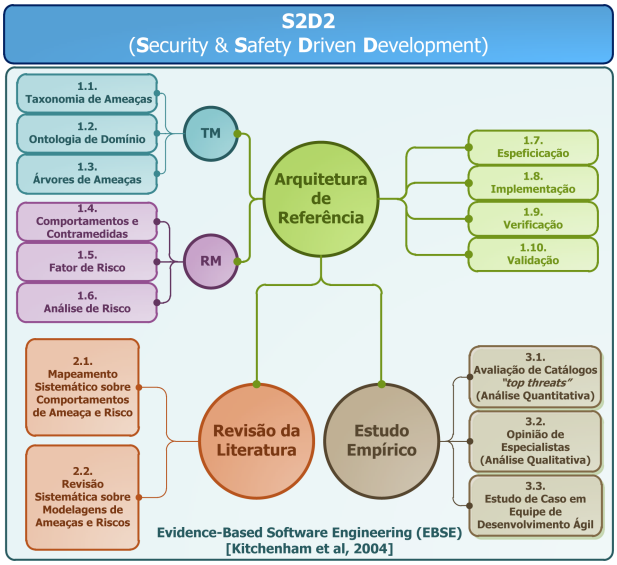

Figura 1: Artefatos produzidos pela Proposta

No que tange ao risco, a proposta considera que nem todas as prevenções estabelecidas em resposta aos incidentes serão bem sucedidas. Diante disso, como determinados riscos devem ser considerados, é interessante estabelecer condutas para quando uma ameaça for concretizada, visando minimizar o prejuízo resultante. Portanto, a RM apresenta **contramedidas** em consonância à riscos comumente observados em serviços na *Web*, caracterizando como uma abordagem que contempla resiliência **(safety)** ao serviço. No que versa sobre as ameaças, a proposta faz uso da TM para delegar **prevenções** associadas à ameaças comumente observadas em serviços na *Web*, caracterizando a proposta como uma abordagem que trás resistência **(security)** ao serviço. O significado das palavras-chave são detalhados a seguir:

- **Security**: Meios de proteger contra defeitos, perdas, criminosos e demais indivíduos ou ações que proporcionem uma ameaça (externos ou internos). Meios de proporcionar resistência para que não ocorra incidentes (consequências indesejáveis).
- **Safety**: Meios de minimizar impactos causados por ameaças exploradas no serviço. Meios de provê resiliência em casos de incidentes. Alcançar um nível aceitável de risco.
- **Driven**: Define o processo como uma metodologia baseada em comportamentos de ameaças e riscos. O objetivo é direcionar a equipe de desenvolvimento relacionando as funcionalidades do serviço com seus respectivos comportamentos de ameaças e riscos. Caracterizando a proposta como um processo guiado.
- **Development**: A metodologia guia o desenvolvimento de *software* considerando suas fases e atividades. Estabelece uma metodologia de boas práticas que considera ações em cada atividade.

Por ser baseada em comportamentos, a proposta é compatível com o *Behavior-Driven Development (BDD)*, que se traduz como uma técnica de desenvolvimento ágil que encoraja os envolvidos a descrever o *software* com base em comportamentos e incentiva o foco no código-fonte e uso de simuladores de testes [24]. Em posse do modelo, o desenvolvedor especifica testes com base na TM, e executa-os em um desenvolvimento guiado à testes, como o *Test-Driven Development* (TDD). Considerando o contexto do BDD, a proposta visa oferecer um processo guiado que pode ser dividido em quatro características, a saber:

- **Especificação**: Apoio na elaboração das estórias de usuário, prototipação e decisões arquiteturais do serviço. Essa característica visa nortear a equipe durante os requisitos e *projeto* do serviço.
- **Implementação**: Apoio nas decisões arquiteturais, padrões de codificação e testes de unidade. Essa característica visa nortear a equipe durante o *projeto* e codificação do serviço.
- **Verificação**: Apoio nos testes de integração, sistema e regressão. Essa característica visa nortear a equipe durante os testes funcionais do serviço. Como resultado, visa avaliar a eficiência das soluções desenvolvidas no serviço que garantem resistência e resiliência.
- **Validação**: Apoio nos testes de aceitação. Essa característica visa nortear a equipe durante o processo de implantação do serviço. Como resultado, visa avaliar se as soluções desenvolvidas no serviço que garantem resistência e resiliência de fato correspondem com as reais necessidades do serviço.

3.1 Arquitetura de Referência

As modelagens de Ameaças e Riscos servirão de apoio na elaboração de uma Arquitetura de Referência (*Reference Architecture*), para o desenvolvimento de serviços *REST* alinhado à processos ágeis que envolvem comportamentos, como o BDD; e também guiada à testes, como o TDD. Ambas metodologias garantem a compreensão dos testes de forma ubíqua junto a equipe de desenvolvimento. Uma vez que o TDD tem sua ótica voltada ao código-fonte, acaba relacionando o desenvolvedor e testador. O BDD abrange esse entendimento para os demais envolvidos, inclusive o próprio cliente do *software* em questão, pois associa os testes com uma documentação de maior abstração e não intrínseca ao código-fonte [24].

Quanto a decisão de enveredar a proposta no estilo arquitetural *REST* foi devido ao tamanho interesse dessa tecnologia, em comparação ao SOAP, pelo desenvolvimento de serviços na Web nos últimos 5 anos[3]. Conforme ilustrado na Figura 2, o estilo arquitetural *REST* possui particularidades que serão guiadas considerando o cerne do desenvolvimento seguro para as equipe ágeis.

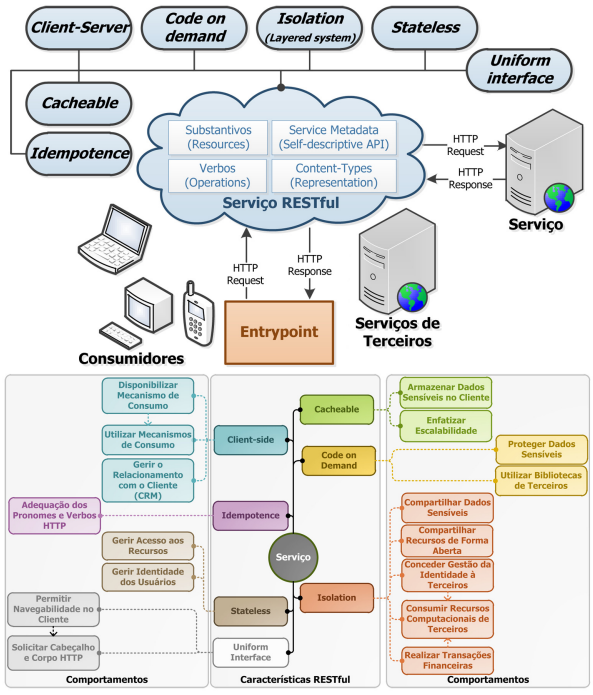

Figura 2: Características e Comportamentos de um serviço *RESTful*

Conforme especificado na obra original[4], são definidos diversos conjuntos de características e elementos no modelo arquitetural *REST*. Uma síntese sobre a relação das características e respectivos comportamentos é descrita a seguir:

- **Cacheable**: Armazenamento de dados no mecanismo de consumo. Aqui serão apresentadas boas práticas quanto ao armazenamento de dados sensíveis no lado do cliente e escalabilidade nas requisições para redução da latência e aumento da disponibilidade.
- **Client-side**: Modelo arquitetural para separação de interesses. Aqui serão abordadas técnicas para reconhecimento e controle de acesso através de dispositivos (*TOKEN*), além das preocupações sobre os termos de uso para o usuário final.
- **Code on Demand**: Personalizar dinamicamente as funcionalidades no cliente (mobile app, web app, etc). Nesse ponto serão levantadas propostas para a codificação segura dos dados sensíveis, considerando boas práticas para evitar vulnerabilidades em mecanismos de terceiros que atuam no consumo *client-side*, como *plug-ins* e extensões do navegador. Também será descrita a importância e implementação de uma eficiente rotina para registro de evidências na aplicação, considerando privacidade, anonimato e não-repúdio.
- **Idempotence**: O recurso fornecerá o mesmo resultado sempre que for solicitado. Aqui serão relatadas boas práticas durante o uso de verbos e pronomes oferecidos pelo *REST* sobre o HTTP.
- **Isolation**: Alto nível de abstração que impossibilita identificar se a requisição é direta ao serviço final ou terceiros. Neste aspecto, serão descritas boas práticas sobre o compartilhamento de dados sensíveis e recursos computacionais, a exemplo de autenticações ou *storage* por terceiros. Além de abordar técnicas de *hardening* para possíveis explorações de recursos disponibilizados de forma aberta.
- **Stateless**: Os pedidos possuem toda a informação necessária para atender a solicitação. Aqui serão abordadas, em enfase, sobre as técnicas de autenticação e autorização ao serviço, considerando o padrão de comportamento do HTTP quanto a suas requisições e respostas, além de técnicas auxiliares que atuam no lado cliente, como *Same-Origin Policy* (SOP) e *Cross-origin resource sharing* (CORS).
- **Uniform interface**: Simplificar e desacoplar a arquitetura, propondo módulos independentes. Nesse ponto serão tratadas as técnicas no tratamento dos parâmetros da requisição e sanitização das entradas.

Considerando o ciclo de vida do desenvolvimento, *Software Development Life Cycle* (SDLC) [20], a arquitetura dispõe de diferentes mecanismos para cada atividade, resultando em um processo guiado que contempla todo o ciclo, dividindo-se entre as fases com práticas que auxiliam na especificação, implementação, verificação e validação, conforme ilustrado na Figura 3. Por exemplo, na fase de requisitos, o modelo oferece suporte para especificação de atividades como estórias de usuário e prototipação.

Importante frisar que os termos verificação e validação, conforme empregado na literatura [20], no contexto da proposta da tese, a verificação remete a inspeção em cada mecanismo de proteção implementado, verificando sua eficiência no cenário (*RESTful*) através de testes funcionais com base nas ameaças identificadas pela modelagem. Já a validação remete a um processo empírico, através de ambientes controlados com ameaças simuladas, no intuito de observar se os mecanismos implementados atendem as expectativas.

[3]Fonte do Google Thrends: https://goo.gl/Hz2jWn
[4]Representational State Transfer (REST): https://goo.gl/2ulgaH

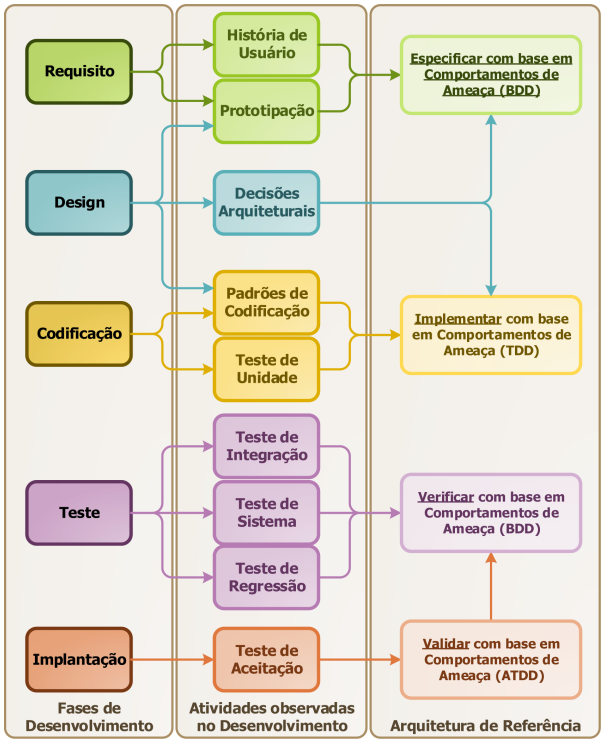

Figura 3: Processo guiado durante as fases do ciclo de vida (SDLC)

4 ESTADO ATUAL DO TRABALHO

Considerando a Figura 1, atualmente a tese encontra-se com as atividades 1.1 até 1.10, 2.1 e 3.1 concluídas. A atividade 2.2 será atualizada no inicio de 2018, visando contemplar na mesma os artigos publicados no ano de 2017. Existe um documento disponível publicamente que relata maiores detalhes sobre a construção da TM[5] e RM[6].

Na atividade 3.2 será pretendida uma análise qualitativa da proposta com base na opinião de especialistas. Tal método é proposto em [12] descrito como DESMET 8, do tipo *Hybrid method 1*, onde especialistas avaliam a proposta com base de resultados obtidos em algum método quantitativo. Como já existe resultados neste aspecto através da atividade 3.1, se fez interessante o uso do DESMET 8 para a etapa 3.2. Atualmente, o SCS[7] é o grupo de especialistas a ser submetido para avaliação. As atividades, com duração de três meses, serão realizadas como um doutorado sanduíche no segundo semestre de 2017.

Por fim, na etapa 3.3 será desenvolvido um Estudo de Caso que, conforme classificado em [12] DESMET 6, do tipo *Qualitative case study*, define-se como uma avaliação por um indivíduo ou organização que tenha utilizado o método em um projeto real. O objetivo é fazer uso do S2D2 *in loco* em uma organização que desenvolva soluções de tecnologia da informação. O local a ser aplicado o estudo será no PRODAP[8], uma entidade que desenvolve soluções baseadas na Web e sua equipe faz uso de metodologias ágeis.

De posse destes dois artefatos, se faz possível extrair uma quantidade de dados satisfatória para obter maior solidez nas hipóteses da tese através de uma teoria fundamentada em dados, proposta em [6]. Essa metodologia sugere a validação de teorias e modelos através de um estudo de caso. Todas as atividades, tanto as já concluídas, em progressão, ou as ainda a realizar, podem ser visualizadas no cronograma ilustrado na Figura 4. As atividades estão enumeradas conforme especificado na Figura 1.

Atividades	JAN	FEV	MAR	ABR	MAI	JUN	JUL	AGO	SET	OUT	NOV	DEZ	ANO
Escrita								X		X		X	
1.1									X	X		X	2014
2.1									X	X	X	X	
Escrita	X	X									X	X	
2.1			X	X	X								
1.1					X	X	X	X					
1.2							X	X					2015
1.3								X	X	X			
1.4										X	X		
2.2									X	X	X	X	
Escrita	X	X			X	X	X				X	X	
1.5		X	X	X									
1.6		X	X										
3.1		X	X	X									
2.2								X	X	X			2016
1.7									X	X	X	X	
1.8										X	X	X	
1.9											X	X	
Escrita	X	X					X	X			X	X	
1.9	X	X	X										
1.10		X	X	X	X	X							2017
3.2									X	X	X	X	
Escrita	X	X				X	X	X					
2.2	X	X											2018
3.3			X	X	X	X	X	X					

Figura 4: Cronograma das atividades da tese

5 AVALIAÇÃO DOS RESULTADOS

Como medida de validação do estado atual da proposta, decidiu-se empregá-la como: (i) um modelo formal e metódico para identificar e classificar as ameaças e (ii) uma ferramenta para o desenvolvimento ágil que ofereça previsibilidade das ameaças e controle dos riscos.

5.1 Modelo Formal: Identificação e Classificação das Ameaças e Riscos

A avaliação da TM e RM como modelos formais visa estabelecer classificações facetadas de diversos ataques publicados na literatura, identificando os atributos de segurança violados ou não por um ataque. Esse tipo de avaliação tem o intuito de identificar similaridades entre categorias de um determinado assunto, possibilitando uma classificação de acordo com suas características e comportamentos.

Como ponto de partida, realizou-se um levantamento de catálogos de ataques registrados na literatura. Nesta linha, catálogos de ameaças são geralmente os mais recomendados por apresentarem os ataques emergentes. Esse tipo de catálogo têm o ponto forte de abordar os ataques mais emergentes da atualidade. Porém, por se

[5]Documento sobre a TM: https://goo.gl/Yhzrra
[6]Documento sobre a RM: https://goo.gl/jg3zbb
[7]Services, Cybersecurity and Safety researchgroup (SCS): Grupo de pesquisa da universidade de Twente (Holanda) com foco em avaliação de níveis de segurança de serviços

[8]Processamento de Dados do Amapá (PRODAP): É a uma autarquia com atuação na área de tecnologia de informação e comunicação do governo do Amapá

enviesar em muita objetividade, ele oferece uma abordagem pouco quantitativa, o que dificultou apresentar todas as possíveis classificações previstas na TM. Esse fato motiva a buscar por catálogos com maior número de ameaças a fim de avaliar com maior precisão o modelo de classificação. Diante disso, foi desenvolvido uma fusão do catálogo do OWASP [18] com outros 6 catálogos disponíveis na literatura. Através dessa avaliação adicional, o estudo classificou um total de 251 ataques[9]. A compilação dos catálogos proporcionou uma avaliação com maior precisão do modelo.

5.2 Ferramenta Ágil: Testes com BDD e TDD

A intenção desta segunda validação é demonstrar a possibilidade de usar a TM proposta como uma ferramenta eficaz para o desenvolvimento ágil, provendo mecanismos que atuam diretamente no código-fonte, com objetivo de auxiliar a equipe de desenvolvimento na implementação de prevenções (*security*) e contramedidas (*safety*).

Desta forma, a TM e RM propostas podem ser implementadas em uma biblioteca independente, separando os interesses com o mínimo de acoplamento através de orientação a aspectos[10]. Por exemplo, é possível que o desenvolvedor informe, através de uma anotação, qual atributo de uma classe ou elemento JSON é considerado sensível. Uma rotina pode ser disparada sempre que o atributo marcado como sensível for acessado, evitando a exposição indevida do mesmo para o caso de ser enviado acidentalmente na resposta da requisição. Na mesma linha, é possível atribuir uma anotação que informe a biblioteca que um determinado método irá submeter um formulário, acionando filtros de sanitização nas entradas do formulário. Para maiores detalhes, é possível obter a codificação em um repositório público no *GitHub* [11].

Diante disso, se faz possível uma adaptação nas ferramentas JBehave[12] e Serenity BDD[13] em consonância aos comportamentos e ameaças da proposta. Os testes são especificados através de documentos similares a *User Stories*, mas destinados à elicitar comportamentos maliciosos. Tais documentos são definidos como *Attacker Story*, para a exploração de vulnerabilidades, e *Trickster Story* para a elicitação de possíveis fraudes. Para maiores detalhes, é possível obter a codificação em um repositório público no *GitHub*[14].

REFERÊNCIAS

[1] Gonzalo Alvarez and Slobodan Petrovic. 2003. A new taxonomy of Web attacks suitable for efficient encoding. *Computers & Security* 22, 5 (2003), 435–449. http://dblp.uni-trier.de/db/journals/compsec/compsec22.html#AlvarezP03

[2] A Amini, N Jamil, AR Ahmad, and MR Zaba. 2015. Threat Modeling Approaches for Securing Cloud Computin. *Journal of Applied Sciences* 15 (2015), 953–967.

[3] Taimur Aslam. 1995. A Taxonomy of Security Faults in the Unix Operating System. (1995).

[4] Matt Bishop. 1995. *A Taxonomy of UNIX System and Network Vulnerabilities.* Technical Report. University of California.

[5] M. A. Douad and Y. Dahmani. 2015. ARTT taxonomy and cyber-attack Framewok. In *New Technologies of Information and Communication.* https://doi.org/10.1109/NTIC.2015.7368742

[6] Kathleen M. Eisenhardt. 1989. Building Theories from Case Study Research. *Academy of Management Review* 14, 4 (1989), 532–550. https://doi.org/10.5465/AMR.1989.4308385

[7] Eduardo Fernandez-Buglioni. 2013. *Security Patterns in Practice: Designing Secure Architectures Using Software Patterns* (1st ed.). Wiley Publishing.

[8] Simon Hansman and Ray Hunt. 2005. A taxonomy of network and computer attacks. *Computers & Security* 24, 1 (2005). http://dblp.uni-trier.de/db/journals/compsec/compsec24.html#HansmanH05

[9] Koteswara Ivaturi and Lech Janczewski. 2011. A Taxonomy for Social Engineering attacks. *International Conference on Information Resources Management* (2011).

[10] Andy Jones. 2002. Identification of a Method for the Calculation of Threat. In *Internal publication.*

[11] Mouna Jouini, Latifa Ben Arfa Rabai, and Anis Ben Aissa. 2014. Classification of Security Threats in Information Systems. In *Proceedings of the 5th International Conference on Ambient Systems, Networks and Technologies (ANT).*

[12] B. Kitchenham, S. Linkman, and D. Law. 1997. DESMET: a methodology for evaluating software engineering methods and tools. *Computing Control Engineering Journal* 8, 3 (June 1997), 120–126. https://doi.org/10.1049/cce:19970304

[13] Carl E. Landwehr, Alan R. Bull, John P. Mcdermott, William, and S. Choi. 1994. A Taxonomy of Computer Program Security Flaws. *Comput. Surveys* 26 (1994), 211–254.

[14] Marc Lankhorst. 2012. *Agile Service Development: Combining Adaptive Methods and Flexible Solutions.* Springer Publishing Company, Incorporated.

[15] MITRE. 2011. Top 25 Most Dangerous Software Errors. *Disponível em: http://cwe.mitre.org/top25/* (2011).

[16] MITRE. 2015. Common Attack Pattern Enumeration and Classification (CAPEC). *Disponível em: https://capec.mitre.org/* (2015).

[17] Ricardo Neisse, Gary Steri, Igor Nai Fovino, and Gianmarco Baldini. 2015. SecKit: A Model-based Security Toolkit for the Internet of Things. *Computers & Security* 54 (2015), 60 – 76. https://doi.org/10.1016/j.cose.2015.06.002

[18] OWASP. 2013. Top Ten 2013. *Disponível em: https://goo.gl/VKz94B* (2013).

[19] OWASP. 2016. Threat Modeling Cheat Sheet. *Disponível em: https://goo.gl/tgn772* (2016).

[20] Roger S. Pressman. 2001. *Software Engineering: A Practitioner's Approach* (5th ed.). McGraw-Hill Higher Education.

[21] Microsoft SDL. 2010. Appendix N: SDL Security Bug Bar (Sample). *Disponível em: https://goo.gl/USXuBM* (2010).

[22] Microsoft SDL. 2010. Security Briefs - Add a Security Bug Bar to Microsoft Team Foundation Server 2010. *Disponível em: https://goo.gl/Qv3smB* (2010).

[23] Adam Shostack. 2014. *Threat Modeling: Designing for Security.* Wiley, 1 edition.

[24] S. Sivanandan and Yogeesha C. B. 2014. Agile development cycle: Approach to design an effective Model Based Testing with Behaviour driven automation framework. In *Advanced Computing and Communications.* 22–25. https://doi.org/10.1109/ADCOM.2014.7103243

[25] C. J. Stettina, W. Heijstek, and T. E. Faegri. 2012. Documentation Work in Agile Teams: The Role of Documentation Formalism in Achieving a Sustainable Practice. In *Agile Conference (AGILE).*

[26] Symantec. 2016. Internet Security Threat Report. *Disponível em: https://goo.gl/nrFcoA* (2016).

[27] Katrina Tsipenyuk, Brian Chess, and Gary McGraw. 2005. Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors. *IEEE Security & Privacy* 3, 6 (2005), 81–84. http://dblp.uni-trier.de/db/journals/ieeesp/ieeesp3.html#TsipenyukCM05

[28] Tony UcedaVelez and Marco Morana. 2015. *Risk Centric Threat Modeling: Process for Attack Simulation and Threat Analysis.* Wiley, 1 edition.

[29] WhiteHat. 2016. Security Predictions 2017. *Disponível em: https://goo.gl/f94Qq8* (2016).

[9] Avaliação adicional de 251 ataques: https://goo.gl/BlEQXd

[10] AspectJ: https://eclipse.org/aspectj/

[11] Projeto exemplo com AspectJ: https://goo.gl/L9TkhE

[12] JBehave: http://jbehave.org/

[13] Serenity BDD: http://thucydides.info

[14] Projeto exemplo com BDD: https://goo.gl/9Yjsid