Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Abordagem Ontológica para Mitigação de Riscos em Aplicações Web

Marcius Montedo Marques

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Orientadora
Profª. Drª. Célia Ghedini Ralha

Brasília
2014

Universidade de Brasília – UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Mestrado em Informática

Coordenadora: Profª. Drª. Alba Cristina Magalhaes Alves de Melo

Banca examinadora composta por:

Profª. Drª. Célia Ghedini Ralha (Orientadora) – CIC/UnB
Prof. Dr. Fernando Magno Quintão Pereira – DCC/UFMG
Dr. Auto Tavares da Câmara Júnior – DPF

CIP – Catalogação Internacional na Publicação

Marques, Marcius Montedo.

Abordagem Ontológica para Mitigação de Riscos em Aplicações Web / Marcius Montedo Marques. Brasília : UnB, 2014.
78 p. : il. ; 29,5 cm.

Dissertação (Mestre) – Universidade de Brasília, Brasília, 2014.

1. Segurança da informação, 2. ontologia, 3. aplicações web, 4. riscos, 5. vulnerabilidades

CDU 004

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro – Asa Norte
CEP 70910–900
Brasília – DF – Brasil

Dedicatória

À minha vovó Eny, por cuidar de mim em todos os sentidos desde antes do meu nascimento até os dias atuais.

Agradecimentos

Agradeço aos meus pais, Dina e Fernando, por me ensinarem a trilhar o meu próprio caminho através do conhecimento.

Aos meus filhos, Laura e Daniel, pelos quais me esforçarei a fazer o mesmo.

À minha mulher, Márcia, que ficou com eles durante todo o tempo em que precisei me dedicar a esse estudo.

À orientadora Prof. Célia, por ter acolhido e acreditado no projeto e pela confiança depositada em mim, permitindo que, com liberdade, o trabalho fosse construído. Agradeço todo o apoio, paciência e conselhos ao longo do processo.

Aos Professores Gondim e Auto Júnior, pelas valiosas contribuições durante o processo de evolução do trabalho.

Ao amigo Daniel Souza, pela ajuda e estudos em todas as disciplinas cursadas.

Obrigado!

"A educação faz com que as pessoas sejam fáceis de guiar, mas difíceis de arrastar; fáceis de governar, mas impossíveis de escravizar."

Peter Drucker

Resumo

Segurança da Informação (SegInfo) está se tornando um quesito de alta prioridade no suporte às atividades de negócios, a medida que as organizações se esforçam para que seus dados fiquem disponíveis e seguros em aplicações *web*. Entretanto, a segurança não é uma preocupação presente desde o início do processo de desenvolvimento, principalmente porque os desenvolvedores não são especialistas no assunto. Consequentemente, sistemas vulneráveis são projetados e, quando atacados, podem comprometer os dados e operações das organizações, resultando em grandes perdas financeiras. Em uma pesquisa realizada com mais de 200 desenvolvedores de aplicativos, verificou-se que, apesar de perceberem o quão importante é o seu papel no processo de garantia da segurança, a grande maioria não está interessada em aprender os detalhes necessários para desenvolver soluções seguras. Como a maioria dos ataques miram a camada de aplicação, propomos uma abordagem inteligente baseada em ontologia para mitigar os riscos em aplicações web. Este tipo de abordagem não requer que os desenvolvedores frequentem cursos de segurança de longa duração, ou leiam extensos livros e pesquisem diversas páginas sobre SegInfo para adquirir os conhecimentos necessários para produzir aplicações mais seguras. Uma abordagem ontológica também contribui para a disseminação do conhecimento sobre SegInfo e reduz o trabalho de implementação de aplicações *web* seguras nas organizações. A base de conhecimento para construção da ontologia se fundamenta em três repositórios conhecidos que tratam de vulnerabilidades: OWASP [1] Top 10, OWASP ASVS [2] e CWE[3]. Eles são combinados e aplicados para reduzir a lacuna entre o desenvolvedor e as informações relacionadas à segurança. O modelo proposto é aplicado na fase de projeto do desenvolvimento em diversos casos reais, tendo como resultado aplicações web mais seguras. A ontologia, extensível e reutilizável, é avaliada quantitativamente e qualitativamente para efeitos de comparação. Os resultados mostram que as vulnerabilidades podem ser reduzidas por meio do aumento direcionado da conscientização sobre segurança dos desenvolvedores *web* durante o processo de desenvolvimento das aplicações.

Palavras-chave: Segurança da informação, ontologia, aplicações web, riscos, vulnerabilidades

[1] Open Web Application Security Project - https://www.owasp.org
[2] Application Security Verification Standard - https://www.owasp.org/
[3] Common Weakness Enumeration - http://cwe.mitre.org/

Abstract

Information Security (InfoSec) is becoming a high priority asset to support business activities, as organizations struggle to assure that data is available and secure in web applications. However, security is not a concern from the beginning of the development process, mainly because developers are not security specialists. Consequently, vulnerable systems are designed and when attacked can compromise organization's data and operations, enclosing high financial losses. On a survey performed with more than 200 application's developers, it was found that although they realize how important is their role in the security assurance process, the huge majority is not interested in learning security in depth to develop solutions. Because most attacks target the application layer, we propose an intelligent approach based on ontology to mitigate risks in web applications. This type of approach does not require developers to go through long time consuming courses, books or different sites about InfoSec in order to acquire the needed knowledge to produce more secure applications. An ontological approach can also contribute to InfoSec knowledge dissemination and reduce the burden of implementing secure web applications on organizations. The knowledge base to build the ontology is from three well known sources about vulnerabilities: OWASP Top 10, OWASP ASVS and CWE. They are merged and applied together to reduce the gap between the application developer and the security related information. The proposed model is employed in the development's design phase of several real case scenarios; with more secure web applications as the outcome. The extensible and reusable developed ontology is evaluated quantitatively and qualitatively for comparison purposes. The results show that vulnerabilities can be reduced by increasing the security awareness of web developers during the application development process.

Keywords: Information security, ontology, web applications, risks, vulnerabilities

Abreviaturas

ASVS - *Application Security Verification Standard.*

BPM - *Business Process Management.*

BS - *British Standard.*

CIA - *Confidentiality, Integrity, Availability.*

CLASP - *Comprehensive, Lightweight Application Security Project.*

CnA - *Certification and Accreditation.*

CSRF - *Cross-site request forgery.*

CWE - *Common Weakness Enumeration.*

InfoSec - *Information Security.*

LDAP - *Lightweight Directory Access Protocol.*

NSA - *National Security Agency.*

OWASP - *Open Web application Security Project.*

OWL - *Web Ontology Language.*

PII - *Personally Identifiable Information.*

RDF - *Resource Description Framework.*

SegInfo - Segurança da Informação.

SI - Sistemas de Informação.

SO - Sistema Operacional.

SQL - *Structured Query Language.*

TI - Tecnologia da Informação.

SDLC - *Software Development Lifecycle.*

SDL - *Secure Development Lifecycle.*

SLAC - *Service Level Agreement Count.*

SPARQL - *SPARQL Protocol and RDF Query Language.*

STRIDE - *Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege.*

W3C - *World Wide Web Consortium.*

XSS - *Cross-site scripting.*

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A dependência crescente das organizações em Sistemas de Informação (SI) para gerenciar suas atividades de negócio é um processo notadamente irreversível (Weske, 2007). A área de Tecnologia da Informação (TI), suportada por uma estrutura de conexão global cada vez mais eficiente, avança a passos largos, com organizações de todos os tipos buscando utilizar o poder da TI para alcançar melhores resultados. Seja por simples controles de vendas em planilhas de cálculo até complexos sistemas de automação de processos de negócio, a área de TI é considerada estratégica para as organizações, que estão cada vez mais conectadas e integradas.

Os gestores concordam que investir em TI pode trazer benefícios imediatos e compensadores relacionados a redução de custo e tempo de atividades chave. Eles empregam a tecnologia juntamente com a conectividade para compartilhar um volume cada vez maior de dados através de aplicações distribuídas, seja com clientes, fornecedores, empregados ou provedores de serviço.

Entretanto, quando se fala em Segurança da Informação (SegInfo), os mesmos gestores apresentam dificuldades para reconhecer a importância de investimentos para essa área. Na maioria das vezes, precisam ser convencidos por fatores econômicos, o que pode acontecer quando já é tarde demais (Gordon and Loeb, 2002). Mas com diversas notícias de problemas relacionados à falhas de segurança em aplicações sendo divulgadas, SegInfo tem recebido cada vez mais investimentos nas organizações, com um expressivo aumento de 51% na média dos orçamentos quando comparados números de 2013 e 2014 (PWC, 2014).

Apesar disso, as taxas de vulnerabilidades e perdas relacionadas à falhas em sistemas aumentam a cada ano, segundo dados da mesma pesquisa. Para ilustrar, em um estudo feito de 2000 a 2004 com mais de 250 aplicações *web* (incluindo aplicações financeiras e de comércio virtual), constatou-se que mais de 90% delas apresentavam vulnerabilidades a pelo menos um tipo de ataque. O número permaneceu inalterado para estudo semelhante cinco anos depois (PRNewswire, 2005).

A estimativa da quantificação das perdas relacionadas à vulnerabilidades é uma árdua tarefa, dado que as consequências de vazamento de informações podem ser exploradas ao longo de muitos anos. Segundo Kerr (2013), essas perdas podem variar entre US$ 300 bilhões e US$ 1 trilhão de dólares, com invasores roubando mais de um *terabyte* de informação por dia de aplicações vulneráveis.

Estudos de empresas de consultoria concluem que ataques a aplicações *web* são e ainda continuarão sendo por muitos anos um dos maiores desafios rotineiros a ser enfrentado pelas organizações (PWC, 2013). Estes estudos baseiam-se no fato de que, ainda que avanços importantes sejam feitos na área, eles não acompanham a evolução dos ataques. Os prejuízos possíveis tornaram-se sem precedentes, podendo gerar incidentes simples que envolvem a necessidade de reinstalação de um aplicativo ou indisponibilidade momentânea de um sistema, bem como impactar vidas humanas, como a perda de um sistema de geração de energia elétrica, de defesa, de transportes, entre outros.

As respostas de 9.600 executivos de 115 países para temas relacionados à SegInfo sugere que modelos antigos de proteção tem sido utilizados para combater ameaças cada vez mais sofisticadas, resultando em pouca eficácia na proteção efetiva das aplicações. Tradicionalmente, prefere-se uma abordagem reativa em relação à estratégia de SegInfo, que é deixada em segundo plano para que o foco sejam os desafios de TI, como se fossem nichos de atuação separáveis (PWC, 2014).

Nesta dissertação é apresentada uma abordagem inteligente que aposta no investimento no conhecimento dos desenvolvedores para melhorar a segurança das aplicações *web* por eles desenvolvidas. Esse conhecimento é estruturado por meio de uma ontologia que utiliza fontes reconhecidamente válidas sobre SegInfo, de forma que possa ser aplicada facilmente e gradativamente à rotina dos desenvolvedores. A proposta está relacionada à atividade de gerenciamento de riscos dentro dos conceitos de SegInfo, importante processo que é considerado requisito obrigatório na política de segurança das organizações (Peltier, 2013).

1.1 Caracterização do Problema

O cenário atual da evolução da tecnologia descrito na Seção 1 é caracterizado pela relação direta entre a conectividade e o compartilhamento com o nível de risco. O risco é tanto menor quanto menos sairmos do perímetro interno da organização, o que coloca as aplicações *web*, tendência de solução utilizada em todos os tipos de organização, no nível máximo de risco, conforme ilustra Sêmola et al. (2003) na Figura 1.1.

Segundo Sêmola et al. (2003), esse cenário é definido como uma 'receita explosiva' que ilustra muito bem o cerne do problema, caracterizada pelos seguintes itens:

- crescimento sistemático da digitalização das informações;
- crescimento exponencial da conectividade da empresa;
- crescimento das relações eletrônicas entre empresas;
- crescimento exponencial do compartilhamento de informações;
- barateamento do computador pela diminuição do custo do hardware;
- facilidade e proliferação do acesso à internet;

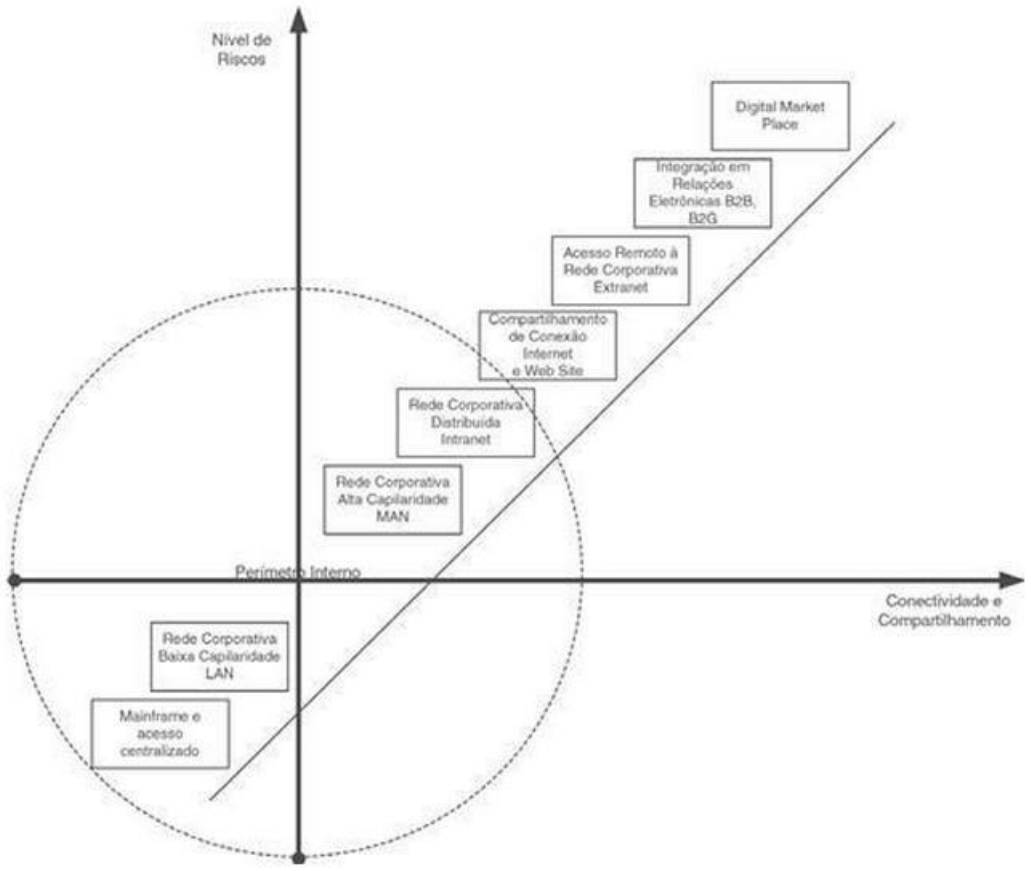

Figura 1.1: Evolução da conectividade e do compartilhamento - Fonte: adaptado de Sêmola et al. (2003)

- aumento da velocidade de acesso à internet;
- alto compartilhamento de técnicas de ataque e invasão;
- disponibilidade de grande número de ferramentas de ataque e invasão;
- facilidade de uso de ferramentas de ataque e invasão;
- carência de jurisprudência sobre atos ilícitos em meio eletrônico;
- crescente valorização da informação como principal ativo das organizações.

Apesar do cenário descrito por Sêmola et al. (2003), existem algumas iniciativas no sentido de regular os crimes cibernéticos, tais como Lei 12.732/12 (Carolina Dickeman) e 12.965/14 (Marco Civil). No entanto, a ausência de um programa de segurança para esse cenário de alto risco, trazem consequências imensuráveis para as organizações. O desenvolvimento de aplicações *web* mais seguras é uma pequena parte de um programa de segurança. Um programa completo de SegInfo deve ser abrangente para que atinja os resultados esperados. A implementação envolve técnicas de segmentos diversos como programas de anti-vírus,

firewalls, técnicas sofisticadas de encriptação, equipamentos e algoritmos detectores de intrusão, segurança física de localidades, processos de *backup*, planos de contingência e restauração, para citar alguns. Cabe às organizações decidir exatamente o que deve ser protegido, qual o nível de proteção que cada um dos recursos exige e quais ferramentas serão utilizadas para atingir esses objetivos (Almeida, 2007).

Surge então um primeiro desafio - encontrar, entre os gestores da organização, unidade no que se refere à definição dos meios para a implantação de programas de segurança. Em muitas organizações, prefere-se lidar diretamente com os problemas técnicos da ausência de políticas de segurança do que com a implementação de ações que evitariam esses problemas (Dhillon and Backhouse, 2000). Acrescenta-se o fato de que o conhecimento necessário para um bem sucedido projeto de SegInfo encontra-se disponível apenas em padrões técnicos (e.g ISO 27001) ou restrito no conhecimento de especialistas em segurança. Como consequência, projetos dessa natureza tendem a ser complexos, caros e de longa duração, com benefícios nebulosos a longo prazo, difíceis de serem medidos, sendo esse o segundo desafio a ser superado (da Silva et al., 2011).

Observa-se na Figura 1.2 que o número de incidentes de segurança aumenta a cada ano. Entretanto, pesquisas mostram que em torno de 75% dos ataques a sistemas são direcionados à camada de aplicação, o que possibilita às organizações um controle maior dos riscos, desde que utilizem ou projetem aplicações seguras (Razzaq et al., 2009). Apesar de também atacada, a camada física de infraestrutura é menos visada principalmente porque invasores consideram o anonimato mais difícil de ser mantido em ataques feitos à essa camada. Também são menos comuns ataques a Sistemas Operacionais (SO), já que a quantidade de vulnerabilidades detectadas nestes são em menor número do que as detectadas em aplicações *web*.

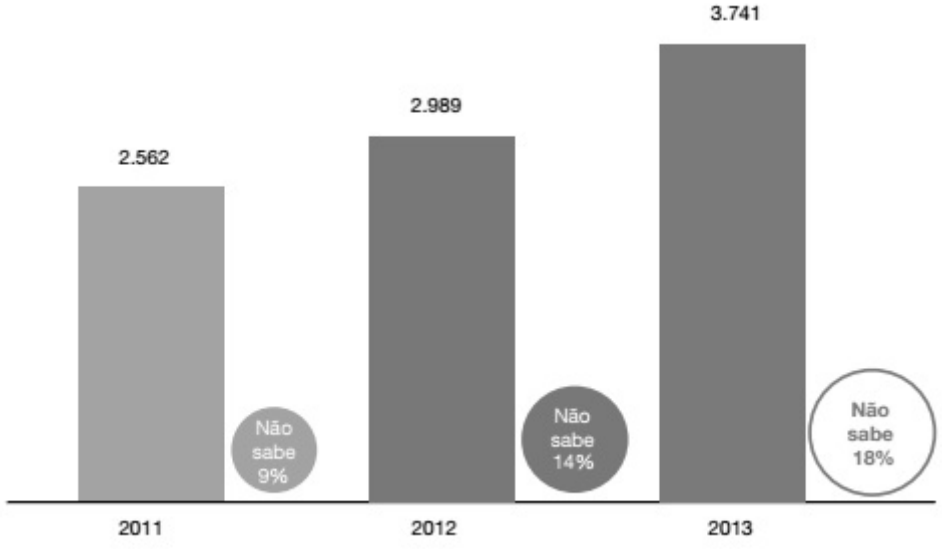

Figura 1.2: Número médio de incidentes de segurança - Fonte: PWC (2014)

A proteção de aplicações desenvolvidas para a *web*, cujas características po-

dem ser encontradas em Yadav et al. (2014), pode ser realizada de forma reativa ou pró-ativa. Na primeira, podem-se citar, por exemplo, mecanismos de análises de requisição de cabeçalhos, onde padrões maliciosos são descartados, com a aplicação já em produção sofrendo alguma tentativa de ataque. No segundo caso, a proteção se baseia em mecanismos de revisão de código e testes de penetração, feitos principalmente por especialistas em segurança, antes do produto final ser disponibilizado *online*. Idealmente, mecanismos de ambas as abordagens devem ser implementados para garantir a melhor segurança possível.

A grande maioria dos incidentes de segurança em aplicações *web* tem, portanto, sua origem na codificação feita de forma ingênua por desenvolvedores que não são especialistas em segurança; principalmente por estes considerarem apenas os cenários positivos de execução de código. Como alternativa para melhorar essa deficiência, nos últimos anos observa-se cada vez mais a inclusão de etapas relacionadas à segurança nos tradicionais ciclos de desenvolvimento de software. Essa é, por exemplo, a abordagem da *Microsoft* em seu processo de ciclo de vida do desenvolvimento da segurança (SDL - *Security Development Lifecycle*), o qual apresenta melhores resultados em relação à segurança quando adotado (Lipner, 2004).

Por outro lado, em uma pesquisa realizada com 286 desenvolvedores de sistemas para *web*, constatou-se que a grande maioria não possui interesse em participar de treinamentos ou processos que aumentassem sua conscientização sobre segurança no desenvolvimento de aplicações, sendo esse outro desafio para implantação de ciclos de desenvolvimento seguros. Essa pesquisa foi feita dentro de uma organização que desenvolve sistemas para *web* e encontra-se contextualizada na Seção 3.6.

Nesse trabalho, com base nos três desafios citados anteriormente, apresenta-se uma proposta para responder a seguinte questão de pesquisa: *Como desenvolver aplicações web mais seguras através do aumento do conhecimento dos desenvolvedores sobre segurança, de forma mais direcionada às questões de desenvolvimento?*

1.2 Justificativa

A negligência de boas práticas de programação em relação à utilização durante a fase de desenvolvimento de código seguro é uma das principais causas para a existência de vulnerabilidades nas aplicações. Esse é um dos maiores desafios para as organizações de todos os tipos, principalmente pela crescente popularidade de aplicações *web* (Vrancianu and Popa, 2010).

Os riscos relacionados à SegInfo tem se intensificado e evoluído, sem que as estratégias de segurança acompanhem essa evolução. As tecnologias mais conhecidas de proteção baseadas em sistemas de detecção de intrusão, dispositivos de encriptação e *firewalls* só são eficientes para vulnerabilidades bem conhecidas, pois utilizam um banco de dados com padrões de ataques para comparar com a requisição efetuada. Em todas essas propostas, além da proteção ineficaz para novos ataques ainda não catalogados, tem-se a geração de muitos alertas falso positivos e falso negativos (Razzaq et al., 2011).

O preocupante cenário descrito na Seção 1.1, onde as soluções para a mitigação de riscos em aplicações *web* são ineficientes ou demasiadamente complexas, demonstra uma clara e urgente necessidade de novas abordagens para ajudar na solução do problema.

Em resposta à essa demanda, soluções baseadas em ontologia tem surgido nos últimos anos. Abordagens ontológicas tem por objetivo a representação do conhecimento dos conceitos de determinado domínio de forma organizada. O formato do conhecimento atual disponível sobre SegInfo justifica esse tipo de abordagem, principalmente por ser vasto (a compreensão do todo demanda definição de extensa quantidade de conceitos) e dinâmico (novas informações e descobertas na área são frequentes e acontecem rapidamente, sendo de difícil acompanhamento por não-especialistas).

O desenvolvimento consistente de aplicações *web* seguras passa necessariamente por uma curva de aprendizado dinâmica de disseminação de conhecimento, educação e princípios (Andrews, 2006). Ontologias de segurança passam a figurar como o caminho para fornecer alternativas na contribuição para a solução do problema, ao apresentarem a capacidade de oferecer a democratização do conhecimento sobre riscos, vulnerabilidades, ameaças e todos os demais conceitos necessários para melhor compreensão do assunto.

É importante ressaltar o crescente uso de ontologias também na área de Gerenciamento de Processos de Negócios (BPM - *Business Process Management*), com o objetivo de alinhar BPM e TI com os melhores interesses da organização (de Azevedo et al., 2007). Baseado no fato que a implementação de atividades de SegInfo é considerada um processo de negócio em qualquer organização que desempenhe atividades de TI, justifica-se por mais esse motivo a tendência de utilização de ontologias para esse objetivo.

Segundo Raskin et al. (2001), a utilização da ontologia em SegInfo pode ser sumarizada a dois tipos de abordagem:

1. raciocínio (*reasoning*) - baseada em processamento de linguagem natural através de mecanismos reativos de análises de grandes volumes de data, como *logs* de segurança e alertas de vulnerabilidade. Nessa abordagem, a ontologia é constantemente atualizada com novas informações e padrões de acertos, sendo principalmente empregada para análise de ataques feitos à aplicação considerando determinado conjunto de regras.

2. representação do conhecimento (*knowledge representation*) - Baseada na definição e relacionamento de conceitos que ajudem os participantes do processo de negócio a tomar decisões relacionadas à segurança, de acordo com os requisitos da organização. A ontologia pode ser definida em diferentes níveis de abstração e utilizada para diferentes objetivos relacionados às atividades de segurança.

Escolheu-se nesta pesquisa o segundo tipo de abordagem, com foco na organização e distribuição do conhecimento de segurança para os desenvolvedores de aplicações *web*. O motivo é que essa abordagem tem sido menos explorada na literatura, sendo escolhida também pois acredita-se que a segurança deva fazer parte de todas as etapas do processo de desenvolvimento de *software*. Essa premissa

também é defendida na iniciativa OWASP de disseminação de conhecimento sobre SegInfo, que contribui para o estudo com duas bases de conhecimento sobre segurança na construção da ontologia - OWASP Top 10 e OWASP ASVS.

1.3 Objetivos

É objetivo geral desse trabalho desenvolver uma proposta ontológica que possa ser utilizada por desenvolvedores de aplicações *web* como auxílio para criação de sistemas mais seguros.

São objetivos específicos do trabalho proposto:

- fazer um levantamento das principais dificuldades e desafios enfrentados por gestores e profissionais relacionados ao processo de desenvolvimento de *software* em relação a mitigação de riscos em aplicações *web*;
- aprofundar o conhecimento em relação a três repositórios sobre vulnerabilidade de forma que possam ser integrados, desenvolvendo uma ontologia de domínio reunindo as informações disponibilizadas pelo repositório integrado;
- propor um mecanismo que utilize a ontologia para que, baseado em informações conhecidas do desenvolvedor, possa sugerir boas práticas de segurança para mitigações de riscos na aplicação;
- validar a proposta por meio de estudos de casos.

Faz-se necessário delimitar o assunto tratado. A solução para mitigação de riscos proposta é para uso em aplicações *web*, não abrangendo todos os tipos de aplicações. A ontologia desenvolvida baseia-se em repositórios dinâmicos que são atualizados com diferentes frequências, necessitando, portanto, de atualizações esporádicas para resultados mais eficientes.

Embora a solução tenha buscado a implementação de uma abordagem inovadora de pesquisa, não se pretende afirmar que essa é a única metodologia passível de ser aplicada.

O trabalho ora descrito é o início de uma pesquisa que une as áreas de segurança, desenvolvimento e da Ciência da Informação (ontologias), por vezes trabalhadas de maneira isolada. Essa proposta multidisciplinar trabalha a integração dessas ciências de maneira transparente, sempre abordando a visão geral do problema, mas tendo como foco a prototipação da ferramenta computacional, não sendo escopo desse trabalho o aprofundamento em nenhuma das áreas.

1.4 Organização do Documento

A realização desse trabalho desenvolveu-se em fases distintas. Inicialmente, buscou-se resgatar os conceitos de três áreas de pesquisa que por vezes trabalham de forma desassociadas: SegInfo, desenvolvimento de software e ontologia para a fundamentação do problema em pauta. Em seguida, pesquisas inerentes ao

problema foram feitas para o embasamento da proposta. Após a consolidação da fundamentação teórica, foi definida a proposta arquitetural e realizada a construção da ontologia. Após essa ser implementada, realizaram-se experimentações por meio de estudos de caso. Por fim, os resultados foram discutidos e o documento estruturado e formatado com o conteúdo estudado.

Assim, o Capítulo 2 desta pesquisa busca esclarecer os principais conceitos sobre SegInfo da área de Ciência da Computação, que serão necessários para compreensão do estudo, destacando as essenciais políticas de segurança e o processo de gerenciamento de riscos das organizações.

Por sua vez, o Capítulo 3 apresenta os conceitos sobre desenvolvimento de software seguro, com conceitos da área de Engenharia de Software e dois modelos para desenvolvimento seguro explicados de forma detalhada. Inclui também as três bases de conhecimento que serão utilizadas para a construção da ontologia sobre SegInfo, bem como a motivação e o modelo de análise que gerou o estudo.

No Capítulo 4, disserta-se sobre o tema de ontologias da área de Ciência da Informação, com os conceitos essenciais e a descrição de um método para construção de ontologias.

O Capítulo 5 apresenta a proposta de solução, com a descrição do modelo utilizado e da ferramenta proposta. A arquitetura e o protótipo são detalhados, explicitando a estrutura conceitual que fundamenta a construção da ontologia. Detalhes da implementação podem ser conferidos, bem como toda a infraestrutura computacional e tecnologias utilizadas. Nesse mesmo capítulo, fazem-se comparações com propostas estudadas do uso de ontologias para tornar aplicações mais seguras. Uma análise das proposições e resultados mais usuais e de lacunas existentes fundamentou a proposição do trabalho desenvolvido.

O Capítulo 6 aduz aos estudos de caso realizados, explorando a proposta de solução formalizada no Capítulo 5. Os experimentos desenvolvidos são então apresentados e os resultados discutidos.

Por fim, com base nas reflexões apresentadas pela fundamentação teórica e seu confronto com a implementação da proposta e seus resultados, o Capítulo 7 conclui o estudo, ao destacar os principais resultados obtidos e as contribuições alcançadas com a pesquisa. A continuidade da solução é discutida e complementada pelos trabalhos futuros que são sugeridos por fim.

Capítulo 2

Segurança da Informação

2.1 Conceitos e Nomenclatura

O conceito geral de segurança, como a qualidade ou o estado de estar seguro, livre de perigo, expande-se para as organizações em múltiplas divisões, para a proteção das operações diárias. Situa-se, nessa categorização, a SegInfo que é o objeto de estudo deste trabalho:

- Segurança física: proteção de items fisicamente, como objetos ou áreas de acesso não-autorizado e uso indevido;
- Segurança de pessoal: proteção do indivíduo que acessa as dependências da organização e suas operações;
- Segurança das operações: proteção dos detalhes de uma operação em particular ou de uma série de atividades;
- Segurança das comunicações: proteção dos meios de comunicação, conteúdo e tecnologia;
- Segurança de redes: proteção dos componentes de rede e conexões;
- Segurança da informação: proteção da confidencialidade, integridade e disponibilidade dos ativos de informação, seja quando armazenados, em processamento ou sendo transmitidos.

Atualmente o conceito de SegInfo está padronizado pela norma ISO/IEC 7799:2005, influenciada pelo padrão inglês (*British Standard*) BS 7799. Esse padrão inglês foi criado somente em 1995 pelo CCSC (*Commercial Computer Security Center*), centro de segurança desenvolvido pelo Reino Unido em 1989 que tinha a criação desses padrões como uma de suas metas. A ABNT homologou esse padrão em 2001 denominada NBR ISO/IEC 17799. A série de normas ISO/IEC 27000 foram reservadas para tratar de padrões de SegInfo, incluindo a complementação ao trabalho original do padrão inglês. A ISO/IEC 27002:2005 continua sendo considerada formalmente como 7799:2005 para fins históricos (Higgins, 2009).

A SegInfo é garantida pelo uso de políticas, educação, treinamento, conscientização e tecnologia. Podem ser estabelecidas métricas (com o uso ou não de ferramentas) para a definição do nível de segurança existente e, com isto, serem estabelecidas as bases para análise da melhoria ou piora da situação de segurança existente. A segurança de uma determinada informação pode ser afetada por fatores comportamentais e de uso de quem se utiliza dela, pelo ambiente ou infra-estrutura que a cerca ou por pessoas mal intencionadas que tem o objetivo de furtar, destruir ou modificar tal informação.

Os conceitos básicos que ajudarão na compreensão da ontologia e que serão utilizados no restante do documento são listados a seguir (Whitman and Mattord, 2011):

- Acesso: habilidade de um objeto ou indivíduo usar, manipular, modificar ou afetar outro objeto ou indivíduo. Usuários autorizados tem acesso legal ao sistema, *hackers* tentam acessar o sistema de forma ilegal. Essa habilidade é regulada por controles de acesso.

- Ativo: recurso da organização que está sendo protegido. O ativo pode ser lógico, como um *site* ou um dado de um banco de dados; ou físico, como uma pessoa ou um servidor em uma sala.

- Ataque: ato que pode causar danos ou comprometer informações de um sistema. Ataques passivos tem por objetivo a liberação de dados confidenciais. Baseiam-se em escutas e monitoramento de transmissões com o intuito de obter informações que estão sendo transmitidas. São difíceis de detectar pois não envolvem alteração de dados, mas são mais fáceis de prevenir com técnicas de criptografia, por exemplo. Ataques ativos resultam na alteração ou destruição do dado. Envolve modificação, criação de objetos falsificados ou negação de serviço, sendo de difícil prevenção, devido à necessidade de proteção completa de todas as funcionalidades de comunicação e processamento dos dados.

- Controle: mecanismos de segurança, políticas ou procedimentos empregados para conter possíveis ataques, reduzir riscos, mitigar vulnerabilidades, e, de forma geral, aumentar a segurança da organização.

- Exposição: condição ou estado de estar exposto, existente quando um invasor está ciente sobre uma vulnerabilidade.

- Perda: instância de um ativo de informação danificada, exposta ou modificada sem autorização.

- Programa de segurança: conjunto completo de controles, incluindo política, educação, treinamento e conscientização que é implementada pela organização para proteger os ativos.

- Risco: probabilidade de que algo indesejado irá ocorrer. As organizações precisam minimizar o risco até o ponto em que estão dispostas a aceitá-los.

- Ameaça: categoria de objetos, pessoas ou outras entidades que representam um perigo para o ativo.
- Agentes de ameaça: a instância relacionada à ameaça. Por exemplo, tempestades são uma ameaça; tendo raios e tornados como agentes dessa ameaça.
- Vulnerabilidade: fraqueza ou falha em um sistema ou mecanismo de proteção que o deixa propício à ataques. Vulnerabilidades podem estar bem documentadas, quando são conhecidas e estudadas, ou latentes, quando existem mas ainda não foram descobertas.
- Impacto: consequência da ocorrência de um incidente de segurança.

Dentre os principais conceitos da área, tem-se os pilares de confidencialidade, integridade e disponibilidade, conhecidos como tríade CIA (*Confidentiality, Integrity, Availability*), como base para definir as características da informação que precisa ser protegida. Entretanto, com as mudanças rápidas e constantes nessa área de estudo, a tríade passou a não ser mais suficiente para definir todas as características inerentes à informação. Outros conceitos passaram a fazer parte dos pilares básicos para o conhecimento sobre SegInfo (Parker, 2002), sendo a adição de autenticidade e não repúdio os mais comumente encontrados atualmente. Esses cinco conceitos estão explicados a seguir:

- Confidencialidade: compreende a proteção de dados transmitidos contra ataques passivos, isto é, contra acessos não autorizados, envolvendo medidas como controle de acesso e criptografia. A perda de confidencialidade ocorre quando há uma quebra de sigilo de uma determinada informação (exemplo: a senha de um usuário ou administrador de sistema), permitindo que sejam expostas informações restritas que seriam acessíveis apenas por um determinado grupo de usuários.
- Integridade: trata da garantia da corretude da informação, protegida contra ataques ativos que tem por objetivo alterações ou remoções não autorizadas. É relevante o uso de um esquema que permita a verificação dos dados armazenados e em transmissão. A perda de integridade surge no momento em que uma determinada informação fica exposta ao manuseio por uma pessoa não autorizada, que efetua alterações que não foram aprovadas e não estão sob o controle do proprietário da informação.
- Disponibilidade: determina que os recursos estejam disponíveis para acesso por entidades autorizadas, sempre que solicitados, representando a proteção contra perdas ou degradações. A perda da disponibilidade acontece quando a informação deixa de estar acessível por quem necessita dela. Seria o caso da perda de comunicação com um sistema importante para a organização que aconteceu com a queda de um servidor ou de uma aplicação crítica de negócio, que apresentou uma falha devido a um erro.
- Autenticidade: visa garantir que um mecanismo é autêntico, como por exemplo, uma comunicação via troca de mensagens. Origem e destino podem verificar a identidade da outra parte, com o objetivo de confirmar que

a outra parte é quem alega ser. A origem e o destino podem ser usuários, dispositivos ou processos.

- Não repúdio: compreende o serviço que previne uma origem ou destino de negar a transmissão de mensagens, isto é, quando dada mensagem é enviada, o destino pode provar que esta realmente foi enviada por determina origem e vice-versa.

Além desses, encontra-se na literatura recente diversos outros conceitos (ex: acuracidade, utilidade, posse, etc) sendo empregados como características básicas da informação, expandido a tríade original para às vezes até dez conceitos fundamentais. E além dos conceitos básicos, divide-se os componentes que tornam possível a utilização da informação de forma adequada e segura nas organizações em seis grandes grupos (Whitman and Mattord, 2011):

- *Software*: engloba as aplicações, sistemas operacionais e todas as suas funcionalidades. É considerado o componente mais difícil de ser protegido, com a exploração de erros por falhas na programação contribuindo substancialmente para a quantidade de ataques. A indústria da tecnologia da informação é diariamente inundada com novos alertas sobre falhas e vulnerabilidades que são descobertas, afetando desde sistemas em *smartphones* a problemas em *softwares* automotivos que obrigam as organizações a lançar correções todo o tempo. Isso ocorre pois os sistemas são criados dentro dos limites de tempo, custo e mão-de-obra dos projetos, geralmente insuficientes para o adequado cuidado com a segurança. Acrescenta-se a isso o fato de que a segurança não é considerada durante todo o ciclo de projeto, como será detalhado na seção seguinte.

- *Hardware*: é a parte física da tecnologia que abriga e executa os sistemas, armazena e transporta os dados e provê a interface para a entrada e retirada de informações do sistema. As políticas de segurança física protegem os ativos de roubo e dano, aplicando as técnicas tradicionais desde fechaduras e chaves até acesso restrito aos componentes do sistema.

- Dados: dados que são armazenados, processados e transmitidos por um sistema computacional devem ser protegidos. É o ativo mais importante das organizações e o principal alvo dos ataques intencionais. Na grande maioria das configurações, estão armazenados em bancos de dados que possuem funcionalidades avançadas de segurança, mas que nem sempre são utilizadas.

- Pessoas: por vezes deixado de lado em modelos de SegInfo, é uma das maiores ameaças a sistemas computacionais, considerado muitaz vezes como o elo mais fraco em um programa de segurança, aquele que pode ser mais facilmente comprometido. A importância desse componente tem aumentado consideravelmente nos últimos anos, com diversos estudos oferecendo soluções para diminuir o risco relacionado aos chamados ataques de engenharia social (Sasse et al., 2001), (Laszka et al., 2013). A engenharia social pode e tem sido utilizada com sucesso para extrair informações direta ou

indiretamente sobre sistemas, seja pela manipulação de pessoas ou em vulnerabilidades criadas quando executam ações indevidas.

- Procedimentos: também muitas vezes não considerados em programas de segurança, procedimentos devem ser distribuídos nas organizações apenas de acordo com a necessidade de conhecimento. Quando um usuário não autorizado tem acesso à execução de procedimentos, a integridade da informação pode vir a ser seriamente comprometida, por exemplo. A proteção de informações sobre procedimentos é geralmente conseguida com programas de educação com os funcionários da organização.

- Redes: quando sistemas de informação estão conectados em redes como a Internet, desafios de segurança surgem rapidamente, já que além da segurança física são necessárias medidas para proteger os sistemas interconectados, que recebem e enviam informações a todo momento e em quantidades cada vez maiores. Equipamentos de proteção de dispositivos de interconexão tem evoluído para atender a essa demanda, tornando-se um componente básico imprescindível em qualquer programa de segurança.

2.2 Programas de Segurança

Programas de segurança de informação devem buscar a disponibilização da informação de forma segura, buscando o meio termo com os mecanismos disponíveis para um acesso racional. O equilíbrio entre segurança e acesso é o grande desafio de SegInfo. De um lado, tem-se os usuários com a necessidade de utilizar os recursos da forma mais prática e eficiente possível. De outros, tem-se os especialistas em segurança buscando métodos para garantir a manutenção das características da informação. O dilema entre segurança e usabilidade pode ser exemplificado com uma organização adotando a criptografia como uma exigência para envio de e-mails; ao mesmo tempo que a medida visa proteger a informação, usuários serão críticos pois o processo se torna mais longo e lento (Lacey, 2011).

Em 1991, foi proposto um modelo que estrutura os conceitos de SegInfo em forma de cubo, o cubo de McCumber (McCumber, 2004), largamente adotado para definição de metodologias relacionadas à segurança desde então. No modelo, propõe-se três eixos conforme Figura 2.1, que, quando extrapolados, geram um cubo 3 x 3 x 3 com 27 células no total. Cada célula representa uma dimensão que deve ser abordada durante o processo de garantia de segurança.

O primeiro eixo lista as características da informação (tríade CIA), o segundo eixo os possíveis estados da informação (guardada, processada, transmitida) e o terceiro eixo as categorias das medidas que devem ser adotadas (política de segurança, educação e tecnologia) em implementações de programas de segurança. Por exemplo, a interseção de 'tecnologia', 'integridade' e 'guardado' pode ser traduzida como a 'necessidade do uso de um mecanismo tecnológico para proteger a integridade dos dados que estão armazenados em um banco de dados'. Como controle para essa dimensão, pode-se citar um sistema de detecção de modificações em arquivos que avisa o administrador de mudanças suspeitas.

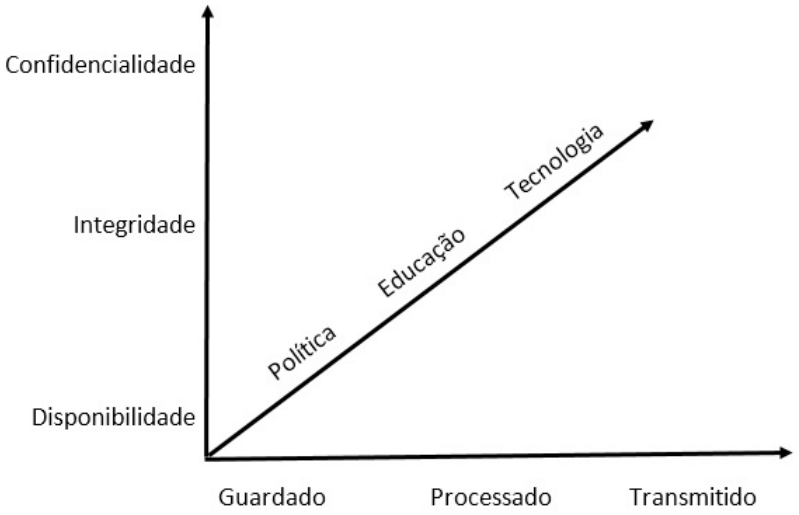

Figura 2.1: Eixos para o modelo do cubo de McCumber - Fonte: adaptado de (McCumber, 2004)

O cubo de McCumber foi a primeira efetiva tentativa de estruturar o conhecimento sobre SegInfo, de modo a facilitar a adoção pelas organizações, que podem utilizá-lo parcialmente ou totalmente dependendo dos requisitos que sejam definidos. Uma das principais características do modelo é a ausência de hierarquia entre os itens que o compõem, isto é, todos são igualmente importantes e necessários na adoção de programas de segurança.

A implantação de programas de SegInfo nas organizações geralmente começa com a elaboração de um política de segurança, que consiste num conjunto formal de regras que devem ser seguidas pelos utilizadores dos recursos da organização. As políticas de segurança devem ter implementação realista, e definir claramente as áreas de responsabilidade dos utilizadores, do pessoal de gestão de sistemas e redes e da direção. Deve também adaptar-se a alterações na organização, fornecendo um enquadramento para a implementação de mecanismos de segurança, definindo procedimentos de segurança adequados, processos de auditoria à segurança e estabelecendo uma base para procedimentos legais na ocorrência de ataques. O documento que define a política de segurança deve deixar de fora todos os aspectos técnicos de implementação dos mecanismos de segurança, pois essa implementação pode variar ao longo do tempo. Deve ser também um documento de fácil leitura e compreensão.

Políticas de segurança devem também especificar claramente o papel de todos os envolvidos no processo de proteção da informação, incluindo ações que devem ser tomadas quando da ocorrência de um incidente. O início de um programa de sucesso é a implementação de uma política de segurança bem escrita, seguido de treinamento e verificações periódicas que avaliem se os itens previstos estão sendo seguidos (Peltier, 2013).

Lorens (2009) define que por ter um caráter muito abrangente as políticas de segurança devem ser sub-divididas em três grupos: diretrizes, normas e procedimentos, destinados respectivamente à ações estratégicas, táticas e operacionais.

As diretrizes, que por si só já possuem um papel estratégico, devem expressar a importância que a empresa dá para a informação, comunicar aos envolvidos seus valores e seu comprometimento em aumentar culturalmente a importância da segurança. Com caráter tático, as normas são o segundo nível da política, necessitando detalhar situações, ambientes, processos e fornecendo orientação para o uso adequado da informação. Por fim, procedimentos devem estar presentes em maior quantidade por seu perfil operacional, descrevendo de forma detalhada cada ação e atividade associada a cada situação distinta do uso da informação. Exemplificando, uma diretriz estabelece que informações confidenciais precisam ser protegidas, enquanto a norma define que essas informações devem ser criptografadas quando enviadas por e-mail e o procedimento especifica qual ferramenta deve ser utilizada para o processo de criptografia, e de que forma.

Fica claro, portanto, o alto grau de complexidade para a implantação e manutenção de todos os componentes de uma política de segurança, sendo valiosas as contribuições que possam facilitar esse processo, especialmente considerando o dinamismo do assunto e as mudanças previsíveis e imprevisíveis que uma organização pode sofrer (Sêmola et al., 2003).

2.3 Gerenciamento de Riscos

Usando a definição do NIST[1] (*National Institute of Standards and Technology*), risco é a probabilidade de que uma fonte de ameaça e uma potencial vulnerabilidade resultem em um evento que cause um impacto adverso à organização.

Gerenciamento de riscos em SegInfo é o processo de identificação de riscos dos ativos de informação e infraestrutura de uma organização, representado por vulnerabilidades, e das ações que devem ser tomadas para que estes sejam reduzidos a níveis aceitáveis. O processo envolve três fases bem definidas: identificação, avaliação e controle, sendo esse conjunto de atividades, parte vital de um programas de segurança e imprescindível em políticas para a área, justamente o foco de atuação do presente estudo.

A análise de riscos é um processo constante, motivado por mudanças tecnológicas que afetam a organização e todos os ativos que com ela se relacionam. Por exemplo, uma nova análise de riscos é necessária quando se troca um fornecedor de um produto ou quando um processo interno é modificado. Em SegInfo, como ilustrado na Figura 2.2, a análise de riscos deve ser realizada de acordo com a legislação vigente, o modelo adotado pela organização, e dentro das etapas previstas na política de segurança, como descrito anteriormente.

Na etapa de identificação, procura-se definir quais são os recursos considerados críticos para a atividade e sobrevivência da organização, recursos estes que são valorizados de acordo com a sua relevância para o negócio, facilidade de substituição e investimento necessário para reparação ou substituição. Identifica-se também as vulnerabilidades que se encontram associadas a estes recursos e as ameaças a que estes se encontram expostos, bem como a sua probabilidade de ocorrência e impacto esperado (quantificado, sempre que possível).

[1] www.nist.gov

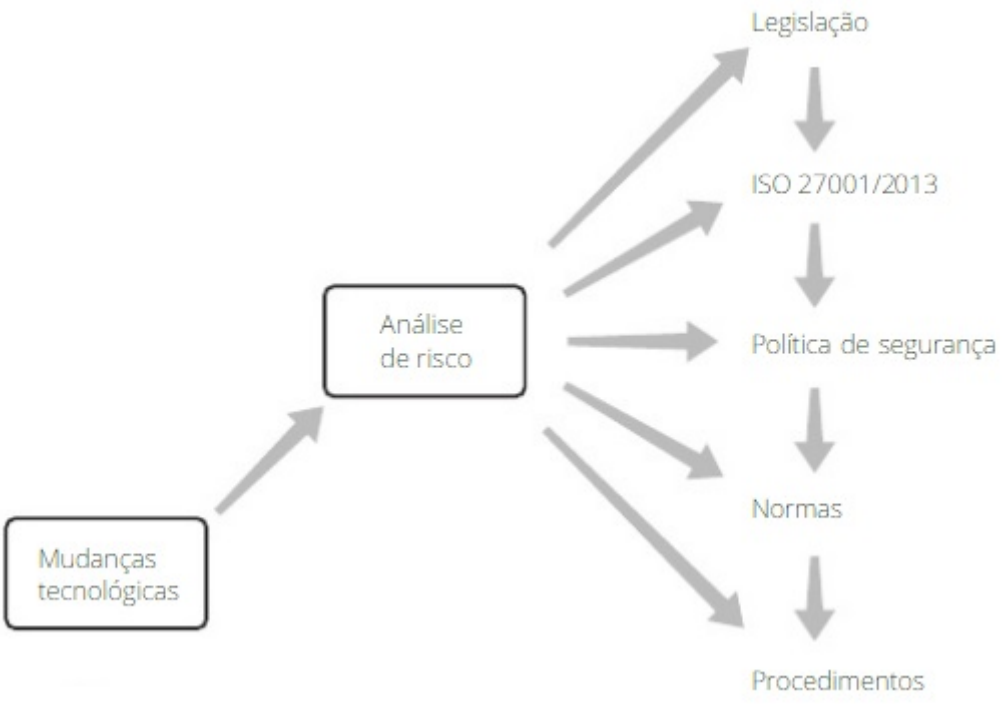

Figura 2.2: Processo de análise de riscos - Fonte: (Coelho et al., 2011)

Em seguida, avalia-se, da forma mais realista possível, as perdas e danos (tangíveis e intangíveis) associados aos impactos resultantes da concretização de uma ou mais ameaças, sobre um dado recurso. Deste cálculo decorre o nível de risco associado ao recurso em questão. De acordo com o nível de risco identificado, segue-se a classificação deste quanto à sua aceitação ou necessidade de mitigação, atendendo ao grau de conforto pretendido pela organização. Um nível de risco pode ser aceito caso a organização decida que este não acarreta consequências significativas para a concretização das suas atividades críticas de negócio, sendo que a aceitação de um determinado nível de risco pode, contudo, presumir a realização de esforços no sentido de mitigá-lo, reduzindo-o a um valor considerado aceitável para a organização, dotando-a de um nível de conforto desejável.

A análise e avaliação devem ser realizadas periodicamente, a fim de contemplar as mudanças nos requisitos de segurança. Diagnosticar o risco envolve estudo de diversas variáveis que passam do aspecto tecnológico. Devem-se considerar também aspectos comportamentais dos recursos humanos, aspectos físicos, legais, mudanças de estratégia organizacional (provocada, por exemplo, pelo surgimento de um novo concorrente), dentre outros. A análise de riscos deve ser feita com conhecimento sobre os desafios do negócio, o mapeamento das funcionalidades dos processos e o relacionamento destes com a diversidade de ativos (físicos, tecnológicos e humanos) que hospedam falhas de segurança Coelho et al. (2011).

A metodologia utilizada pode ser quantitativa, para mensurar os impactos financeiros provocados por um incidente de segurança baseado na valoração dos ativos, ou qualitativa, quando orientada por critérios que visam estimar os impactos provocados ao negócio pela exploração de uma vulnerabilidade. Ambas

possuem suas dificuldades e limitações e geralmente utiliza-se uma combinação das duas abordagens, com destaque para a análise qualitativa pela dificuldade do processo de valoração dos ativos.

Após o diagnóstico dos riscos, deve-se definir junto à alta administração da organização, quais os níveis de risco aceitáveis e não-aceitáveis. Entre os não aceitáveis, pode-se escolher uma entre as seguintes opções (Martins and Santos, 2005):

- Reduzir o nível de risco - aplicando controles de segurança.
- Aceitar o risco - considerar que ele existe, mas não aplicar qualquer controle.
- Transferir o risco - repassar a responsabilidade de segurança a um terceiro.
- Negar o risco - opção menos recomendada.

Riscos relacionados à aplicações *web* estão quase que sempre categorizados dentro da primeira opção, isto é, precisam ser controlados até níveis aceitáveis. Algumas vezes podem também ser transferidos para um terceiro, como por exemplo um fornecedor de serviço de hospedagem de páginas, que deve garantir que seus sistemas estejam o mais protegido quanto possível.

O processo de gerenciamento de riscos encontra-se inserido nas atividades de diagnóstico da abordagem de segurança em camadas. Para a implementação desta estratégia de segurança, deve-se segmentar o acesso à informação em perímetro físico e lógico, visando oferecer níveis diferenciados de resistência e proteção. Perímetros físicos são, por exemplo, área de recepção, circulação de pessoal, ambiente de produção e servidores. Perímetros lógicos são o sistema de autenticação de visitantes, a Intranet, a própria Internet.

Segundo Peltier (2013), a defesa em camadas prevê a implementação de fases com níveis de resistência crescentes :

- Desencorajar - Visa exclusivamente fazer com que as ameaças percam seu estímulo inicial para exploração de vulnerabilidades, o que pode ser feito utilizando mecanismos físicos, tecnológicos ou humanos, implementando-se câmeras de vídeo. Campanhas de divulgação da política de segurança, informação sobre os procedimentos de auditoria, etc.
- Dificultar - É a segunda barreira. É complementar a anterior e visa a adoção efetiva de controles para dificultar o acesso indevido. Exemplos são roletas, detectores de presença com alarmes, senhas, certificados digitais, etc.
- Discriminar - É a ação de identificar uma situação de risco, ou seja, a tentativa de alguém de burlar as ações da fase anterior. Exemplos dessas ações são os antivírus, sistemas de detecção de intrusão, entre outros.
- Deter - Objetiva impedir que a ameaça atinja os ativos, caso as barreiras anteriores não tenham sido eficazes. Ações administrativas, punitivas, bloqueio de acesso físico e lógicos são exemplos.

- Diagnosticar - Não é simplesmente o último passo dessa segmentação, por representar o elo de ligação com a primeira barreira, sendo portanto a mais importante para o aprimoramento das estratégias de defesa dos ativos. Um bom diagnóstico é representado por uma detalhada análise de riscos, que deverá corrigir rumos e atualizar estratégias de acordo com todo o dinamismo que certa o ambiente de negócios.

O termo defesa em camadas (*defense in depth*) é também utilizado em SegInfo para designar a abordagem que prevê diversos mecanismos de segurança sobrepostos para garantir proteção em forma de redundância, caso um dos controles de segurança falhe. Ele tem origem nas estratégias militares de atrasar ao máximo um ataque para tentar evitá-lo por diversas abordagens, também ganhando tempo enquanto isso. Soluções únicas de segurança, por mais fortes e bem implementadas que sejam, acabam por se tornar um ponto único de falha, comprometendo toda a informação caso ela seja vencida (Byres, 2008).

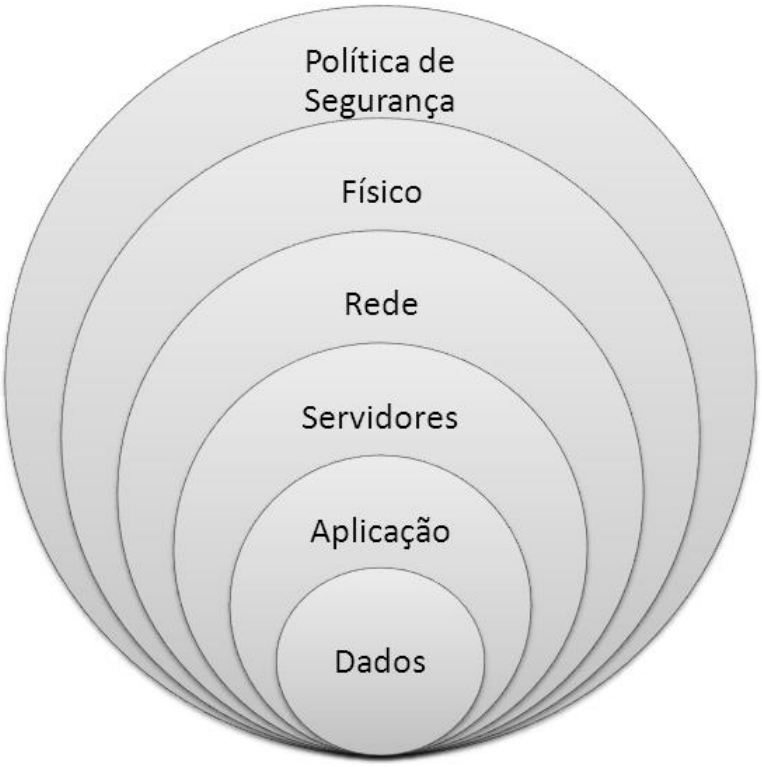

Figura 2.3: Defesa em camadas - Fonte: adaptado de (Peltier, 2013)

Um exemplo holístico desse tipo de abordagem é ilustrado na Figura 2.3. A informação é protegida pela camada de aplicação, hospedada em servidores e computadores que se conectam pela rede, encapsuladas pela proteção física. A última camada de política de segurança engloba também conscientização e treinamento sobre segurança, além de todas as camadas inferiores. Cada uma das camadas tem o seu conjunto de práticas e mecanismos, sendo o foco do presente trabalho atuar tanto na proteção via redução de vulnerabilidades na camada de aplicação, como na última camada por meio do aumento do conhecimento sobre segurança de aplicações *web*.

O desafio para a implantação da defesa em camadas é a determinação do equilíbrio entre custo, funcionalidades, desempenho e necessidades operacionais (NSA, 2007). As características das aplicações *web* fazem com que a abordagem em camadas tenha que ser implementada de forma cuidadosa, pois a arquitetura possibilita ataques diretamente à camada de aplicação, que, se comprometida, pode resultar em acesso direto à informação que precisa ser protegida. Um estudo sobre a importância da defesa em camadas para aplicações pode ser encontrado com mais detalhes em Stytz (2004).

Modelos de ameaça tem sido utilizados com sucesso em processos de análise de riscos, como uma forma de garantir que os aplicativos tenham as propriedades exigidas pela política de segurança (Hernan et al., 2006). O modelo da Microsoft STRIDE, acrônimo de *Spoofing* (falsificação), *Tampering* (violação), *Repudiation* (repúdio), *Information Disclosure* (divulgação não autorizada de informação), *Denial of Service* (negação de serviço) e *Elevation of Privilege* (elevação de privilégio) mapeia as ameaças às propriedades de segurança conforme a Tabela 2.1.

Tabela 2.1: Modelo STRIDE de ameaças - Fonte: (Hernan et al., 2006)

Ameaça	Propriedade de Segurança
(S) Falsificação	Autenticação
(T) Violação	Integridade
(R) Repúdio	Não-repúdio
(I) Divulgação não autorizada de informação	Confidencialidade
(D) Negação de serviço	Disponibilidade
(E) Elevação de privilégio	Autorização

Para aplicações *web*, o modelo STRIDE pode ser utilizado em diagramas de fluxo de dados com o intuito de determinar quais as possíveis vulnerabilidades para cada tipo de ataque (Klöti, 2013). Com o mapeamento das vulnerabilidades, realiza-se a análise de riscos como citado anteriormente.

Capítulo 3

Desenvolvimento de Aplicações Seguras

Segundo a definição em Pressman (2011), *software* é o produto que profissionais de tecnologia da informação constroem e depois mantém ao longo do tempo. O conceito abrange programas que executam em computadores de qualquer tamanho e qualquer arquitetura. O conteúdo que é apresentado ao programa a ser executado e documentos, tanto em forma impressa quanto virtual, que combinam todas as formas de mídia eletrônica. Aplicações *web* são, portanto, programas desenvolvidos por esses profissionais, dentro de um ciclo de desenvolvimento conhecido como ciclo de desenvolvimento de *software*.

As aplicações *web* variam imensamente em relação a complexidade. Podem ser compostas por algumas dezenas de linhas de código desenvolvidas por um único desenvolvedor ou por milhares de linhas de códigos e componentes desenvolvidos modularmente por centenas de pessoas. Em qualquer um dos casos, o processo de desenvolvimento deve seguir uma metodologia formal definida pela organização, de modo a garantir as funcionalidades e o atendimento das expectativas iniciais.

Esse processo de desenvolvimento contempla um conjunto de atividades e seus respectivos resultados associados, sendo as atividades fundamentais de acordo com Sommerville et al. (2011):

- Especificação: processo de definição das funcionalidades e restrições de operação do *software*.
- Desenvolvimento: processo deve fazer com que o *software* a ser produzido atenda às especificações.
- Validação: processo deve validar o *software* para garantir que ele faça o que foi solicitado.
- Evolução: processo deve permitir que, durante a evolução do *software* ao longo de várias versões, continue a atender à periódica mudança de necessidades anteriormente especificadas.

A execução dessas atividades formuladas com base em padrões previamente descritos é conhecida como modelo de processo de *software*, área de estudo da engenharia de *software*. Esse modelo de processo define como será o ciclo de

desenvolvimento empregado, dos quais podemos citar os modelos cascata e processo unificado, que são descritos detalhadamente juntamente com diversos outros modelos em Sommerville et al. (2011) e Pressman (2011).

Esses modelos foram concebidos em um momento histórico e tecnológico onde ainda não havia preocupação generalizada com os vários problemas de segurança decorrente da exposição das aplicações computacionais às redes abertas, não sendo em seus formatos originais recomendados para o desenvolvimento de aplicações *web*(de Holanda and Fernandes, 2009). Em resposta a esse novo cenário, o desenvolvimento de *software* tem evoluído nos últimos dez anos, visando incorporar de forma mais explícita o trato de questões de SegInfo durante as fases de desenvolvimento.

O foco inicial do desenvolvimento nesses modelos mais antigos é fundamentalmente o atendimento aos requisitos de funcionamento, para satisfazer as necessidades contratuais de clientes e requisitos dos usuários. Dessa forma, os critérios de segurança para tornar um *software* seguro muitas vezes só são realizados tardiamente, durante as fases de teste e validação. Realizar atividades de segurança em uma fase posterior de desenvolvimento produz um impacto negativo sobre o projeto, culminado com a criação de uma aplicação insegura com elevado número de vulnerabilidades.

De acordo com a ISO/IEC 17799:2000, todos os requisitos de segurança devem ser identificados na fase de levantamento de requisitos do ciclo de desenvolvimento, o que demandou novas abordagens para o processo que serão descritas a seguir. No presente estudo, atua-se de forma alinhada com a norma, de forma a patrocinar as práticas de segurança justamente na fase anterior ao processo de desenvolvimento da aplicação *web*. Importante destacar também o custo envolvido quando os conceito de segurança são introduzidos nas etapas recomendadas, citando o próprio texto da norma: "Controles introduzidos no estágio de projeto são significativamente mais baratos para implementar e manter do que aqueles incluídos durante ou após a implementação". Também segundo o NIST, estima-se que correções em código fonte após o lançamento podem resultar em 30 vezes o custo de correções realizadas durante a fase de projeto.

Uma metodologia de desenvolvimento voltada para a segurança deve incluir necessariamente os seguintes atributos (Wiesmann et al., 2005):

- Facilidade de adaptação para testes, design e documentação;
- Atividades onde os controles de segurança (tais como análise de risco, ameaças, revisões de código, etc.) podem ser aplicados;
- Métricas que auxiliam o aumento do nível de maturidade de segurança da organização e;
- Potencial para reduzir as taxas de erros correntes, e melhorar a produtividade do desenvolvimento.

Dentre os modelos para desenvolvimento seguro mais conhecidos, podemos citar o SDL (*Secure Development Lifecycle*) da Microsoft e o CLASP (*Comprehensive, Lightweight Application Security Project*) do OWASP, que serão descritos a seguir, ambos disponíveis de forma gratuita.

3.1 Ciclo de Desenvolvimento Seguro

O modelo proposto pela Microsoft para desenvolvimento de *software* tem como pilar um conjunto de princípios de alto nível para a compilação mais segura. A Microsoft se refere a esses princípios como SD3+C - Seguro por Design, Seguro por Padrão (*Default*), Seguro na Implantação (*Deployment*) e Comunicações. As definições resumidas desses princípios são:

- Seguro por Design: a arquitetura, o design e a implementação do *software* devem ser executados de forma a protegê-lo e proteger as informações que ele processa, além de resistir a ataques.
- Seguro por Padrão: na prática, o *software* não atingirá uma segurança perfeita; portanto, os arquitetos devem considerar a possibilidade de haver falhas de segurança. Para minimizar os danos que ocorrem quando invasores miram nessas falhas restantes, o estado padrão do *software* deve aumentar a segurança. Por exemplo, o *software* deve ser executado com o privilégio mínimo necessário, e os serviços e os recursos que não sejam amplamente necessários devem ser desabilitados por padrão ou ficar acessíveis apenas para uma pequena parte dos usuários (administradores).
- Seguro na Implantação: o *software* deve conter ferramentas e orientação que ajudem os usuários finais e/ou administradores a usá-lo com segurança. Além disso, a implantação das atualizações deve ser fácil.
- Comunicações: os desenvolvedores de *software* devem estar preparados para a descoberta de vulnerabilidades do produto e devem comunicar-se de maneira aberta e responsável com os usuários finais e/ou com os administradores para ajudá-los a tomar medidas de proteção (como instalar *patches* ou implantar soluções alternativas).

O ciclo de desenvolvimento seguro da Microsoft (SDL) é uma proposta para criação de processos com práticas de segurança consistentes, o qual é dividido em sete fases: treinamento (*training*), requisitos (*requirements*), desenho (*design*), construção (*implementations*), verificação (*verification*), liberação (*release*) e resposta (*response*). O SDL envolve a modificação dos processos de uma organização de desenvolvimento de *software* por meio da integração de medidas que levam a uma segurança aprimorada. A intenção dessas modificações não é revisar totalmente o processo, mas adicionar pontos de verificação e produtos de segurança bem definidos.

Cada um dos processos possui suas respectivas práticas de segurança, ilustradas na Figura 3.1, em total de dezessete na atual versão.

As fases do modelo são descritas a seguir (Lipner, 2004):

1. Treinamento: envolve a aplicação de treinamento básico e avançado aos membros da equipe de desenvolvimento de software, de forma que aborde conceitos sobre princípios, tendências e privacidade. Os conceitos básicos que impreterivelmente precisam ser abordados para o desenvolvedor, segundo o modelo SDL, são:

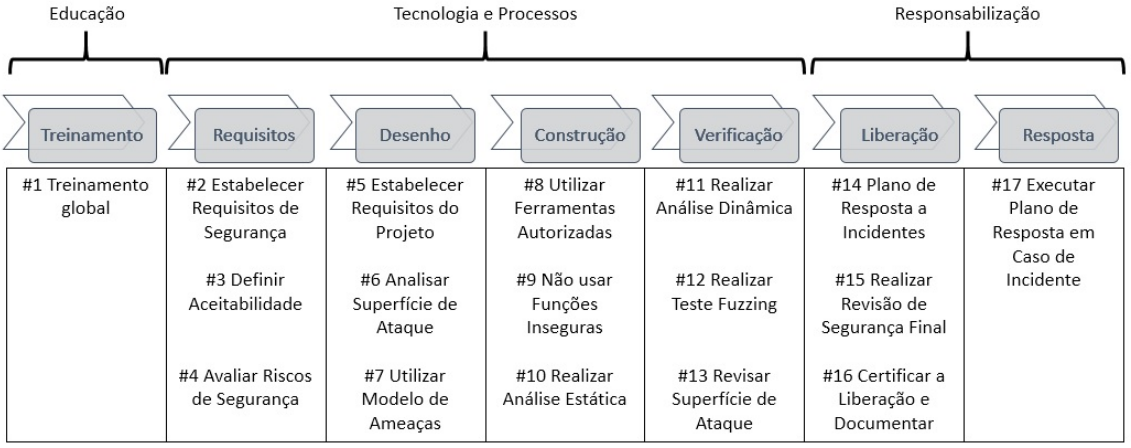

Figura 3.1: Modelo Microsoft SDL - Fonte: adaptado de (Microsoft, 2010)

- Desenho seguro - que estabelece conceitos sobre a redução da superfície de ataque, defesa em profundidade, princípio dos privilégios mínimos e princípios do SD3+C descritos acima.
- Modelagem de ameaças - que consiste na realização de estudos visando analisar como um possível atacante visualiza a aplicação, quais ativos despertam interesse, quais vulnerabilidades podem ser exploradas e como podem ser evitadas. Um exemplo seria o uso do modelo STRIDE descrito em 2.3.
- Codificação segura - envolve desenvolver capacidades de construção e revisão de códigos que não sejam suscetíveis a ataques, especialmente os mais conhecidos e documentados. Deve considerar particularidades específicas da plataforma utilizada no desenvolvimento da aplicação.
- Teste de segurança - envolve o equilíbrio entre testes funcionais com os testes de segurança, baseados nos riscos e ameaças definidos anteriormente para a aplicação em questão.
- Questões relacionadas a privacidade de informações pessoais, especialmente PII (*Personally Identifiable Information*), como nome, sobrenome e data de nascimento, existentes nas base de dados da aplicação.

2. Requisitos: a necessidade de considerar a segurança 'de baixo para cima' é um princípio fundamental do desenvolvimento de sistemas seguros. Embora vários projetos de desenvolvimento produzam 'próximas versões' baseadas nas versões anteriores, a fase de requisitos e o planejamento inicial de uma nova versão oferecem a melhor oportunidade para criar *software* seguro. A fase de requisitos é a oportunidade para a equipe de produto considerar como a segurança será integrada no processo de desenvolvimento, identificar os objetivos-chave de segurança e maximizar a segurança de *software*, minimizando a quebra de planos e cronogramas. Como parte desse processo, a equipe precisa considerar como os recursos de segurança e as medidas de controle de seu *software* serão integrados com outros softwares que provavelmente serão usados com ele. As atividades de determinação do nível de

risco aceitável e gerenciamento de risco são realizadas durante essa fase.

3. Desenho: a fase de desenho (*design*) identifica a estrutura e os requisitos gerais do *software*. Possui como elementos-chave:

   - Definir as diretivas de desenho e arquitetura de segurança: definir a estrutura geral do *software*, tendo como ponto de vista a segurança, e identificar os componentes cujo funcionamento correto é essencial para a segurança (a 'base de computação confiável'). Identificar técnicas de desenho, como a organização em camadas, o uso de linguagem com rigidez de tipos, a aplicação de privilégios mínimos e a minimização da superfície de ataque, que se aplicam ao *software* globalmente. As particularidades dos elementos individuais da arquitetura serão detalhadas nas especificações individuais, mas a arquitetura de segurança identifica uma perspectiva geral no desenho da segurança.
   - Documentar os elementos da superfície de ataque do *software*: como o *software* não atingirá uma segurança perfeita, é importante que apenas os recursos que serão usados pela grande maioria dos usuários sejam expostos a todos eles por padrão, e que esses recursos sejam instalados com o nível de privilégio mais baixo possível. A medição dos elementos da superfície de ataque fornece à equipe de produto uma métrica constante da segurança padrão e permite que a equipe detecte as instâncias em que o *software* se torna mais suscetível a ataques. Algumas instâncias com uma maior superfície de ataque podem ser justificadas pela usabilidade ou por uma função avançada requerida no produto, mas é importante detectar e questionar cada uma dessas instâncias durante o design e a implementação, de forma a fornecer *software* com uma configuração padrão tão segura quanto possível.
   - Realizar a modelagem de ameaças: a equipe de produto realiza a modelagem de ameaças componente por componente. Usando uma metodologia estruturada como a STRIDE, identifica-se os ativos que o *software* deve gerenciar e as interfaces pelas quais esses ativos podem ser acessados. O processo de modelagem de ameaças identifica as ameaças que podem danificar cada ativo e a probabilidade de acontecerem danos (uma estimativa de risco). O processo de modelagem de ameaças deve ter o suporte de uma ferramenta que capture modelos de ameaças em um formulário legível por máquina para armazenamento e atualização.
   - Definir critérios de fornecimento complementar: os critérios básicos de fornecimento de segurança devem ser definidos no nível da organização, mas as equipes de produto individuais ou de versões do *software* podem ter critérios específicos que devem ser atendidos antes do lançamento do software. Por exemplo, uma equipe de produto que desenvolva a versão atualizada de um *software* fornecido aos clientes e sujeito a ataques extensivos pode solicitar que, por determinado tempo, a nova versão fique livre de vulnerabilidades relatadas externamente antes de ser considerada pronta para o lançamento.

4. Implementação: também considerada como processo de codificação segura, as etapas seguidas para remover falhas de segurança ou evitar sua inserção inicial durante essa fase tem um aproveitamento alto; elas reduzem significativamente a probabilidade de que vulnerabilidades de segurança estejam presentes na versão final da aplicação. Os elementos dessa fase são:

    - Aplicar padrões de codificação e teste: os padrões de codificação ajudam os desenvolvedores a evitar a introdução de falhas que podem levar a vulnerabilidades de segurança. Por exemplo, a utilização de construções de manipulação de sequências e de *buffer* mais consistentes e seguras pode ajudar a evitar a introdução de vulnerabilidades de *buffer overflow*. As práticas recomendadas e os padrões de testes ajudam a garantir que os testes se concentrem na detecção de possíveis vulnerabilidades de segurança e não apenas na operação correta de funções e recursos do *software*.
    - Aplicar ferramentas de testes de segurança, incluindo ferramentas de *Fuzzing* (difusão): A 'difusão' oferece entradas estruturadas mas inválidas para APIs (*Application Program Interface* - interfaces de aplicativos) de *software* e interfaces de rede, de forma a maximizar a probabilidade de detectar erros que podem levar a vulnerabilidades de *software*.
    - Aplicar ferramentas de verificação de código de análise estática: as ferramentas podem detectar alguns tipos de falhas de código que resultam em vulnerabilidades, incluindo saturações do *buffer*, de números inteiros e variáveis não inicializadas. A Microsoft fez um investimento importante no desenvolvimento dessas ferramentas (as duas que tem sido mais usadas são conhecidas como PREfix e PREfast[1]) e as aperfeiçoa continuamente conforme novos tipos de falhas de código e vulnerabilidades de *software* são descobertos.
    - Realizar revisões de código: as revisões de código complementam os testes e as ferramentas automatizadas; para isso, elas aplicam os esforços de desenvolvedores treinados no exame do código-fonte e na detecção e remoção de possíveis vulnerabilidades de segurança. Elas são uma etapa essencial no processo de remoção de vulnerabilidades de segurança do *software* durante o processo de desenvolvimento.

5. Verificação: a fase de verificação é o ponto em que o *software* está funcionalmente concluído e entra em testes por usuários. Durante essa fase, enquanto o *software* passa por testes, desenvolvedores devem realizar um 'esforço de segurança' que inclui revisões do código de segurança além das concluídas na fase de implementação, bem como testes de segurança direcionados, podendo repetir testes realizados, como testes de difusão. É importante notar que as revisões de código e os testes do código de alta prioridade (aquele que é parte da 'superfície de ataque' do *software*) são críticos para várias partes do SDL. Por exemplo, essas revisões e esses testes devem ser exigidos na fase de implementação para permitir a correção precoce de quaisquer

[1] `http://www.microsoft.com/windows/cse/paprojects.mspx`

problemas, além da identificação e da correção da origem desses problemas. Eles também são críticos na fase de verificação, quando o produto está perto de ser concluído.

6. Liberação e Resposta (Fase de suporte e manutenção): Apesar da aplicação do SDL ou qualquer outro método durante o desenvolvimento, as práticas de desenvolvimento mais avançadas ainda não dão suporte ao fornecimento de aplicações completamente livres de vulnerabilidades, e isso nunca acontecerá. Mesmo que o processo de desenvolvimento pudesse eliminar todas as vulnerabilidades do *software* fornecido, novos ataques seriam descobertos e o *software* que era 'seguro' estaria vulnerável. Parte do processo de resposta envolve a preparação para avaliar relatórios de vulnerabilidades e lançar orientações e atualizações de segurança quando apropriado. O objetivo durante a fase de resposta é aprender a partir dos erros e utilizar as informações fornecidas em relatórios de vulnerabilidade para ajudar a detectar e eliminar mais vulnerabilidades antes que sejam descobertas no campo e utilizadas para colocar os usuários em risco. O processo de resposta também ajuda a equipe de segurança a adaptar processos de forma que erros semelhantes não sejam introduzidos no futuro.

A Microsoft faz uma análise do seu modelo SDL comparando os boletins de segurança em aplicações de grande porte, através de estudos pré-SDL e pós-SDL, com os resultados sugerindo a utilidade do modelo. As equipes de produto do SQL Server e do Exchange Server realizaram esforços de segurança (incluindo modelagem de ameaças, revisões de código e testes de segurança) antes de lançar um *service pack* (um lançamento de *software* que agrega correções para vulnerabilidades de segurança e outros problemas). Os resultados do esforço de segurança do SQL Server foram incorporados ao SQL Server 2000 Service Pack 3, e os resultados do esforço de segurança do Exchange Server foram incorporados ao Exchange 2000 Server Service Pack 3. As Figuras 3.2 e 3.3 mostram os números dos boletins de segurança lançados em períodos iguais antes de depois do lançamento do respectivo service pack (um período de 24 meses para o SQL Server 2000 e de 18 meses para o Exchange 2000 Server) (Lipner, 2004).

A experiência da Microsoft indica que o SDL é eficiente na redução da incidência de vulnerabilidades de segurança. A implementação inicial do SDL (no Windows Server 2003, no SQL Server 2000 Service Pack 3 e no Exchange 2000 Server Service Pack 3) resultou em aprimoramentos significativos na segurança do software, e as versões subseqüentes, que refletem os aperfeiçoamentos do SDL, aparentemente estão mostrando ainda mais avanços de segurança. Atualmente, o modelo com os dezessete processos continua em evolução, com ferramentas sendo desenvolvidas e aprimoradas para todos os aspectos de segurança.

O desenvolvimento e a implementação do ciclo de vida do desenvolvimento da segurança representa um investimento importante e uma mudança essencial na forma como o *software* é criado, desenvolvido e testado. A proposta do presente estudo se enquadra nas etapas de treinamento, desenho e implementação do SDL, com abordagem semelhante de avaliação de resultados, conforme será detalhado nos capítulos seguintes.

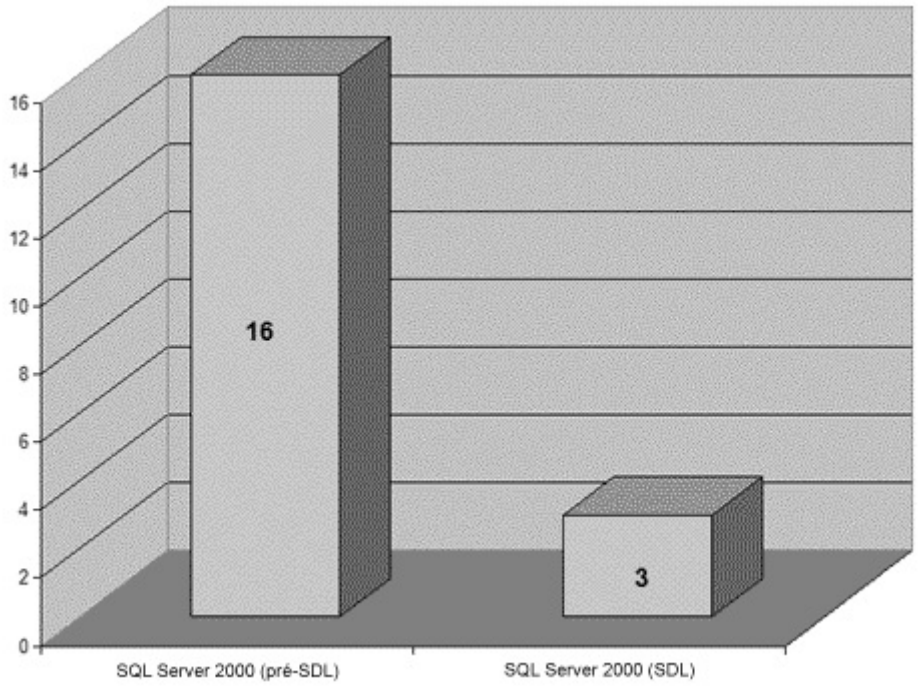

Figura 3.2: Comparativo de uso do SDL para o SQL Server 2000 - Fonte: (Lipner, 2004)

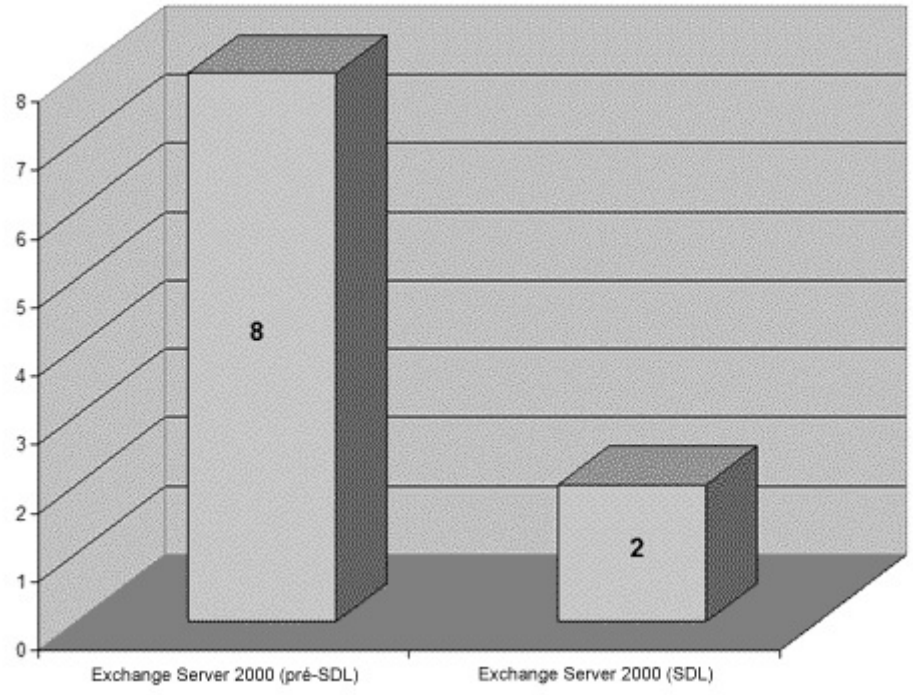

Figura 3.3: Comparativo de uso do SDL para o Exchange Server 2000 - Fonte: (Lipner, 2004)

3.2 OWASP CLASP

O modelo OWASP CLASP veio atender a demanda de desenvolvimento de aplicações seguras onde a aplicação de modelos extensos e complexos como o SDL não era viável, seja por falta de recursos, pela acentuada curva de aprendizado dos modelos ou pela ausência de necessidade, em virtude do tamanho da organização ou da aplicação.

O projeto CLASP foi inicialmente desenvolvido pela empresa *Secure Software* e depois passou para a responsabilidade do OWASP. É um conjunto de componentes de processo dirigido por atividades e baseado em regras, articulando práticas para a construção de aplicações seguras. Seu grande diferencial é poder ser integrado a qualquer processo de desenvolvimento de *software*, devido a sua característica modular, projetado para ser de fácil utilização. Possui um enfoque prescritivo, documentando as atividades que as organizações devem realizar, referenciando os recursos de segurança que possibilitam a implementação dessas atividades (OWASP, 2011b). Apesar de estar atualmente inativo, produziu um livro sobre desenvolvimento de aplicações seguras e ainda é utilizado de diversas formas nesse sentido (Viega, 2005).

A estrutura do CLASP e as dependências entre os componentes do processo são organizados em três categorias:

1. Visões CLASP: analisa o processo de desenvolvimento sob perspectivas de alto nível, subdividida em cinco categorias, conforme Figura 3.4:

    - Visão de conceitos (I): visão geral de como funciona o processo e como seus componentes interagem. Nessa categoria são introduzidas as práticas recomendadas de segurança e a interação entre o CLASP e a política de segurança.
    - Visão baseada em regra (II): apresenta as responsabilidades básicas de cada membro envolvido no projeto de segurança, relacionando-os com as atividades propostas e os requisitos necessários a cada função.
    - Visão de avaliação de atividades (III): descreve o propósito de cada atividade, incluindo o custo de implementação, aplicabilidade, impacto e riscos associados.
    - Visão de implementação (IV): descreve o conteúdo das 24 atividades definidas pelo modelo, identificando os responsáveis pela implementação das mesmas.
    - Visão de vulnerabilidade: apresenta um catálogo com 104 tipos de vulnerabilidades no desenvolvimento de aplicações, divididas em categorias (erros de tipo e limites de tamanho, problemas do ambiente, erros de sincronização e temporização, erros de protocolo e erros lógicos em geral). As atividades de gerenciamento de risco se encontram nessa etapa.

2. Recursos CLASP: fornecem acesso para os artefatos que são utilizados na implantação da ferramenta, composto de onze artefatos no total. A lista de artefatos e as visões onde podem ser aplicados encontra-se na Tabela 3.1.

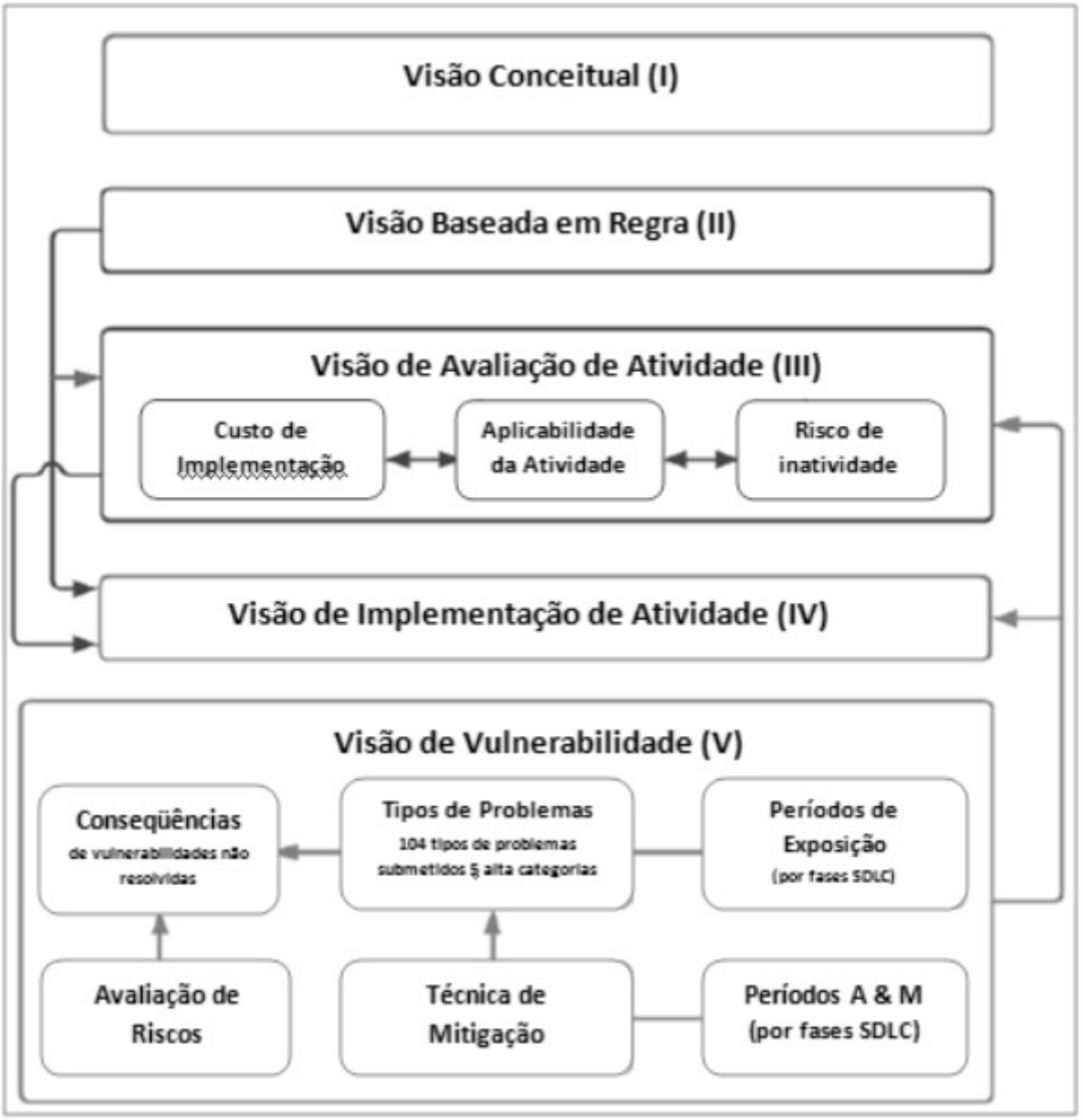

Figura 3.4: Visões modelo OWASP CLASP - Fonte: (OWASP, 2011b)

Tabela 3.1: Lista de artefatos CLASP - Fonte: (OWASP, 2011b)

Recurso	Visão
Princípios básicos em segurança de aplicações	Todas
Exemplo de princípio básico: validação de entrada	Todas
Exemplo de princípio básico: modelo de penetração	Todas
Serviços essenciais de segurança	III
Planilha com exemplos de codificação	II, III e IV
Planilha de avaliação de sistemas	III e IV
Mapa de caminho exemplo: projeto legado	III
Mapa de caminho exemplo: novo projeto	III
Plano de engenharia de processo	III
Formação de equipe	III
Glossário Equipe de Segurança	Todas

3. Caso de Uso de Vulnerabilidade: descrevem condições nas quais os serviços de segurança podem se tornar vulneráveis, fornecendo exemplos específicos e de fácil entendimento para o usuário. A relações causa e efeito são utilizadas como metodologia, com possíveis resultados de exploração de vulnerabilidades em serviços de segurança básicos como autorização e autenticação. Cada caso de uso é composto pelo diagrama com a descrição de cada um dos itens. Um exemplo de caso de uso pode ser encontrado em de Holanda and Fernandes (2009).

Por fim, uma comparação entre os modelos SDL e CLASP pode ser encontrada em Gregoire et al. (2007). A proposta do presente estudo pode ser encaixada em diversas atividades do modelo OWASP CLASP, especialmente nas atividades das visões IV e V, como será detalhado nos capítulos subsequentes.

3.3 OWASP Top 10

A inciativa OWASP constitui-se de uma comunidade sem fins lucrativos cujo principal objetivo é mudar a cultura de desenvolvimento de aplicações *web*, de forma que códigos mais seguros sejam produzidos. Para isso, informações sobre segurança em aplicações são fornecidas de forma gratuita , geralmente organizadas em projetos com objetivos bem definidos. Foi estabelecido como uma organização internacional em 2004 e pretende ajudar outras organizações a desenvolver, operar e manter sistemas que sejam confiáveis (OWASP, 2011a). Por ser livre de relações comerciais ou governamentais, a informação disponibilizada é livre de influências externas, com foco nos conceitos e usabilidade, sem citar preferência por produtos.

Um desses projetos é o OWASP Top 10 2013, o mais popular atualmente na comunidade, que conta com aproximadamente outros 200 projetos ativos. O objetivo é aumentar a conscientização sobre segurança em aplicações *web*, através da identificação dos riscos mais críticos que as organizações enfrentam. Para isso, pretende educar desenvolvedores, projetistas, arquitetos, gestores e organizações sobre as consequências das mais importantes vulnerabilidades de segurança de

aplicações *web*. O projeto foi traduzido integralmente para diversos outros idiomas, inclusive o português brasileiro.

O Top 10 teve a sua primeira versão lançada em 2003, com pequenas alterações em 2004 e 2007. Versões subsequentes bastante modificadas foram lançadas em 2010 e 2013, ambas priorizando os itens por risco, sendo essa última (2013) a versão atual que será utilizada no presente trabalho (OWASP, 2013).

A fonte de informação que forma o OWASP Top 10 é baseada em oito conjuntos de dados de sete empresas que se especializaram em segurança de aplicações, abrangendo mais de 500.000 vulnerabilidades em centenas de organizações e milhares de aplicações. Os itens Top 10 são selecionados e priorizados de acordo com dados de prevalência, em combinação com estimativas do consenso da exploração, detecção e impacto. Os conceitos estão organizados para cada risco, conforme parâmetros da Figura 3.5.

Agentes de Ameaça	Vetores de Ataque	Prevalência da Vulnerabilidade	Detecção Vulnerabilidade	Impactos Técnicos	Impactos no Negócio
Específico da Aplicação	Fácil	Generalizada	Fácil	Severo	Específico do Negócio/ Aplicação
	Média	Comum	Média	Moderado	
	Difícil	Rara	Difícil	Pequeno	

Figura 3.5: Esquema de classificação OWASP Top 10 - Fonte: (OWASP, 2013)

Os riscos atuais na versão 2013, em ordem são:

- A1 - Injeção: As falhas de Injeção, tais como injeção de SQL (*Structured Query Language*), de SO (Sistema Operacional) e de LDAP (*Lightweight Directory Access Protocol*), ocorrem quando dados não confiáveis são enviados para um interpretador como parte de um comando ou consulta. Os dados manipulados pelo atacante podem iludir o interpretador para que este execute comandos indesejados ou permita o acesso a dados não autorizados.

- A2 - Quebra de autenticação e gerenciamento de sessão: As funções da aplicação relacionadas com autenticação e gerenciamento de sessão geralmente são implementadas de forma incorreta, permitindo que os atacantes comprometam senhas, chaves e *tokens* de sessão ou, ainda, explorem outra falha da implementação para assumir a identidade de outros usuários.

- A3 - *Cross-site scripting* (XSS): Falhas XSS ocorrem sempre que uma aplicação recebe dados não confiáveis e os envia ao navegador sem validação ou filtro adequados. XSS permite aos atacantes executarem scripts no navegador da vítima que podem 'sequestrar' sessões do usuário, desfigurar sites, ou redirecionar o usuário para sites maliciosos. Pode-se dizer que o XSS é um tipo específico de injeção.

- A4 - Referência insegura e direta a objetos: Uma referência insegura e direta a um objeto ocorre quando um programador expõe uma referência à implementação interna de um objeto, como um arquivo, diretório, ou registro da base de dados. Sem a verificação do controle de acesso ou outra proteção, os atacantes podem manipular estas referências para acessar dados não-autorizados.

- A5 - Configuração incorreta de segurança: Uma boa segurança exige a definição de uma configuração segura e implementada na aplicação, frameworks, servidor de aplicação, servidor web, banco de dados e plataforma. Todas essas configurações devem ser definidas, implementadas e mantidas, já que geralmente a configuração padrão é insegura. Adicionalmente, o *software* deve ser mantido atualizado.

- A6 - Exposição de dados sensíveis: Muitas aplicações web não protegem devidamente os dados sensíveis, tais como cartões de crédito, IDs fiscais e credenciais de autenticação. Os atacantes podem roubar ou modificar esses dados desprotegidos com o propósito de realizar fraudes de cartões de crédito, roubo de identidade, ou outros crimes. Os dados sensíveis merecem proteção extra como criptografia no armazenamento ou em trânsito, bem como precauções especiais quando trafegadas pelo navegador.

- A7 - Falta de função para controle de nível de acesso: A maioria das aplicações web verificam os direitos de acesso em nível de função antes de tornar essa funcionalidade visível na interface do usuário. No entanto, as aplicações precisam executar as mesmas verificações de controle de acesso no servidor quando cada função é invocada. Se estas requisições não forem verificadas, os atacantes serão capazes de forjar as requisições, com o propósito de acessar a funcionalidade sem autorização adequada.

- A8 - *Cross-site request forgery* (CSRF): Um ataque CSRF força a vítima que possui uma sessão ativa em um navegador a enviar uma requisição HTTP forjada, incluindo o cookie da sessão da vítima e qualquer outra informação de autenticação incluída na sessão, a uma aplicação web vulnerável. Esta falha permite ao atacante forçar o navegador da vítima a criar requisições que a aplicação vulnerável aceite como requisições legítimas realizadas pela vítima.

- A9 - Utilização de componentes vulneráveis conhecidos: Componentes, tais como bibliotecas, frameworks, e outros módulos de software quase sempre são executados com privilégios elevados. Se um componente vulnerável é explorado, um ataque pode causar sérias perdas de dados ou o comprometimento do servidor. As aplicações que utilizam componentes com vulnerabilidades conhecidas podem minar defesas e permitir uma gama de possíveis ataques e impactos.

- A10 - Redirecionamentos e encaminhamentos inválidos: Aplicações web frequentemente redirecionam e encaminham usuários para outras páginas e sites, e usam dados não confiáveis para determinar as páginas de destino.

Sem uma validação adequada, os atacantes podem redirecionar as vítimas para sites de *phishing* ou *malware*, ou usar encaminhamentos para acessar páginas não autorizadas.

Além das informações da Figura 3.5 para cada um dos riscos, o projeto apresenta mais quatro categorias de informação:

- Estou vulnerável? - apresenta as principais atividades que devem ser realizadas para verificar se a aplaicação pode estar vulnerável ao risco em questão.
- Como faço para evitar? - de forma resumida, lista ações que devem ser realizadas para se evitar o risco.
- Exemplos de cenários de ataque - exibe para algumas linguagens como o ataque pode ser executado, com exemplos de código para melhor compreensão.
- Referências - apresenta *links* para outras fontes de informações, internas e externas, relacionadas ao risco.

3.4 OWASP ASVS

Outro projeto que será utilizado nesse estudo é o OWASP ASVS 2014 *Application Security Verification Standard*, cujo principal objetivo é fornecer informações sobre o processo de verificação de aplicações *web*, pretendendo tornar-se um padrão que seja facilmente adotado em todos os tipos de organizações. Em sua descrição, relata que os requisitos foram definidos para serem utilizados (OWASP, 2014):

- Como métrica: ao fornecer para desenvolvedores de sistema um método de avaliação que possa aferir o nível de confiança das aplicações desenvolvidas;
- Como guia: ao funcionar como um guia para desenvolvedores sobre quais controles de segurança devem ser implantados de acordo com os requisitos do sistema especificado;
- Em processos de compra: ao prover uma base para a especificação de requisitos de segurança que devem ser verificados em contratos.

A última versão do projeto, lançada em agosto de 2014, busca atender igualmente as necessidades tanto dos desenvolvedores de sistema como dos especialistas de segurança responsáveis pelas verificações, sendo esse o maior desafio relatado no projeto. As verificações são organizadas em treze áreas e cada uma delas possui atividades de verificação em quatro níveis de maturidade, conforme Figura 3.6.

A quantidade de itens de verificação para cada item é apresentada na Tabela 3.2. Conforme modelos de maturidade de outras áreas, para se atingir um nível, todas as atividades desse nível e do nível anterior precisam estar satisfeitas. Dessa forma, para se atingir o nível 2 (N2) na área de autenticação (V2), tanto as

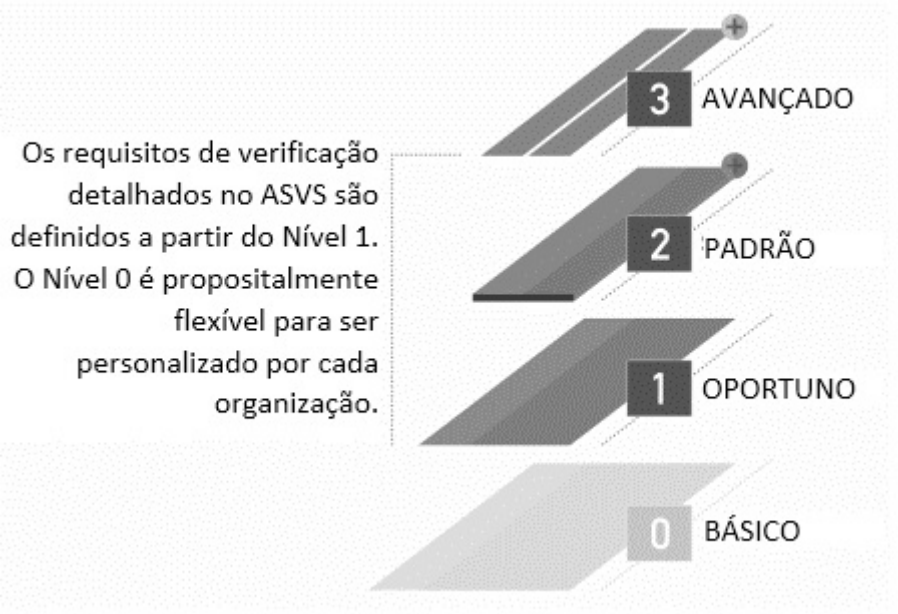

Figura 3.6: Níveis de maturidade do modelo OWASP ASVS - Fonte: adaptado de (OWASP, 2014)

8 atividades de N1 como as 11 atividades de N2 devem estar sendo executadas. Para o nível 3 (N3) na área de controle de acesso (V4), um total de 13 atividades dentros dos 3 níveis devem ter sido implementadas.

Os quatro níveis de maturidade são descritos a seguir:

- N0 (Nível 0) - Básico (*Cursory*): foi concebido sem nenhuma atividade, de forma que a organização que já tenha seu conjunto mínimo de atividades possa utilizá-lo. Recomenda-se a definição de atividades de forma clara, realística e verificável para que a consistência do modelo seja mantida.

- N1 (Nível 1) - Oportuno (*Opportunistic*): possui um conjunto básico de verificações que podem ser feitas rapidamente, ideal para ser utilizado como meta inicial em organizações que estejam em seu início de programa de segurança. Aborda os conceitos básicos mas ainda fundamentais para verificação de possíveis vulnerabilidades. Possui 49 verificações distribuídas em 10 áreas do modelo.

- N2 (Nível 2) - Padrão (*Standard*): garante que procedimentos validados de segurança estão implementados, de forma eficiente e alinhado com as regras de negócio. Espera-se que a grande maioria das organizações encontrem-se nesse nível ou pelo menos próximo em relação à maturidade. Aplicações que atendam esse nível teoricamente só seriam comprometidas com avançadas técnicas de ataque ou comprometimento de pessoal interno. Possui 72 verificações distribuídas em 12 áreas do modelo.

- N3 (Nível 3) - Avançado (*Advanced*): Para esse nível de maturidade, atividades de 9 das áreas devem ser observadas durante as fases de desenvolvi-

Tabela 3.2: Relação entre áreas e níveis de maturidade

Área de Verificação	N1	N2	N3	Total
V2. Autenticação	8	11	2	21
V3. Gerenciamento de sessão	7	7	0	14
V4. Controle de acesso	8	4	1	13
V5. Manipulação de input malicioso	9	4	3	16
V7. Criptografia em repouso	0	5	2	7
V8. Manipulação e log de erros	1	8	5	14
V9. Proteção de dados	2	2	4	8
V10. Comunicações	1	4	4	9
V11. HTTP	3	4	0	7
V13. Controles maliciosos	0	0	11	11
V15. Lógica de negócio	0	10	0	10
V16. Arquivo e recurso	6	4	0	10
V17. Portáteis	4	9	15	28
Total de cada nível	49	72	47	—
Total acumulado N1+N2	—	121	—	—
Total acumulado N1+N2+N3	—	—	168	—

mento e desenho no desenvolvimento de aplicações. É recomendado como meta para aplicações críticas relacionadas à proteção de vidas, defesas de perímetros, atividades financeiras de alta demanda ou aplicações que sejam estratégicas para a organização. Ameaças à segurança nesse nível de maturidade terão como fonte atacantes direcionados para comprometimento de dados específicos da organização.

Para ilustrar, listam-se abaixo algumas atividades da área V2 - Autenticação:

- Nível 1:
  - V2.1 - Verificar que todas as páginas e recursos exigem autenticação, exceto aquelas especificamente direcionadas ao público em geral;
  - V2.2 - Verificar se todos os campos de senha não repetem a senha do usuário quando preenchido;
  - V2.3 - Verificar se todos os controles de autenticação são obrigatórios do lado do servidor;
- Nível 2:
  - V2.9 - Verificar se os usuários podem mudar suas credenciais através de um sistema que é tão seguro quanto o primeiro mecanismo de autenticação;
  - V2.12 - Verificar se todas as tentativas de autenticação são salvas em um log, incluindo as que falharem;
- Nível 3:

– V2.5 - Verificar se todos os controles de autenticação possuem uma implementação central;

– V2.26 - Verificar se mecanismos de reautenticação, autenticação adaptativa ou combinada com outros fatores está implementado em todas as aplicações que contenham dados sensíveis para as operações.

Para a listagem completa de todas as 168 atividades das treze áreas, consultar OWASP (2014).

3.5 MITRE CWE

O MITRE [2] é uma organização sem fins lucrativos patrocinada pelo governo federal americano que atua em pesquisa e desenvolvimento em diversas áreas de interesse público, sendo SegInfo uma delas. É responsável por manter e atualizar uma lista de fraquezas encontradas em aplicações que ocorrem nas fases de arquitetura, desenho, codificação e implementação de sistemas. Essa extensa lista, denominada CWE (*Common Weakness Enumeration*) foi criada para ser uma linguagem padrão de segurança em aplicações, de forma que possa ser utilizada para identificar, mitigar e prevenir tentativas de exploração dessas vulnerabilidades. O principal objetivo da organização com o CWE é "ajudar a moldar e amadurecer a avaliação de segurança de código e também acelerar drasticamente o uso e utilidade das capacidades de verificação de aplicação das organizações na revisão dos sistemas que forem adquirir ou desenvolver" (MITRE, 2010).

O projeto tem contribuição de diversos setores da sociedade como acadêmicos, especialistas em segurança de empresas, institutos de pesquisa independentes e governo. Em parceria com a SANS [3], o MITRE disponibilizou também em 2010 e 2011 uma lista semelhante ao OWASP Top 10, com os 25 maiores erros em desenvolvimento de sistemas que acarretam vulnerabilidades nas aplicações - *2011 CWE/SANS Top 25 Most Dangerous Software Errors*(MITRE/SANS, 2011).

As fraquezas listadas na lista CWE incluem defeitos, falhas, bugs, vulnerabilidades e outros erros no código da aplicação que, se não tratada, pode resultar em sistemas e redes vulneráveis a ataques. Exemplos de pontos fracos listados incluem sobrecarga de *buffer*, formatação de strings; estrutura problemáticas; manipulações de elementos especiais; erros de manipulação de variáveis; erros de interface de usuário; erros de autenticação; erros de gestão dos recursos; verificação insuficiente de dados; aleatoriedade e previsibilidade.

Para cada fraqueza listada no CWE, as seguintes informações são fornecidas:

- Número Identificador / Nome da fraqueza
- Descrição do tipo de fraqueza
- Termos alternativos para a fraqueza
- Descrição do comportamento da fraqueza

[2] www.mitre.org
[3] www.sans.org

- Descrição da exploração da fraqueza
- Probabilidade de explorar a fraqueza
- Descrição das consequências da exploração
- Mitigações potenciais
- Informações de relacionamento com outras fraquezas
- Taxonomia
- Exemplos de código para as linguagens/arquiteturas
- Números CVE Identificador de vulnerabilidades para as quais exista referências para esse tipo de fraqueza

Em relação ao tipo de fraqueza, este pode ser classificado conforme a seguir:

- Elemento Composto (*Compound Element*): Uma entrada que consiste de duas ou mais fraquezas distintas, nos quais todas as fraquezas devem estar presentes ao mesmo tempo para que uma potencial vulnerabilidade possa surgir. A remoção de qualquer uma das fraquezas elimina ou reduz drasticamente o risco.
- Classe de Fraqueza (*Weakness Class*): Uma fraqueza que é descrito de uma forma muito abstrata, normalmente independente de qualquer linguagem ou tecnologia específica. É mais geral do que uma Base de fraqueza.
- Base de Fraqueza (*Weakness Base*): Uma fraqueza que é descrita de uma forma abstrata, mas com detalhes suficientes para inferir métodos específicos para detecção e prevenção. É mais geral do que uma Variante de fraqueza *Variant*, mas mais específico do que uma Classe de fraqueza *Class*.
- Variante de Fraqueza (*Weakness Variant*): Uma fraqueza que é descrita em um nível muito baixo de detalhe, normalmente limitada a uma linguagem ou tecnologia específica. Ele é mais específico do que uma Base de fraqueza.
- Categoria (*Category*): A entrada CWE que contém um conjunto de outras entradas que compartilham uma característica comum.
- Vista (*View*): Um subconjunto de entradas CWE que fornece uma maneira de examinar o conteúdo CWE. Os dois principais tipos de vista são *Slices* (listas planas) e gráficos (contendo as relações entre entradas).

O CWE possui uma relação direta com o OWASP Top 10 através do ID 928 *Weaknees in OWASP Top 10 2013*. Cada um dos dez riscos é uma outra entrada CWE do tipo 'categoria', que define classes e bases de fraqueza relacionados aos riscos, desde o CWE 929 (para o A1) até o CWE 938 (para o A10). No total, existem 30 fraquezas, 14 categorias e 2 elementos compostos relacionadas aos riscos listados no OWASP Top 10 conforme Figura 3.7. Essa relação será utilizada na ontologia, conforme descrito nas seções posteriores.

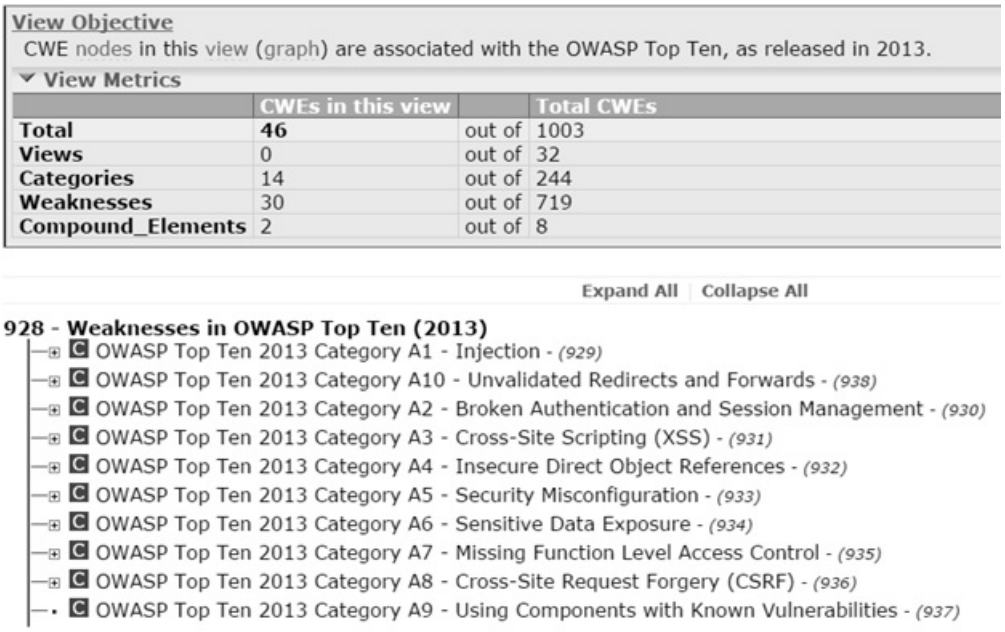

Figura 3.7: Entrada do modelo CWE-928 - Weaknesses in OWASP Top Ten (2013) - Fonte: (CWE, 2014)

3.6 Modelo de Avaliação CnA

A ideia do presente estudo teve como fonte um projeto desenvolvido por uma organização de cunho diplomático com representação em mais de 200 países, que começou a observar que problemas com segurança no desenvolvimento de aplicações estavam se tornando cada vez mais frequentes. Uma das etapas do processo de desenvolvimento de sistemas dessa organização consiste em avaliar o sistema desenvolvido em relação a possíveis vulnerabilidades, de acordo com a fase de 'Teste de Segurança' do modelo SDL proposto pela Microsoft, conforme detalhado em 3.1. Esse teste, denominado *Certification and Accreditation* (CnA), exige uma pontuação mínima obrigatória para que o sistema entre em produção. Não sendo atingida essa pontuação (atualmente em 6,0), o sistema precisa ser revisto e corrigido para que os riscos sejam controlados.

Realizou-se um levantamento para verificar como o processo de CnA estava se comportando ao longo dos anos, com dados de 2010 a 2012 (três anos), e com 2013 sendo adicionado recentemente conforme apresentado na Figura 3.8. Em média, são desenvolvidas 140 aplicações por ano, considerando todas as missões da organização espalhadas pelo mundo. Observe que a quantidade de aplicações que passam pelo processo apenas uma vez vem diminuindo ao longo dos anos, com 60% em 2010 e 29% em 2013. Ao mesmo tempo, observa-se um aumento no número de aplicações sendo submetidas duas ou mais vezes, com 8% em 2010 e 20% em 2013 para três ou mais submissões, por exemplo. Além do evidente retrabalho que acaba por atrasar o início do lançamento da aplicação em produção, há de se considerar também o custo e o tempo envolvido para cada um dos processos de análise.

Segundo especialistas em segurança responsáveis pelas pontuações das análises

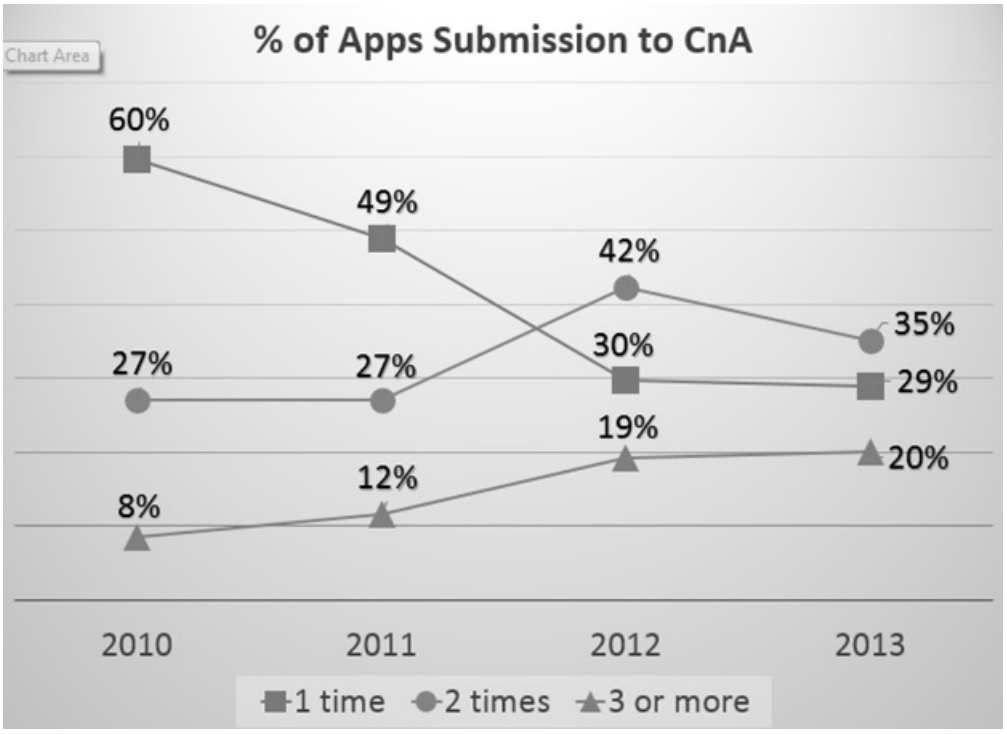

Figura 3.8: Distribuição do retrabalho em processos de CnA

do CnA, o aumento do retrabalho se deve principalmente ao aumento do número de vulnerabilidades descobertas e incorporadas ao processo ao longo dos anos, sem que as preocupações com a segurança tenham evoluído no mesmo ritmo. Em suas previsões, confirmam o cenário preocupante apresentado no Capítulo 1, de que novas abordagens precisam ser sugeridas para auxiliar no processo de desenvolvimento de aplicações seguras.

Em resposta a esses dados, a organização iniciou em 2012 um projeto de treinamento sobre segurança em desenvolvimento de aplicações para todos os seus funcionários que efetivamente participavam do processo de desenvolvimento. O projeto consistiu de um curso sobre desenvolvimento seguro de 40 horas, que teve diversos projetos da iniciativa OWASP, incluindo o OWASP TOP 10 e o OWASP ASVS, como principal fonte de conhecimento.

O curso foi conduzido por especialistas em SegInfo que possuíam também experiência em desenvolvimento de sistemas, de modo que poderiam identificar quais eram os maiores desafios para o desenvolvimento seguro dentro da realidade da rotina diária do desenvolvedor. Como parte do processo de treinamento, realizou-se também uma pesquisa com os desenvolvedores da qual destacamos dois itens: 'grau de interesse em cursos e certificações de segurança' e 'grau de interesse de assumir uma posição como analista de segurança mantendo-se o mesmo salário'.

Conforme pode ser observado na Figura 3.9, a imensa maioria dos 286 desenvolvedores que respondeu à pesquisa não tem interesse em elevar seu conhecimento sobre segurança, ao menos de um jeito formal, através de cursos e certificações da área. Não possuem, da mesma forma, interesse em deixar a área de desen-

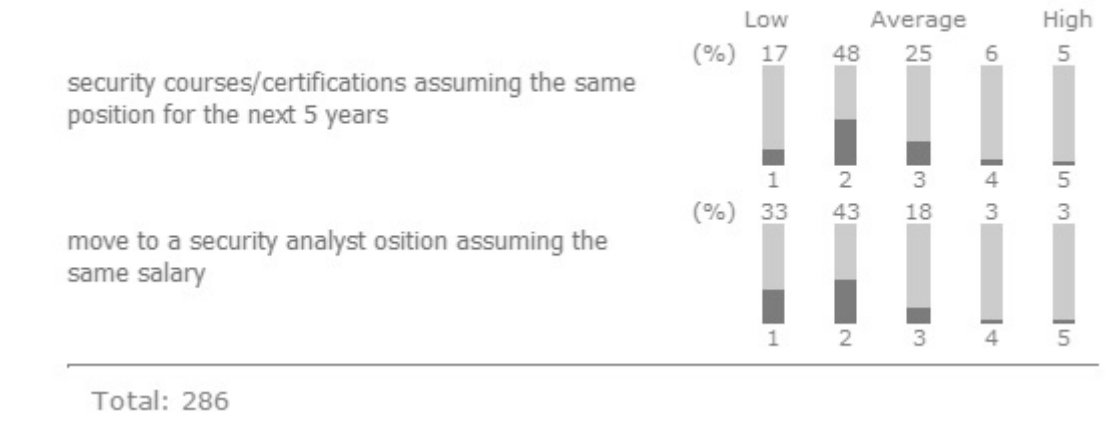

Figura 3.9: Resultado de pesquisa sobre interesse de desenvolvedores em temas sobre SegInfo

volvimento para optar pela carreira de analista de segurança. Ressalta-se que a pesquisa foi preenchida por profissionais de diversos países, com diferentes experiências e realidades em relação ao mercado de trabalho e vivências profissionais.

A eficácia do projeto só poderá ser verificada a partir dos dados completos das etapas de CnA para o ano de 2014. Entretanto, o resultado parcial de junho de 2014 indicava uma redução para 24% em aplicações sendo submetidas duas vezes ao processo de CnA (35% em 2013) e redução para 16% em aplicações sendo submetidas três ou mais vezes (20% em 2013), sugerindo que o projeto possa estar trazendo resultados iniciais positivos.

Por questões de recurso e tempo, capacitar através de um treinamento de 40 horas todos os profissionais que participam das etapas de desenvolvimento de sistemas em diversos países não é uma opção plausível para um grande número de organizações, ainda que os bons resultados obtidos se confirmem. Baseado nesse conjunto de experiências, desenvolveu-se a proposta aqui apresentada.

Capítulo 4

Ontologia

Nos últimos anos, o aumento exponencial dos dados disponíveis tem conferido importância significativa às técnicas de organização da informação. Essas técnicas fazem parte de um corpo de disciplinas que busca melhorias no tratamento de dados, atuando na sua seleção, no seu processamento, na sua recuperação e na sua disseminação.

Nesse sentido, ontologias tem sido cada vez mais usadas em diferentes áreas de estudo, especialmente para organizar informação sobre determinado assunto ou formalizar conhecimento que se encontra de forma desestruturada. Dentro da área de Ciência da Computação, tem recebido atenção especial e crescente a medida que estudos demonstram que projetos de TI são longos e caros, justificando o sucesso dessa abordagem (Bai and Zhou, 2011).

Segundo Gruber (1993), ontologia é uma especificação explícita de uma contextualização. Essa definição vem evoluindo com o tempo, sendo a encontrada em Guarino (1998) outra definição inúmeras vezes referenciada na literatura: 'ontologia refere-se a um artefato de engenharia, constituído por um vocabulário específico utilizado para descrever determinada realidade, com a adição de um conjunto de suposições acerca do significado pretendido desse vocabulário'. Diversas outras definições podem ser encontradas, mas a maioria simplesmente complementa ou expande o significado das duas aqui já mencionadas. Para o presente estudo, a ontologia pode ser definida como uma ferramenta para organização e representação do conhecimento sobre determinados conceitos na área de SegInfo.

Uma ontologia é criada por especialistas e define as regras que regulam a combinação entre termos e relações em um domínio do conhecimento. Os usuários formulam consultas usando conceitos definidos pela ontologia. O que se busca, em última instância, são melhorias nos processos de recuperação da informação.

Uma visão geral sobre ontologias pode ser encontrada em Almeida and Bax (2003), onde são classificadas em relação a diferentes aspectos, conforme reproduzido a seguir:

1. Quanto à função (Mizoguchi et al., 1995):

    - Ontologias de domínio: reutilizáveis no domínio, fornecem vocabulários sobre conceitos, seus relacionamentos, sobre atividades e regras que os governam.

- Ontologias de tarefa: fornecem um vocabulário sistematizado de termos, especificando tarefas que podem ou não estar no mesmo domínio.
- Ontologias gerais: incluem um vocabulário relacionado a coisas, eventos, espaço, casualidade, comportamento, funções, etc.

2. Quanto ao grau de formalismo (Uschold and Gruninger, 1996):

   - Ontologias altamente informais: expressa livremente de forma não-estruturada.
   - Ontologias semi-informais: expressa como uma linguagem natural de forma restrita e estruturada.
   - Ontologias semiformais: expressa em uma linguagem artificial definida formalmente.
   - Ontologias rigorosamente formais: os termos são definidos com semântica formal, teoremas e provas.

3. Quanto à aplicação (Jasper et al., 1999):

   - Ontologias de autoria neutra: um aplicativo é escrito em uma única língua e depois convertido para uso em diversos sitemas, reutilizando-se as informações.
   - Ontologias como especificação: cria-se uma ontologia para um domínio, a qual é usada para documentação e manutenção no desenvolvimento de aplicações.
   - Ontologias de acesso comum à informação: quando o vocabulário é inacessível, a ontologia torna a informação inteligível, proporcionando conhecimento compartilhado dos termos

4. Quanto à estrutura (Haav and Lubi, 2001):

   - Ontologias de alto nível: descrevem conceitos gerais relacionados a todos os elementos da ontologia, os quais são independentes do problema ou domínio.
   - Ontologias de domínio: descrevem o vocabulário relacionado a um domínio, como, por exemplo, medicina ou automóveis.
   - Ontologias de tarefa: descrevem uma tarefa ou uma atividade, como, por exemplo, diagnósticos ou compras, mediante inserção de termos especializados na ontologia.

5. Quanto ao conteúdo (Van Heijst et al., 1997):

   - Ontologias terminológicas: especificam termos que serão usados para representar o conhecimento em um domínio.
   - Ontologias de informação: especificam a estrutura de registros de bancos de dados (por exemplo, os esquemas de bancos de dados).

- Ontologias de modelagem do conhecimento: especificam conceitualizações do conhecimento, tem uma estrutura interna semanticamente rica e são refinadas para o uso do domínio do conhecimento que descrevem.
- Ontologias de aplicação: contém as definições necessárias para modelar o conhecimento em uma aplicação.
- Ontologias de domínio: expressam conceitualizações que são específicas para um determinado domínio do conhecimento.
- Ontologias genéricas: similares às ontologias de domínio, mas os conceitos que as definem são considerados genéricos e comuns a vários campos.
- Ontologias de representação: explicam as conceitualizações que estão por trás dos formalismos de representação do conhecimento.

Os diferentes tipos de ontologias podem se relacionar ou serem derivadas de outras. Por exemplo, Ontologias de alto nível (ou fundamentação) que são construídas sobre conceitos de alta generalização, podem ser aplicados em diferentes domínios. Ontologias de domínio podem fazer a instanciação dos conceitos encontrados na ontologia de fundamentação. Nesse trabalho, propõe-se a construção de uma ontologia de domínio (em relação à função), semi-informal, como especificação, de domínio (em relação à estrutura) e de aplicação, para os conceitos relacionados à área de SegInfo.

Os termos essenciais que precisam ser compreendidos para a construção de uma ontologia, segundo Novello (2002), são:

- Conceito: é a representação de algo acerca do domínio em questão, por exemplo, 'pessoas'. Na maioria das ferramentas são chamados de classes.
- Atributos: são as propriedades dos conceitos, por exemplo, 'idade' relacionada à 'pessoas'.
- Relacionamentos: são as interações entre os conceitos especificados, onde define-se a cardinalidade. Por exemplo, entre os conceitos 'pessoas' e 'escola' pode existir o relacionamento 'estuda em'.
- Funções: relações especiais entre elementos, definidos formalmente.
- Axiomas: é a modelagem de sentenças que são sempre verdadeiras.
- Instâncias: representação dos conceitos e relações que foram estabelecidos. 'Ana' é uma instância de 'pessoa'.

4.1 Critérios e Vantagens

Independente do tipo ou domínio, a construção de uma ontologia requer cuidados para que o produto final possa ser utilizado da forma como projetado, principalmente por ser um trabalho dispendioso e complexo. Para garantir maior

eficiência e interoperabilidade, Uschold and King (1995) define critérios básicos para serem considerados na construção de ontologias:

- Clareza: uma ontologia deve comunicar efetivamente o significado projetado dos termos definidos e, assim, suas definições devem ser objetivas e independentes de contexto.
- Consistência: uma ontologia deve garantir consistência na sua definição, tanto dos axiomas lógicos quanto dos conceitos informais. Coerência também deve ser aplicada para os conceitos definidos informalmente, como aqueles descritos em documentos de linguagem natural e exemplos.
- Extensibilidade: a partir de uma ontologia, deve ser possível definir novos termos para usos específicos, sem haver necessidade de rever definições existentes, sendo projetada para antecipar os usos de um vocabulário compartilhado.
- Compromissos de codificação mínimos: não deve haver dependência em relação a uma tecnologia particular de representação do conhecimento. Agentes que compartilham conhecimento podem ser implementados em diferentes sistemas e estilos de representações.
- Compromissos ontológicos mínimos: uma ontologia deve fazer o mínimo de imposições possíveis, permitindo que as partes comprometidas com a ontologia fiquem livres para especializar e instanciar a ontologia.

A criação de uma ontologia de domínio tem como principal ganho imediato a melhor compreensão do domínio representado. Em relação a desenvolvimento de aplicações, conforme destacado em de Araujo (2003), o uso de uma abordagem ontológica apresenta os seguintes benefícios:

- Propicia ao desenvolvedor uma compreensão mais apurada do domínio abordado;
- Possibilita o compartilhamento de conhecimento, levando em consideração o compartilhamento de termos de um dados domínio;
- Possibilita a troca de informações;
- Oferece suporte à interoperabilidade entre sistemas computacionais, considerando o relacionamento de diferentes paradigmas, linguagens e métodos;
- Auxilia no reuso do conhecimento;
- Auxilia em processos de especificação de requisitos;
- Auxilia no processo de verificação de um sistema computacional, porque ontologias explicitam a especificação de tais sistemas, servindo como base de comparação entre o modelo conceitual e o modelo computacional;
- Auxilia na manutenção de documentação de sistemas computacionais.

4.2 Construção de Ontologias

Para desenvolver uma ontologia, princípios de projeto, atividades e processos de desenvolvimento, semelhante aos utilizados em engenharia de software, devem ser empregados. Surge então a engenharia de ontologias, que tem como objetivo fornecer informações e artefatos que possam deixar esse trabalho menos árduo e mais reprodutível (Gasevic et al., 2006). Com a semelhança entre as duas engenharias, no processo de desenvolvimento de ontologias, usualmente, são aceitas as atividades de especificação, conceitualização, formalização, implementação e manutenção. A cada uma destas atividades existem tarefas a serem executadas, como seguem (Ye et al., 2007):

- Especificação: identificar o propósito e o escopo da ontologia. O propósito responde a questão 'por que a ontologia é construída?', enquanto o escopo responde a questão 'quais são as intenções de uso e usuários da ontologia?'

- Conceitualização: descrever, em modelo conceitual, a ontologia a ser construída, de acordo com as especificações encontradas no estágio anterior. Cabe ressaltar que o modelo conceitual de uma ontologia pode ser construído mediante o emprego de ferramentas formais e informais. Tal modelo consiste em conceitos do domínio, as relações entre os conceitos e as propriedades dos conceitos.

- Formalização: transformar a descrição conceitual em um modelo formal. Nesta fase, conceitos são definidos através de axiomas que restringem as possíveis interpretações de seu significado e também organizados hierarquicamente através de relações estruturais, tais como 'é-um' ou 'parte-de'.

- Implementação: implementar a ontologia formalizada em uma linguagem de representação do conhecimento. Para isso, um pré-requisito é a escolha da linguagem de representação adequada.

- Manutenção: atualizar e corrigir a ontologia desenvolvida, de acordo com o surgimento de novos requisitos.

Com o objetivo de auxiliar esse processo, diversas metodologias surgiram, dentre as quais destaca-se a *Ontology Development 101*, originalmente elaborada por Noy and McGuinness (2001). O Método 101 destaca algumas regras fundamentais para o processo de engenharia de ontologia:

- Não há um modo único correto de modelar um domínio, existem alternativas viáveis.

- O desenvolvimento de ontologia é necessariamente um processo iterativo.

- Conceitos em ontologia devem estar ligados a objetos e relacionamentos em seu domínio de interesse.

O Método 101 é utilizado como um guia, o qual sugere uma abordagem iterativa para o desenvolvimento da ontologia, isto é, partindo-se de uma versão inicial,

a ontologia é revisada e atualizada, evoluindo ao longo do tempo, como representado na Figura 4.1. Em (a) tem-se os sete passos sugeridos pelos pesquisadores, com a ideia de um processo iterativo cíclico representado em (b).

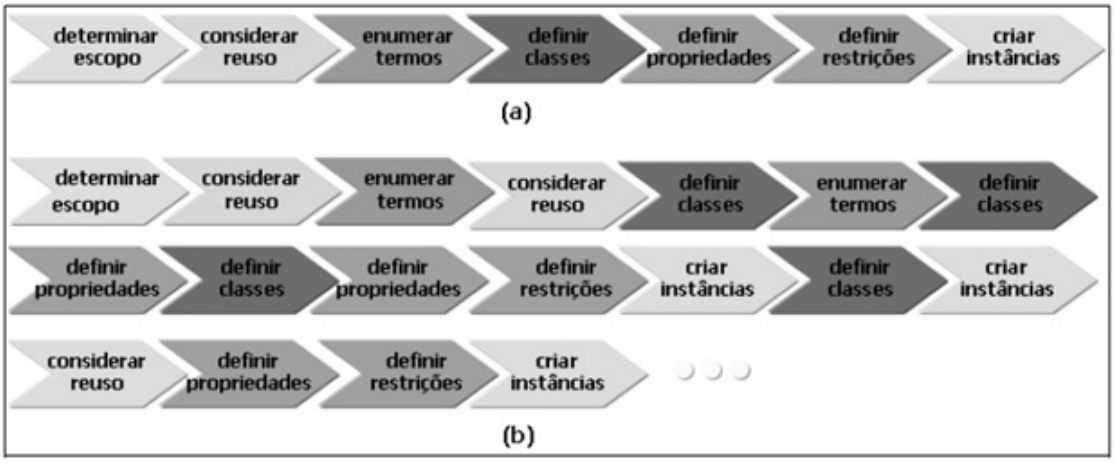

Figura 4.1: Etapas para a construção de ontologias segundo o Método 101 - Fonte: adaptado de (Noy and McGuinness, 2001)

Os sete passos são descritos a seguir:

1. Passo 1 - Determinar o domínio e escopo da ontologia. Para começar definindo o escopo e o domínio da ontologia são apresentadas algumas das questões básicas:

   - Qual é o domínio que a ontologia cobrirá?
   - Qual será o uso da ontologia?
   - Para que tipos de perguntas a ontologia deverá fornecer respostas?
   - Quem vai utilizar e manter a ontologia?

   As respostas a estas questões podem mudar durante o processo de construção da ontologia, mas em determinado momento, podem ajudar a limitar o âmbito de aplicação do modelo. Uma das maneiras de se determinar o alcance da ontologia é esboçar uma lista de perguntas que uma ontologia deve ser capaz de responder. Estas questões irão servir para testar a ontologia posteriormente, podendo ser somente um esboço e não precisam ser exaustivas.

2. Passo 2 - Considerar o reuso de ontologias existentes. O reuso de ontologias pode ser um requisito se o novo sistema interage com outras aplicações que já têm ontologias ou vocabulários controlados. Podemos encontrar bibliotecas de ontologias reusáveis na web. Por meio do reuso de ontologias é possível refinar e estender as fontes existentes para o nosso domínio particular.

3. Passo 3 - Enumerar os termos importantes da ontologia. Os termos sobre os quais podem ser feitas declarações devem ser listados, junto com suas

propriedades. Por exemplo, em uma ontologia sobre vinhos, termos importantes relacionados a vinhos podem ser: localização, cor e uva. Inicialmente, é importante ter uma lista de termos, sem se preocupar com a sobreposição entre os conceitos que eles representam, relações entre os termos e as propriedades que eles podem ter. Os próximos dois passos - definição da hierarquia de classes e das propriedades dos conceitos - estão intimamente interligados. Geralmente, são criadas algumas definições de conceitos na hierarquia e descrevemos propriedades desses conceitos. Depois definimos mais conceitos e assim por diante.

4. Passo 4 - Definir classes e a hierarquia de classes. Há vários enfoques no desenvolvimento de hierarquia:

   - Um processo de desenvolvimento *top-down* começa com a definição dos conceitos mais gerais no domínio e posteriormente, é feita a especialização dos conceitos;
   - Um processo de desenvolvimento *bottom-up* começa com a definição das classes mais específicas, com subsequente agrupamento destas classes em conceitos mais gerais;
   - Um processo de desenvolvimento de combinação é formado pela combinação dos dois enfoques anteriores. São definidos os conceitos que mais se sobressaem primeiro e depois esse conceitos são generalizados e especializados adequadamente.

   Qualquer que seja o enfoque escolhido, geralmente se inicia definindo classes. Da lista formada no Passo 3, são selecionados os termos que descrevem os objetos que possuem uma existência independente. Esses termos serão classes na ontologia. Organiza-se então as classes em uma taxonomia hierárquica, verificando que: se a classe A é uma superclasse de B, então toda instância de B é também uma instância de A. Neste caso, a classe B representa um conceito que é um tipo de A.

5. Passo 5 - Definir as propriedades das classes. Após definir algumas classes, deve-se descrever a estrutura interna dos conceitos. Os termos que sobraram da lista de termos do Passo 3 provavelmente são propriedades. Para cada propriedade na lista, é determinada quais classes ela descreve. Todas as subclasses de uma classe herdam as propriedades desta classe.

6. Passo 6 - Definir os valores das propriedades. Propriedades podem descrever o tipo de valor, valores permitidos e o número de valores (cardinalidade). Uma propriedade pode ter como tipo um valor que é uma instância, cujo escopo é uma classe específica. Em algumas linguagens, como OWL (*Web Ontology Language*), é permitido utilizar tipos de dados no preenchimento de valores de propriedades. Os tipos de dados mais comuns são: cadeia de caracteres, números, booleanos e listas enumeradas de elementos.

7. Passo 7 - Criar instâncias. O último passo é criar instâncias individuais das classes na hierarquia. Para isto é necessário: escolher uma classe, criar uma instância dessa classe e preencher os valores dos campos.

O Método 101 foi utilizado no estudo por proporcionar a construção da ontologia por não especialistas da área de ontologia, além de sugerir um método didático que promove a manutenção das características ontológicas desejadas, como consistência. Existe na literatura diversos outros métodos sugeridos para a construção de ontologias, resumidos a seguir, e que podem ser encotrados com mais detalhes em Kawano (2009):

- Cyc - Codifica manualmente o conhecimento implícito e explícito das diferentes fontes, e, quando já se tem conhecimento suficiente na ontologia, um novo consenso pode ser obtido por ferramentas que utilizam linguagem natural (Lenat and Guha, 1989).

- KACTUS - Método recursivo que consiste em uma proposta inicial para uma base de conhecimento, de forma que quando é necessária uma nova base em domínio similar, generaliza-se a primeira base em uma ontologia, adaptada a ambas as aplicações; quanto mais aplicações, mais genérica a ontologia (Bernaras et al., 1996).

- Sensus - Constrói ontologias a partir de outras ontologias como ponto de partida, identificando os termos relevantes para o domínio e ligando-os à ontologia mais abrangente; um algoritmo monta a estrutura hierárquica do domínio (Swartout et al., 1996).

- On-to-knowledge - Auxilia a administração de conceitos em organizações, identificando metas para as ferramentas de gestão do conhecimento e utilizando cenários e contribuições dos provedores/clientes de informação da organização (Staab et al., 2001).

- Uschold e King - Identifica o propósito, os conceitos e relacionamentos entre os conceitos, além dos termos utilizados para codificar a ontologia e, em seguida, documentá-la (Uschold and Gruninger, 1996).

- Gruninger e Fox - Método formal que identifica cenários para uso da ontologia, utiliza questões em linguagem natural para determinação do escopo da ontologia, executa a extração sobre os principais conceitos, propriedades, relações e axiomas (Grüninger and Fox, 1995).

- Methontology - constrói uma ontologia por reengenharia sobre outra, através do uso do conhecimento sobre o domínio; as atividades principais são especificação, conceitualização, formalização, implementação e manutenção (López et al., 1999).

- OntoClean - fundamentada por conceitos formais que são gerais o suficiente para serem usados em qualquer ontologia, independente do domínio. Dá ênfase aos conceitos básicos de essência, rigidez, identidade e unidade (Guarino and Welty, 2002).

Após a construção, tem início a fase do ciclo de vida da ontologia, onde outras atividades devem ser executadas (Pinto and Martins, 2004):

- Aquisição do conhecimento: adquirir conhecimento sobre um domínio por meio de técnicas de elicitação do conhecimento com especialistas de domínio ou recorrer a bibliografia relevante. Várias técnicas podem ser utilizadas, como *brainstorming*, entrevistas, questionários, análise de texto e técnicas indutivas.

- Avaliação: julgar tecnicamente a qualidade da ontologia por meio da:

  - Avaliação técnica: julgar a ontologia e a documentação diante um padrão de referência, através da verificação (garante a correção da ontologia de acordo com o entendimento aceito sobre o domínio em fontes de conhecimento especializadas) e validação (garante que a ontologia corresponde a sua suposta finalidade, de acordo com os documentos de especificação de requisitos).
  - Avaliação dos usuários: julgar a ontologia do ponto de vista do usuário, em relação a sua usabilidade e utilidade; e do ponto de vista da (re)utilização em outras aplicações conforme a sua documentação.

- Documentação: relatar o que, como e por que foi feito. Uma documentação associada com os termos presentes na ontologia é importante, não somente para melhorar a clareza da ontologia, mas também para facilitar a manutenção, uso e reuso.

4.3 Trabalhos Correlatos

O uso de ontologias de domínio como suporte para atividades relacionadas à SegInfo é encontrado na literatura para finalidades diversas, dentro de ambas as abordagens apresentadas na Seção 1.2. Analisaram-se trabalhos relacionados com ontologias sendo propostos em diferentes níveis de abstração e utilizando diferentes fontes de informação, sempre objetivando a criação de um repositório que possa ser útil para organizações. O maior desafio identificado nas abordagens é definir exatamente como o modelo pode ser utilizado de forma eficiente.

Por exemplo, em Almeida et al. (2010) e de Azevedo et al. (2007), ontologias sobre SegInfo foram criadas com o intuito de classificar a informação para ajudar na definição de que tipo de informação precisar protegida, e como essa proteção deve ser implementada. Esse tipo de abordagem está alinhado com as atividades já citadas de BPM, inclusive com a linguagem SPARQL sendo utilizada para identificar preocupações relacionadas à segurança em um nível mais alto da organização. Consultas com SPARQL são também utilizadas em da Silva et al. (2011) e Martimiano and Moreira (2005), com dados provenientes de incidentes de segurança sendo combinados com outras fontes de informação para validar a ontologia e ajudar na previsão de possíveis ataques futuros.

Em Razzaq et al. (2009) e Razzaq et al. (2014) encontra-se abordagens ontológicas do tipo reativa, onde sistemas de detecção de intrusão são desenvolvidos com ontologias sobre SegInfo, que é atualizada de forma constante a medida que dados reais são recebidos. O sistema tem por principal objetivo a identificação e mitigação de ataques do tipo 'dia-zero', para os quais ainda não há controle

disponível. Proposta similar é apresentada em Rosa et al. (2009), onde uma ontologia é sugerida para detectar ataques por injeção de XML através de *web services*. Também em Vorobiev and Han (2006), o foco do uso da ontologia é sobre ataques realizados via *web services*. Nesses estudos, a ontologia é aplicada para varreduras e avaliações quando a aplicação *web* já está operando em produção.

A formalização de conceitos sobre SegInfo com o uso de ontologias é desenvolvida em Fenz and Ekelhart (2009), sem no entanto o modelo ser efetivamente aplicado em algum cenário. Em Obrst et al. (2012) encontra-se trabalho semelhante, onde também o exemplo de aplicação da ontologia fica proposto para trabalhos futuros.

A contribuição do estudo difere dos trabalhos relacionados em relação a três conceitos principais. Primeiramente, o público-alvo da ontologia é bem definido, sendo este desenvolvedores de aplicações *web*; em segundo lugar, o momento de utilização da ontologia é especificamente na fase de arquitetura/planejamento, para aplicar controles antes que a aplicação seja disponibilizada em produção; por último, a fonte de conhecimento utilizada é única, não sendo encontrada semelhante abordagem na literatura. A proposta assemelha-se a outras no fato do uso da linguagem SPARQL em modelos baseados em representação do conhecimento, sem, no entanto, exigir do utilizador da ontologia conhecimento prévio de SPARQL para alcançar os resultados desejados.

Capítulo 5

Proposta de Solução

5.1 Ontologia OWASP-CWE

O presente estudo tem por objetivo elaborar um repositório consultável de conhecimento sobre SegInfo, que possa ser facilmente manipulado por desenvolvedores de sistemas para obter informação sobre desenvolvimento seguro. O desenvolvimento da proposta iniciou-se com a construção da ontologia:

A ontologia OWASP-CWE foi construída utilizando a metodologia 101 já citada anteriormente de Noy and McGuinness (2001) e as três bases de conhecimento apresentadas no Capítulo 3. A ferramenta empregada foi o construtor de ontologias de código aberto *Protégé* [1], desenvolvido e disponibilizado gratuitamente pela Universidade de *Stanford*, em sua versão 5.0.0 (*build beta-15*).

Iniciou-se a construção respondendo às questões básicas do Passo 1 do método 101:

- Qual é o domínio que a ontologia cobrirá? A ontologia abordará os conceitos relacionados à SegInfo, com foco em desenvolvimento de aplicações *web* seguras, abrangendo informações dos projetos OWASP Top 10 e ASVS, bem como da iniciativa do CWE.

- Qual será o uso da ontologia? Será utilizada para fornecer a um desenvolvedor de sistemas informações que possam auxiliar no desenvolvimento de aplicações *web* mais seguras. Especificamente, o desenvolvedor deverá, na fase anterior ao desenvolvimento, responder à um conjunto de perguntas relacionadas às características da aplicação que está para ser desenvolvida. Com base nessas respostas, o sistema irá procurar na ontologia informações relacionadas à segurança que possam ser úteis para o desenvolvedor.

- Para que tipos de perguntas a ontologia deverá fornecer respostas? 'Quais informações sobre SegInfo podem ser úteis ao desenvolvedor que está prestes a desenvolver um sistema com as características X, Y e Z (definidas pelo questionário)?';'Para uma aplicação que tenha tal característica, com quais riscos eu deveria me preocupar mais? E como evitá-los?'.

[1] `http://protege.stanford.edu`

- Quem vai utilizar e manter a ontologia? A ontologia será utilizada por desenvolvedores de sistemas *web* e mantida por especialistas em segurança da informação.

Seguindo para o Passo 2, consultaram-se repositórios de ontologia mas em nenhum deles encontrou-se uma ontologia que pudesse atender os objetivos do estudo. As poucas relacionadas à SegInfo possuem um nível de abstração muito maior que o necessário para ser utilizada pelo desenvolvedor, isto é, abrigam os conceitos de forma muito genérica, se distanciando da nossa necessidade de uma ontologia mais relacionada à aplicação. A proposta visa oferecer informações sobre SegInfo de forma objetiva e enxuta, sem 'inundar' o desenvolvedor com todos os aspectos relacionados a esse domínio. Em nenhuma delas foi encontrado também como fonte de informação as três bases que iremos utilizar nesse estudo. Consultamos os repositórios em *Protégé Ontology Library*[2], *DAML Ontology Library*[3], *Security Ontology*[4], *An Ontology for Information Security*[5] e *ONKI Library*[6].

No Passo 3 do método, enumeram-se os termos importantes que serão utilizados na ontologia, provenientes de cada uma das bases de conhecimento. Cada um dos termos foi definido como uma classe ou subclasse, conforme Passo 4, imediatamente abaixo da classe obrigatória *owl:thing*. As classes são interpretadas como conjuntos de objetos, sendo a classe *owl:thing* a que representa o conjunto que contém todos os objetos na ontologia, já que todas as classes são subclasses dela.

No Passo 5, foram definidas as propriedades (*slots*, nas versões anteriores do *Protégé*) de cada uma das classes, cuja nomenclatura no *Protégé* atual é *Data Properties*. Seguiu-se para o Passo 6, onde o tipo de cada propriedade foi escolhido. Destaca-se que para a classe *Verification*, sublcasse de *Control*, definiram-se como subclasses cada uma das treze áreas de verificação.

A Figura 5.1 ilustra a hierarquia entre todas as classes criadas, sendo desenvolvida através da aba *OWLViz* do *Protégé*. Adicionalmente, a lista completa das classes, respectivos *Data Properties* e tipos estão na Tabela 5.1.

Nesse ponto, fez-se a definição das propriedades dos objetos da ontologia, com o intuito de descrever as classes e os relacionamento entre elas. Para cada propriedade de objeto, foi criada também a propriedade inversa que possibilita o inter-relacionamento entre as duas classes. No *Protégé*, essa seção é denominada *Object Properties (e.g causes, iscausedby, mitigates, ismitigatedby)*.

No Passo 7, último passo do método 101, definem-se as instâncias, que no *Protégé* são exibidas na seção *Individuals*. As instâncias são criadas para cada uma das classes, utilizando as propriedades de dados (*slots*) definidos nos passos anteriores como atributos. Por exemplo, a classe Risco tem como *slots* designados: *code* e *name*. Para instanciar o primeiro risco 'A1 - Injection' do projeto OWASP Top 10, define-se uma instância *Risk1* com atributos *code*=A1 e *name=Injection*. Da mesma forma, a classe AttackVector tem como *slots* designados: *description*

[2] `http://protegewiki.stanford.edu/wiki/ProtegeOntologyLibrary`
[3] `http://www.daml.org/ontologies/`
[4] `http://www.securityontology.com/`
[5] `http://www.ida.liu.se/~iislab/projects/secont/`
[6] `https://onki.fi/`

Figura 5.1: Hierarquia de classes da ontologia OWASP-CWE

Tabela 5.1: Classes e *slots* da ontologia OWASP-CWE

Fonte	Classe	Propriedades	Tipo
O W A S P T O P 10	Risk	code	string
		name	string
	ThreatAgent	description	string
	AttackVector	description	string
		exploitability	easy / average / difficult
	Weakness	description	string
		prevalence	widespread / common / uncommon
		detectability	easy / average / difficult
	Impact	description	string
		severity	severe / moderate / minor
	Control	description	string
OWASP ASVS	Verification (subclasse)	code	string
		name	string
		number	integer
		level	integer (1-2-3)
		description	string
C W E	CWE	ID	integer
		example	string
		language	string

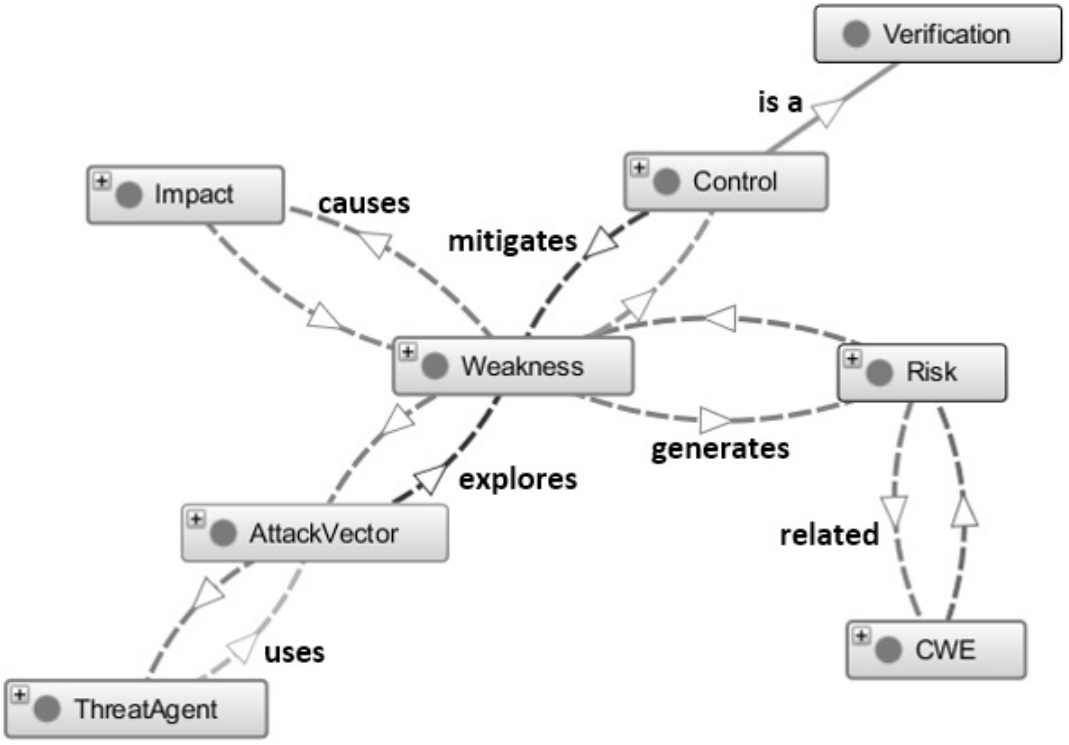

Figura 5.2: Classes e Relacionamentos da Ontologia OWASP-CWE

e *explotability*. Para instanciar o *attackvector* relacionado ao Risco A1, define-se uma instância *AttackVector1* com atributos *description='Attacker sends simple text-based attacks that...'* e *exploitability=easy*.

Após a instanciação de todos os *Individuals* relacionados ao *Risk1* do OWASP Top 10 (*Weakness1, AttackVector1, ThreatAgent1, Control1, Impact1*), cria-se a relação entre eles através das opções de relacionamento criadas anteriormente pela aba *Object Properties* do *Protégé*.

Os relacionamentos são os apresentados na Figura 5.2. Exemplificando para um indivíduo: o *Individual Weakness1 isexploredby AttackVector1, causes Impact1, generates Risk1 and ismitgated by Control1*. Uma ilustração do indivíduo *Weakness1* com todas as suas propriedades definidas (*data properties* e *object properties*) é apresentada na Figura 5.3. Usando esse mesmo critério, criaram-se todos os indivíduos do OWASP Top 10 com todas as suas respectivas relações entre si.

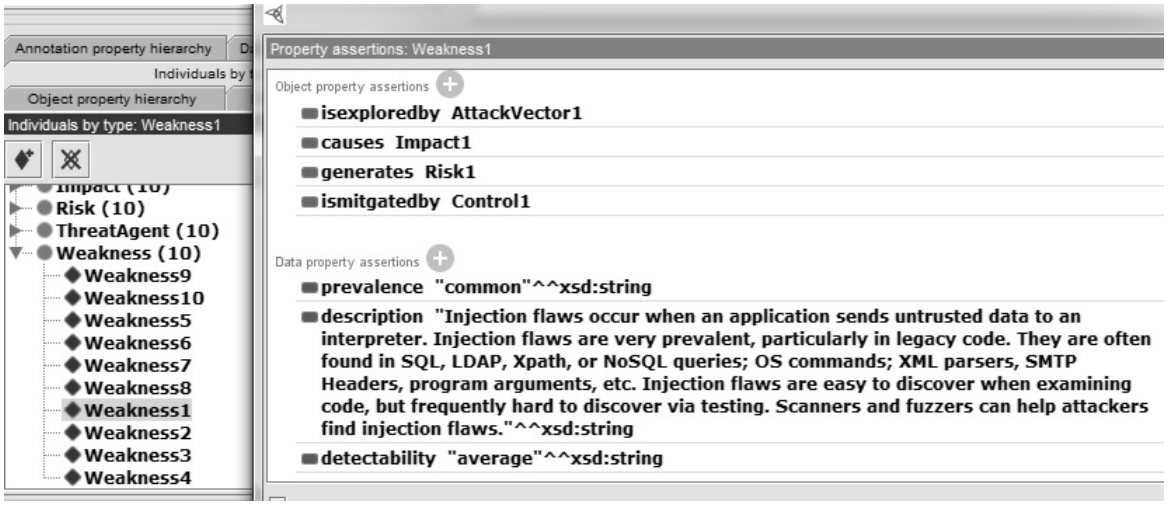

Figura 5.3: Indivíduo *Weakness1* da ontologia OWASP-CWE

Seguindo essa metodologia, instanciaram-se as informações presentes nas três bases de conhecimento de acordo com suas respectivas classes. Para a subclasse *Verification* do OWASP ASVS, as 168 instâncias dos três níveis de maturidade foram criadas. Para a classe CWE, as instâncias relacionadas aos riscos baseados na entrada CWE-ID 928 [7] foram criadas. Nesse caso, os exemplos disponíveis estão distribuídos em códigos das seguintes linguagens: Java (27), PHP (12), C (8), Perl (7), JSP (3), HTML (3), C++ (3), SQL (2) e C# (1), totalizando 66 exemplos.

Por fim, realizaram-se as conexões entre as três bases de conhecimento:

1. Para a relação entre o OWASP Top 10 e o OWASP ASVS, definiram-se os axiomas das instâncias das sublcasses de *Verification* provenientes do OWASP ASVS com os riscos do OWASP Top 10 através do *Object Property 'relatesto'*. Por exemplo, a instância *V2.4* da subclasse *Authentication*, cujo

[7] `http://cwe.mitre.org/data/definitions/928.html`

slot 'description' é *'Verify all authentication controls are enforced on the server side'* e *slot 'level'* é '1', relaciona-se com a instância *Control7* através da propriedade de objeto *'isrelatedto'*. Assim, *V2.4 isrelatedto Control7*. Por sua vez, a instância *Control7 'mitgates' Weakness7* que *'generates' Risk7 = 'Missing Function Level Access Control'*. Ressalta-se que nem todas as instâncias de OWASP ASVS foram necessariamente relacionadas a um dos riscos do OWASP Top 10, isto é, existem instâncias que pertenceriam a outra categoria de risco não listada no Top 10 (do total de 168 instâncias do OWASP ASVS, 126 foram relacionadas). A relação de todos os itens foi criada por consenso entre três profissionais especialistas em segurança de informação e encontra-se completa na ontologia desenvolvida. Para melhor ilustrar, enumeramos a seguir a relação de todas as entradas da categoria *V2 Authentication* do OWASP ASVS com os respectivos riscos do OWASP Top 10:

   - A2 (Quebra de Autenticação e Gerenciamento de Sessão) - V2.1, V2.2, V2.5, V2.6, V2.7, V2.8, V2.9, 2.13, 2.17, 2.18, 2.21, 2.22, 2.23, 2.24, 2.26
   - A5 (Configuração Incorreta de Segurança) - V2.12, 2.19, 2.20, 2.25
   - A6 (Exposição de Dados Sensíveis) - V2.16
   - A7 (Falta de Função para Controle do Nível de Acesso) - V2.4

2. Para a relação entre o OWASP Top 10 e o CWE, utilizou-se a já existente relação direta fornecida pelo CWE, instacinado-se os exemplos de código disponíveis para várias linguagens, relacionando-os também através da propriedade de objeto *'isrelatedto'*. Assim, as instâncias relacionadas ao CWE-77 - *'Improper Neutralization of Special Elements used in a Command'* está diretamente relacionada à instância *Risk1='Injection'*, isto é, *77a 'isrelatedto' Risk1*. Destaca-se que um exemplo pode estar relacionado a mais de um risco, e que todos estão relacionados a pelo menos um dos riscos instanciados.

Após a instanciação e criação dos axiomas que definem os relacionamentos de interesse, a versão final da ontologia foi verificada em relação à sua consistência. Para isso, utilizou-se o *plugin Pellet* (Sirin et al., 2007) na sua versão 2.0, um motor de inferência capaz de checar inconsistências hierárquicas, axiomáticas, de domínio e escopo, dentre outras. Com a confirmação da consistência, a ontologia pode ser utilizada para fornecer informações.

A ontologia completa com todas as classes e instâncias está publicada em *www.marciusmarques.com/owasp/owasp-cwe.owl*, disponível para download e uso.

5.2 Arquitetura

Uma vez que a ontologia OWASP-CWE esteja consistente, é possível, utilizando o formato RDF (*Resource Description Framework*), recuperar informações através do uso de uma poderosa linguagem de consulta - SPARQL, acrônimo recursivo para *SPARQL Protocol and RDF Query Language*. O *Protégé* fornece a

opção de salvar a ontologia em formato RDF, criando um modelo semântico formal através de 'triplas' sujeito-predicado-objeto. Informações detalhadas sobre a linguagem e a utilização com RDF pode ser encontrada em Prud et al. (2008). A combinação de consultas com SPARQL sobre bases de informação ontológicas no formato RDF tem sido cada vez mais utilizadas, ambas sendo atualmente padrões recomendados pela W3C[8] (*World Wide Web Consortium*) (Pérez et al., 2006).

A aplicação das consultas sobre a ontologia foi feita utilizando um servidor SPARQL gratuito - Fuseki[9], disponível dentro do projeto java Jena[10]. O Fuseki permite a execução de consultas a modelos RDF diretamente através do protocolo HTTP, isto é, sobre ontologias RDF publicadas em servidores *web*. Exemplificando, a seguinte consulta retorna todas as triplas existentes no modelo:

```
PREFIX uni:<http://www.marciusmarques.com/owasp/owasp-cwe.owl#>
select * {?x ?y ?z}
```

Uma ilustração da arquitetura desenvolvida pode ser encontrada na Figura 5.4. Os parâmetros para a consulta são definidos através do **Questionário** respondido pelo desenvolvedor, cuja versão atual contempla vinte perguntas, detalhadas no Apêndice A.

Para responder ao **Questionário**, o desenvolvedor utiliza seus conhecimento dos **Requisitos da Aplicação**. O questionário foi desenvolvido dentro da opção de biblioteca de documentos do tipo 'Pesquisa', da solução da Microsoft SharePoint na versão 2010. Cada uma das perguntas, cuja resposta pode ser Sim (*Yes*), Não (*No*) ou N/A (Não Aplicável/*Not Available*), está relacionada a um ou mais riscos do OWASP Top 10, no caso de resposta positiva.

Baseado nas vinte perguntas, o sistema identifica quais são os três maiores riscos (**Risco Info**) associados às características da aplicação. Em seguida, entrega essa informação ao **Apache Jena-Fuseki**, que através da linguagem **SPARQL** consulta a **Ontologia OWASP-CWE**, retornando todas as informações sobre esses riscos no formato **XML**. Por fim, a informação é formatada utilizando o **XML Parser** do próprio navegador, para disponibilizar o **Relatório de Riscos** ao desenvolvedor.

Espera-se que essas informações sejam levadas em consideração no processo de desenvolvimento. Destaca-se que o desenvolvedor pode escolher receber as recomendações em três níveis - básico, intermediário e avançado, cada um deles relacionados ao nível 1, 2 ou 3 de recomendações de verificação do OWASP ASVS. Desenvolvedores iniciantes em atividades relacionadas à segurança devem utilizar os níveis 1 ou 2, enquanto desenvolvedores mais experientes podem utilizar o nível 3.

Os passos para utilização do sistema com suas respectivas tecnologias envolvidas são enumerados abaixo:

1. Preparação do ambiente

[8] `http://www.w3.org`

[9] `http://jena.apache.org/documentation/servingdata/index.html`

[10] `https://jena.apache.org/`

(a) Criação da ontologia (*Protégé*) - conforme passos descritos na Seção 5.1, gerou-se o arquivo *owasp-cwe.owl*, com 21 classes, 13 propriedades de objetos, 11 propriedades de dados, 292 indivíduos e 1562 axiomas.

(b) Publicação da ontologia no formato RDF em servidor *web* (*Protégé*) - realizado através de envio do arquivo gerado para um servidor *web* via FTP.

(c) Execução do servidor RDF/HTTP (Apache Jena - Fuseki) - executado em memória pela porta 3030, integrado ao modelo RDF desenvolvido. O comando de execução da ferramenta é *fuseki-server –mem /ds* e para acesso pelo navegador é suficiente utilizar o endereço `http://localhost:3030/` para execução local.

2. Execução do sistema

(a) Desenvolvedor preenche questionário e nível de maturidade desejado (Microsoft SharePoint)

(b) Definição dos três riscos associados às respostas (Microsoft SharePoint)

(c) Consulta às informações sobre os riscos na ontologia (Apache Jena - Fuseki - SPARQL)

(d) Retorno das informações para conhecimento do desenvolvedor (Microsoft SharePoint)

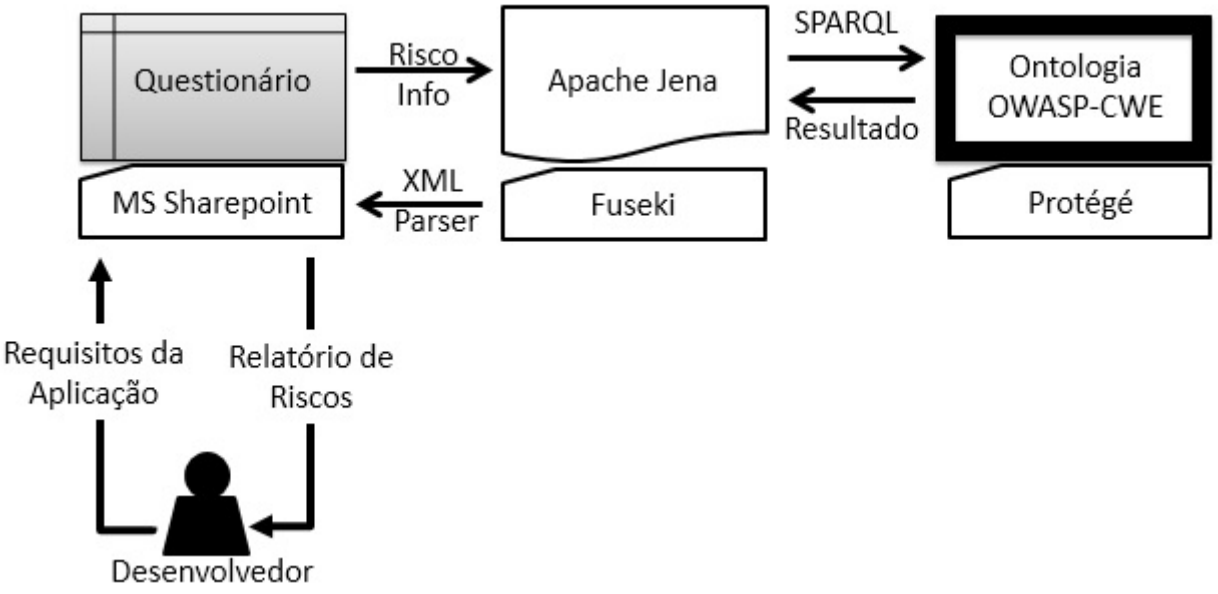

Figura 5.4: Arquitetura de uso da Ontologia OWASP-CWE

Capítulo 6

Estudos de Caso e Resultados

Para avaliar a abordagem proposta, realizaram-se quatro experimentos em casos reais de solicitações de desenvolvimento de aplicações *web* da organização, com os resultados sendo analisados quantitativa e qualitativamente. As aplicações selecionadas possuem a maioria das características encontradas nesse tipo de aplicação de forma constante, como mecanismos de autenticação, comunicação via parâmetros, níveis de privilégios distintos, dentre outras.

6.1 Cenário 1 - *SMS Broadcast*

- Descrição da demanda: desenvolvimento de uma aplicação que possa enviar mensagens de texto para funcionários a partir de uma interface *web*. O sistema deve possibilitar que os funcionários sejam selecionados baseado em uma funcionalidade de filtros. Deve ser possível filtrar por localização e seção de trabalho. Por exemplo, deve ser possível filtrar o envio da mensagem apenas para os funcionários do setor de Finanças do Rio de Janeiro. A aplicação deve ter alta disponibilidade, para ser utilizada em caso de emergências onde se faz necessário enviar comunicações aos funcionários dentro de um limitado número de opções (por exemplo, quando de uma indisponibilidade do sistema de e-mail). A aplicação deve possuir um módulo de cadastro onde qualquer funcionário da organização pode enviar os seus dados e de seus dependentes relacionados. O módulo de envio deve ser restrito a apenas alguns funcionários pré-determinados. A aplicação é considerada de baixa complexidade.

- Tecnologias envolvidas: *backend* Microsoft SQL Server, *frontend* Microsoft SharePoint, linguagem *ASP* Clássico.

- Experimento: a tarefa de desenvolvimento foi designada para quatro desenvolvedores com experiência similar em desenvolvimento de aplicações, mas diferentes experiências em relação a desenvolvimento seguro. As aplicações foram desenvolvidas sob diferentes condições:

  – Os desenvolvedores DA1 e DB1 realizaram o desenvolvimento sem utilizar a ontologia OWASP-CWE. O desenvolvedor DA1, entretanto, participou do curso de 40 horas baseado em OWASP no ano anterior.

  - Os desenvolvedores DA2 e DB2 utilizaram a ontologia OWASP-CWE antes do processo de desenvolvimento, através do preenchimento do questionário para identificação dos maiores riscos associados à aplicação. O desenvolvedor DA2, além disso, participou do curso de 40 horas baseado em OWASP no ano anterior. Ressalta-se que nesse primeiro cenário a ontologia, em sua primeira versão, ainda não possuía informações das bases OWASP ASVS e CWE, sendo composta da informação presente no OWASP Top 10.

- Avaliação: os quatro códigos desenvolvidos - DA1, DB1, DA2 e DB2 - foram submetidos ao processo de CnA da organização, cujo resultado pode ser verificado na Tabela 6.1 e na Figura 6.1.

6.2 Cenário 2 - *After hours*

- Descrição da demanda: determinado grupo de funcionários só podem ter acesso à rede dentro do horário pré-estabelecido de 07:00 da manhã às 07:00 da noite, durante dias úteis. Entretanto, podem solicitar acesso à rede fora desse horário em caso de necessidade, por aplicação *web* automatizada. A aplicação deve permitir que o funcionário seja identificado automaticamente, não sendo possível solicitar acesso *after hours* para terceiros. Após o preenchimento do período e motivo, a solicitação deve ser submetida à autoridade competente, e, em caso de aprovação, deve realizar de forma automática os ajustes no serviço de diretórios para permitir o acesso. A aplicação deve possuir um módulo de cadastro onde qualquer funcionário da organização pode enviar os seus dados, um módulo de aprovação pelo chefe imediato e um módulo de consulta com um resumo dos acessos válidos. A aplicação é considerada de média complexidade.

- Tecnologias envolvidas: *backend* Microsoft SQL Server, *frontend* Microsoft SharePoint, linguagens Visual Basic e PL/SQL por *stored procedures* T-SQL.

- Experimento: a tarefa de desenvolvimento foi designada para quatro desenvolvedores com experiência similar em desenvolvimento de aplicações, mas diferentes experiências em relação à desenvolvimento seguro. As aplicações foram desenvolvidas sob diferentes condições:

  - Os desenvolvedores DC1 e DD1 realizaram o desenvolvimento sem utilizar a ontologia OWASP-CWE. O desenvolvedor DC1, entretanto, participou do curso de 40 horas baseado em OWASP no ano anterior.
  - Os desenvolvedores DC2 e DD2 utilizaram a ontologia OWASP-CWE no nível 1 de maturidade do OWASP ASVS antes do processo de desenvolvimento, através do preenchimento do questionário para identificação dos maiores riscos associados à aplicação. O desenvolvedor DC2, além disso, participou do curso de 40 horas baseado em OWASP no ano anterior.

- Avaliação: os quatro códigos desenvolvidos - DC1, DD1, DC2 e DD2 - foram submetidos ao processo de CnA da organização, cujo resultado pode ser verificado na Tabela 6.2 e na Figura 6.2.

6.3 Cenário 3 - Teste Cruzado

- Descrição da demanda: realizou-se um novo experimento com os desenvolvedores que não possuíam cursos sobre desenvolvimento seguro e não haviam ainda utilizado a ontologia OWASP-CWE, solicitando que desenvolvessem a aplicação que não haviam desenvolvido anteriormente.
- Tecnologias envolvidas: as mesmas citadas anteriormente para cada uma das aplicações.
- Experimento: o desenvolvedor DB1 que havia desenvolvido a aplicação *SMS Broadcast* desenvolveu a aplicação *After Hours*, só que dessa vez utilizando o questionário da ontologia OWASP-CWE. O desenvolvedor DD1 que havia desenvolvido a aplicação *After Hours* desenvolveu a aplicação *SMS Broadcast*, só que dessa vez utilizando o questionário da ontologia OWASP-CWE no nível 1 de maturidade do OWASP ASVS.
- Avaliação: ambos os códigos foram submetidos ao processo de CnA da organização, cujo resultados podem ser verificados nas Tabelas 6.3 e 6.4.

Os resultados dos três primeiros experimentos foram consolidados graficamente para melhor análise quantitativa da pontuação de cada um dos dez processos de desenvolvimento, conforme Figura 6.3.

6.4 Cenário 4 - *iPolvo*

- Descrição da demanda: desenvolver dois módulos *web* de uma aplicação que possibilita o cadastramento de veículos roubados e consulta à bases designadas sobre situação atual de veículos. O primeiro módulo deve possibilitar o cadastro de usuários e veículos no site e o segundo a busca por veículos pela placa. O sistema deve ser disponibilizado no domínio *www.ipolvo.com.br* e não se pode fazer uso de componentes externos em nenhum deles.
- Tecnologias envolvidas: HTML5 e PHP.
- Experimento: contratou-se dois desenvolvedores (DE1 e DE2) com semelhante experiência de desenvolvimento nas linguagens para desenvolverem os mesmo módulos da aplicação em questão, nenhum deles tendo feito cursos formais de segurança da informação. A um dos desenvolvedores (DE2), forneceu-se a ontologia OWASP-CWE para ser utilizada no nível 2 de maturidade do OWASP ASVS.
- Avaliação: ambos os códigos foram analisados qualitativamente para identificação de problemas relacionados à segurança, buscando-se suas respectivas relações com as informações existentes na ontologia.

6.5 Resultados

Durante o processo de CnA da organização, especialistas em segurança avaliam os artefatos produzidos utilizando diversos critérios relacionados à avaliação de aspectos de segurança. Esses profissionais possuem vasta experiência nessa metodologia, com no mínimo cinco anos trabalhando nessa mesma área. O método é uma combinação de revisão de código e testes de penetração, que não podem ser detalhados neste estudo em razão do termo de confidencialidade da organização. Esse tipo de análise é considerado um dos mais eficientes na avaliação de segurança de aplicações segundo Curphey and Arawo (2006). Uma análise detalhada dos benefícios da metodologia pode ser encontrada em Wichers (2010), onde explica-se como ambas as abordagens se complementam. Essa metodologia está alinhada com as atividades de gerenciamento de risco discutidas na Seção 2.3 e também com o método de desenvolvimento de código seguro SDL discutido na Seção 3.1.

6.5.1 Cenário 1

No Cenário 1, observou-se que a aplicação mais vulnerável foi a desenvolvida pelo desenvolvedor DB1 (12,3), que não realizou treinamento formal sobre segurança e também não utilizou a ontologia OWASP-CWE antes do desenvolvimento. Os artefatos produzidos pelos desenvolvedores DA1 (6,4) e DA2 (6,6), apesar de precisarem ser revistos, pois em nenhum deles foi atingida a pontuação mínima exigida (6,0), são os mais seguros dentro desse cenário. Ambos desenvolvedores haviam concluído o curso de segurança baseado no OWASP, destacando-se que o segundo também fez uso da ontologia. Por fim, o desenvolvedor DB2, apenas utilizando a ontologia, alcançou uma pontuação aceitável de 7,6 no desenvolvimento da aplicação. Os resultados estão apresentados na Tabela 6.1 e na Figura 6.1.

Questionados sobre o uso da ontologia em si, os desenvolvedores DA2 e DB2 relataram que foi útil ter respondido o questionário inicial para obter informações sobre riscos associados à aplicação, os quais possivelmente eles não teriam considerado com o nível de detalhamento da informação apresentada, ou mesmo não teriam avaliado inicialmente.

Para o desenvolvedor DB2, o uso da ontologia foi o seu primeiro contato com os conceitos existentes na proposta do projeto OWASP. A partir da informação inicial disponibilizada pelo sistema e durante o processo de desenvolvimento, ele buscou mais informações sobre segurança no site do projeto, tendo se interessado por outros projetos dessa iniciativa.

Tabela 6.1: Cenário e Resultados *SMS Broadcast*

Desenvolvedor	DA1	DB1	DA2	DB2
Experiência em desenvolvimento	Alta	Alta	Alta	Alta
Conscientização segurança	Alta	Baixa	Alta	Baixa
Uso da ontologia OWASP-CWE	Não	Não	Sim	Sim
Pontuação processo de CnA	6,4	12,3	6,6	7,6

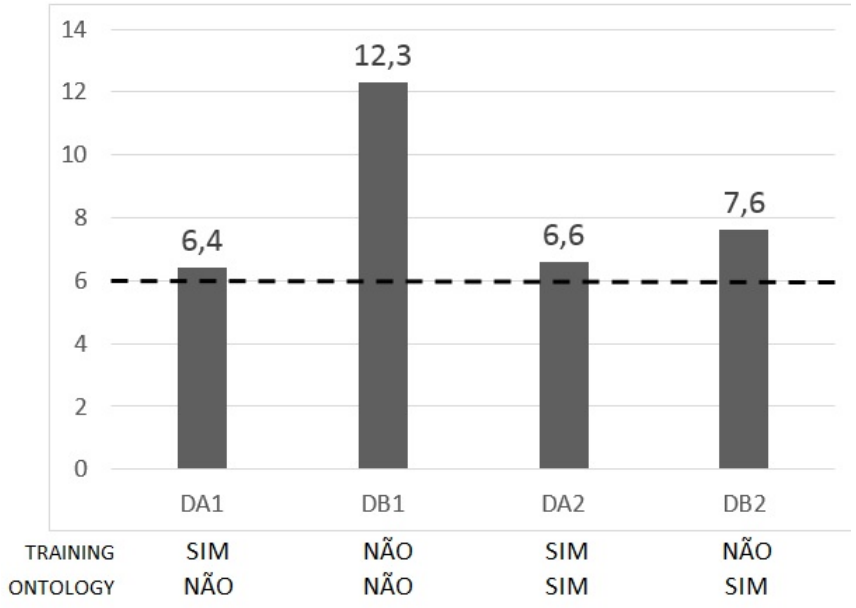

Figura 6.1: Resultados *SMS Broadcast*

6.5.2 Cenário 2

Para o segundo cenário, novamente a aplicação mais problemática em relação à segurança foi a desenvolvida pelo desenvolvedor que não possuía treinamento e nem fez uso da ontologia, DD1 (11,5). As aplicações mais seguras na primeira análise foram as desenvolvidas por DC1 (7,5) e DD2 (7,3), o primeiro tendo apenas realizado o treinamento formal e o segundo tendo apenas utilizado a ontologia. Uma pontuação intermediária foi obtida pelo desenvolvedor DC2 (8,5) que além do treinamento fez também o uso da ontologia. Essa variação de resultados reforça a ideia da subjetividade e existência de variáveis de difícil controle para análises quantitativas, conforme será discutido no Capítulo 7. Por conta desse resultado optou-se por incluir análises qualitativas no quarto experimento. Os resultados estão apresentados na Tabela 6.2 e na Figura 6.2.

Ressalta-se que também ambos os desenvolvedores DC2 e DD2 fizeram considerações positivas sobre o uso da ontologia OWASP-CWE ao serem questionados após a utilização, com destaque para o pouco tempo necessário para obter conhecimento de forma direcionada ao cenário em que se irá trabalhar.

Tabela 6.2: Cenário e Resultados *After Hours*

Desenvolvedor	DC1	DD1	DC2	DD2
Experiência em desenvolvimento	Alta	Alta	Alta	Alta
Conscientização segurança	Alta	Baixa	Alta	Baixa
Uso da ontologia OWASP-CWE	Não	Não	Sim	Sim
Pontuação processo de CnA	7,5	11,5	8,5	7,3

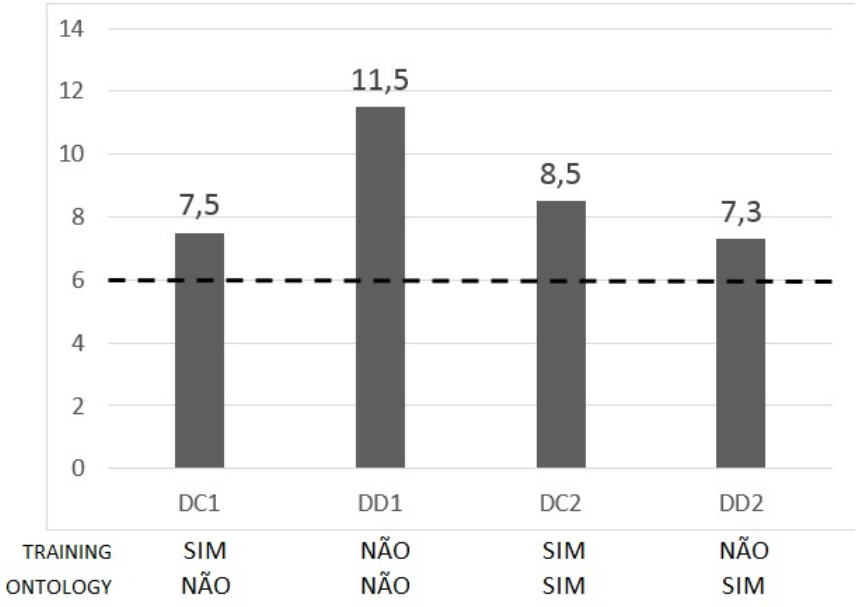

Figura 6.2: Resultados *After Hours*

6.5.3 Cenário 3

No Cenário 3, procurou-se avaliar o aumento da preocupação com a segurança da aplicação através de um 'teste cruzado', onde desenvolvedores que não haviam realizado o curso formal de segurança estariam utilizando a ontologia pela primeira vez. O desenvolvedor DB1 obteve pontuação de 8,9 ao desenvolver artefatos da aplicação *After Hours* e o desenvolvedor DD1 obteve pontuação de 7,1 ao desenvolver artefatos da aplicação *SMS Broadcast*. Apesar de não ser possível uma comparação direta por serem aplicações e tecnologias distintas, observa-se considerável melhora ante os resultados anteriores de 12,3 e 11,5, respectivamente, conforme apresentados nas Tabelas 6.3 e 6.4. Os resultados consolidados dos três experimentos encontram-se na Figura 6.3.

Tabela 6.3: Cenário e Resultados Teste Cruzado 1

Desenvolvedor	DB1
Experiência em desenvolvimento	Alta
Conscientização segurança	Baixa
Uso da ontologia OWASP-CWE	Sim
Pontuação processo de CnA	8,9

Tabela 6.4: Cenário e Resultados Teste Cruzado 2

Desenvolvedor	DD1
Experiência em desenvolvimento	Alta
Conscientização segurança	Baixa
Uso da ontologia OWASP-CWE	Sim
Pontuação processo de CnA	7,1

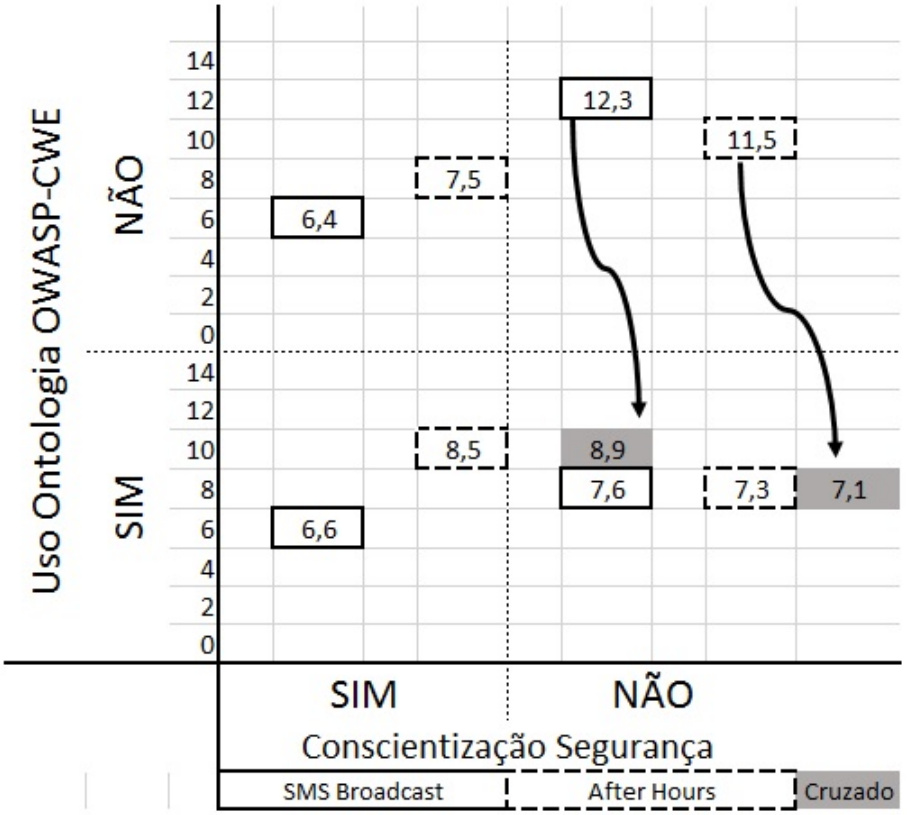

Figura 6.3: Resultado consolidados dos experimentos 1, 2 e 3

Outra interessante comparação pode ser feita nos testes cruzados entre os desenvolvedores DB1 que teve pontuação de 8,9 com o desenvolvedor DD2 que teve pontuação de 7,3 para o sistema *After Hours* utilizando apenas a ontologia. Da mesma forma, o desenvolvedor DD1 obteve pontuação de 7,1 enquanto o desenvolvedor DB2 obteve pontuação de 7,6 para o sistema *SMS Broadcast*, salientando que ambos utilizaran apenas a ontologia anteriormente ao processo de desenvolvimento.

6.5.4 Cenário 4

Para o Cenário 4, três especialistas de segurança revisaram os códigos produzidos pelos desenvolvedores DE1 e DE2. No total, 7 recomendações foram feitas em relação à segurança para o código produzido por DE2, enquanto mais 5 outras foram feitas para o desenvolvedor DE1, totalizando 12 recomendações para este último. Listamos abaixo alguns dos itens relatados somente em DE1, que não fez uso da ontologia OWASP-CWE, com a respectiva entrada relacionada na ontologia que poderia ter ajudado na mitigação da vulnerabilidade.

- Falta de validação de entrada de dados fornecidos pelo usuário - Em um dos requisitos do módulo de registro, é possível enviar uma imagem no caso de um veículo ser localizado. No código DE1, todas as entradas de formulário foram verificadas, com exceção da entrada que faz o envio da imagem que

fica associada ao carro cadastrado. Como é um parâmetro manipulável pelo usuário, ele pode ser utilizado para um ataque do tipo 'injeção de comando', que é descrito na entrada 78a da ontologia para a construção de um caminho de diretório baseado no nome do usuário, estando relacionado ao Risco A1 - Injeção.

- Redirecionamento indevido de URL - Observou-se uma URL interna cujo parâmetro era definido pelo método GET do PHP. Apesar de estar dentro de uma função interna, é possível verificar a vulnerabilidade através de debugadores de navegadores com facilidade. Se utilizada com o parâmetro indevido, a navegação pode ser direcionada para uma localização indevida fora do sistema. Exemplo semelhante pode ser encontrado na entrada 601a da ontologia, estando relacionado ao Risco A10 - Redirecionamentos e Encaminhamentos Inválidos.

- Falta de validação de credenciais para nível de controle de acesso - A validação para acesso à uma das funcionalidade (reportar roubo) foi implementada de forma indevida, sem que a variável que verifica a autorização fosse definida inicialmente. Apesar de não ter nenhum exemplo de código relacionado, a vulnerabilidade está relacionado ao Risco A7 - Falta de Função para Controle do Nível de Acesso. Uma discussão com maiores detalhes sobre essa questão pode ser encontada em *http://pageconfig.com/post/insecure-php-constants-and-variables*.

O desenvolvedor DE2 destacou a facilidade de uso da ontologia, apontando algum itens do relatório recebido que foram levados em consideração nas etapas de desenvolvimento. Relatou, entretanto, que os exemplos poderiam ser filtrados por linguagem, já que na aplicação em questão, a linguagem estabelecida era PHP.

Acrescenta-se que os desenvolvedores relataram que o uso da ontologia no processo de desenvolvimento foi positivo e eficiente. Positivo porque admitem que riscos que não haviam considerados inicialmente foram mitigados, e eficiente porque não tiveram que passar várias horas em processos de pesquisa ou cursos de longa duração para obter informação sobre as vulnerabilidades, isto é, receberam a informação de forma estruturada ou, nas palavras deles, "limpa e suficiente". O uso da ontologia também despertou o interesse em buscar mais informações sobre o tema, sobretudo relacionadas ao projeto OWASP.

Entretanto, destaca-se que nenhuma das aplicações analisadas conseguiu passar pelo processo de avaliação da organização na primeira tentativa, já que não atingiram a pontuação de 6,0 requerida. Tal fato enfatiza a importância das atividades de avaliação de risco no processo de desenvolvimento de aplicações, reforçando que essas atividades sejam feitas em todas as etapas do desenvolvimento.

Os resultados nesses cenários estão sujeitos a fatores subjetivos de difícil controle. As aplicações foram desenvolvidas por diferentes pessoas, em diferentes dias e momentos distintos. Ainda que tenha se buscado uniformidade em aspectos que possam influenciar esses resultados, como por exemplo em relação ao conhecimento da linguagem e do próprio idioma inglês, uma maior quantidade

de experimento seria necessária para minimizar essa variável, que ainda assim não poderia ser ignorada. Tal fato pode ter contribuído para, por exemplo, o resultado de 8,5 alcançado pelo desenvolvedor DC2 no desenvolvimento de uma das aplicações.

Buscando-se outros parâmetros, realizou-se o experimento do Cenário 4 fora da organização, de forma que os resultados pudessem ser divulgados para enriquecer o estudo e as discussões. Na análise do Cenário 4 realizada, destaca-se que vulnerabilidades poderiam ter sido mitigadas através do uso da ontologia, já que foram identificadas cinco recomendações de segurança a mais onde não houve o uso desta. Todas essas recomendações estavam presentes através de exemplos na ontologia desenvolvida.

Capítulo 7

Conclusões e Trabalhos Futuros

Com a crescente necessidade de serem eficientes e eficazes para se manterem competitivas, as organizações buscam constantemente se atualizar em relação aos seus processos tecnológicos, especialmente os relacionados a sistemas de informação. Pelas características presentes em aplicações do tipo *web*, essas tem se tornado a solução preferida para gerir os sistemas, sendo utilizadas com frequência cada vez maior para oferecer aos clientes uma abordagem moderna para realização de negócios. Entretanto, essas mesmas características possuem um custo relacionado à garantia da SegInfo, que nem sempre está na lista de prioridades da maioria das organizações. E isso ocorre não por opção, mas porque atividades de SegInfo são caras e complexas.

Um dos pontos chaves para o seu correto funcionamento, e também um dos pilares do projeto OWASP utilizado nesse estudo, é de que atividades de segurança devam existir em todas as fases do desenvolvimento de aplicações. Em consonância com essa proposta, nesse trabalho focou-se no processo de arquitetura de aplicações, buscando inserir atividades relacionados à segurança no momento que antecede a escrita do código em si. Como citado anteriormente, não são comuns atividades desse tipo nessa fase do projeto, apesar de que cada vez mais iniciativas (como o SDL) tem reforçado a importância de se falar em segurança nesse momento do ciclo do projeto, para que se possa desenvolver aplicações mais seguras.

Os desenvolvedores possuem um papel crucial na linha de defesa de aplicações *web*, já que a maioria das vulnerabilidades exploradas atualmente são consequência de atividades de desenvolvimento que não foram executadas com as devidas preocupações de segurança necessárias. "Os problemas de segurança deste domínio (*web*) não estão sendo adequadamente considerados durante o processo de desenvolvimento, tanto por ignorância como pela pressão causada por cronogramas de entrega apertados" (Uto and Melo, 2009).

Com base nesse cenário, o presente estudo propôs uma abordagem ontológica que auxilia na produção de aplicações *web* mais seguras. A proposta se baseia no conhecimento que o desenvolvedor possui sobre a aplicação que ele irá desenvolver para fornecer informações sobre os riscos relacionados à essa aplicação. O objetivo maior é identificar as boas práticas de segurança que precisam ser aplicadas ao processo de desenvolvimento para mitigar os riscos. O fardo de longas horas de treinamento e pesquisa são abstraídos do desenvolvedor, o qual pode,

utilizando somente a ontologia, obter as informações que necessita, a qual inclui recomendações e exemplos de código seguro. O desenvolvedor pode escolher dentre três níveis de profundidade de conhecimento, sendo utilizável portanto por profissionais com diferentes níveis de experiência e conhecimento.

A proposta foi testada em quatro cenários reais de desenvolvimento com diferentes aplicações e linguagens, com níveis de conhecimento diversificado dos desenvolvedores, utilizando-se diferentes metodologias de análise. Na análise dos Cenários 1, 2 e 3, contatou-se que o uso da ontologia OWASP-CWE foi útil para produzir aplicações *web* mais seguras. Desenvolvedores que utilizaram a ontologia obtiveram pontuações comparáveis a desenvolvedores que realizaram cursos formais de segurança de acordo com o processo de avaliação de artefatos (CnA) da organização. Esses mesmos desenvolvedores também obtiveram melhores resultados que outros desenvolvedores que não realizaram qualquer curso e não utilizaram a ontologia proposta.

Por fim, destaca-se que o modelo proposto buscou durante todas as suas etapas:

- Abordagem direcionada, para que não houvesse a pretensão ou necessidade de que o desenvolvedor se torne um especialista de segurança;
- Extensibilidade ontológica, para que a informação modelada possa ser combinada ou estendida para outras necessidades identificadas.

Como especialista em segurança, a consolidação de diferentes fontes de conhecimento sobre vulnerabilidades foi de extrema utilidade para proporcionar uma visão global de como essas bases se relacionam, e como se pode extrair a melhor informação de cada uma delas. A modelagem de uma ferramenta que possa fornecer a um desenvolvedor uma informação que seja, ao mesmo tempo, suficiente e acurada, foi um grande desafio, dada a dimensão incalculável de conceitos relacionados à SegInfo. Resumidamente, considera-se a ontologia desenvolvida e disponibilizada em *www.marciusmarques.com/owasp* como a principal contribuição do presente estudo. Ressalta-se ainda que o presente estudo foi publicado e apresentado no SBSeg Marques and Ralha (2014).

Como trabalho futuro, sugere-se o uso de outras ferramentas de inferência que possam consultar a base de conhecimento definida, bem como mais experimentos para aumentar a massa de dados e proporcionar outras análises minimizando a subjetividade. A ontologia pode ser utilizada para fornecer orientação à outros tipos de questionamentos facilmente realizados com o uso do SPARQL. Por exemplo, pode-se questionar "Quais exemplos de código relacionados à linguagem PHP que existem para o Risco A1 - Injeção?". Adicionalmente, o questionário desenvolvido pode evoluir para uma interface adaptativa que, baseada na resposta recebida, conduza o desenvolvedor à outras perguntas mais pertinentes à aplicação que está para ser produzida.

O aspecto ontológico possibilita também que o modelo seja expandido ou modificado para utilizar outras bases de conhecimento por exemplo a norma ISO27001, sendo esta um requisito cada vez mais exigido nas organizações. Para maior aproveitamento do potencial ontológico, sugere-se ainda a exploração da ontologia para processos de inferência de forma automatizada. Por exemplo, a

partir de um arquivo de casos de uso ou regras de negócio, a ontologia pode ser empregada para identificar os riscos associados à aplicação, sem que um questionário precise ser respondido.

Sugere-se também a inclusão de aspectos mais gerais na ontologia de forma que possa ser utilizada como ferramenta para prover informações em um nível mais alto de tomada de decisão da organização, como por exemplo através da integração com o projeto OWASP SAMM[1] (*Software Assurance Maturity Model*). Esse modelo pode ser aplicado na definição ou avaliação de políticas de segurança, em completo alinhamento com os outros projetos do OWASP, inclusive os utilizados na ontologia OWASP-CWE.

[1] `https://www.owasp.org/index.php/Category:SoftwareAssuranceMaturity_Model`

Referências Bibliográficas

Almeida, M. B. (2007). Aplicação de ontologias em segurança da informação. *Diretoria da Prodemge*.

Almeida, M. B. and Bax, M. P. (2003). Uma visão geral sobre ontologias: pesquisa sobre definições, tipos, aplicações, métodos de avaliação e de construção. *Ciência da Informação, Brasília*, 32(3):7–20.

Almeida, M. B., Souza, R. R., and Coelho, K. C. (2010). Uma proposta de ontologia de domínio para segurança da informação em organizações: descrição do estágio terminológico. *Informação & Sociedade: Estudos*, 20(1).

Andrews, M. (2006). Guest editor's introduction: The state of web security. *IEEE Security & Privacy*, 4(4):0014–15.

Bai, X. and Zhou, X. (2011). Development of ontology-based information system using formal concept analysis and association rules. In *Advances in Computer Science, Intelligent System and Environment*, pages 121–126. Springer.

Bernaras, A., Laresgoiti, I., and Corera, J. (1996). Building and reusing ontologies for electrical network applications'. In *ECAI*, pages 298–302. PITMAN.

Byres, E. (2008). Defense in depth. *Control Engineering Asia June 2008*.

Coelho, F. E., Araújo, L. G., and Bezzera, E. K. (2011). *Gestão da Segurança da informação NBR27001 e NBR 27002*. Escola Superior de Redes RNP.

Curphey, M. and Arawo, R. (2006). Web application security assessment tools. *Security & Privacy, IEEE*, 4(4):32–41.

CWE (2014). Cwe-928: Weaknesses in owasp top ten (2013). `http://cwe.mitre.org/data/graphs/928.html`. [Online; accessado 23-Julho-2014].

da Silva, P. F., Otte, H., Todesco, J. L., and AO, F. (2011). Uma ontologia para gestão de segurança da informação. In *Joint IV Seminar on Ontology Research in Brazil*, page 141. Citeseer.

de Araujo, M. (2003). *Educação à distância ea WEB Semântica: modelagem ontológica de materiais e objetos de aprendizagem para a plataforma COL*. PhD thesis, Universidade de São Paulo.

de Azevedo, R., Almeida, M., and Carvalho Filho, E. (2007). Uma ontologia genérica de segurança aplicada a gestão de processos de negócios. XIII Brazilian Symposium on Multimedia and the Web. I Brazilian Workshop on Business Process Management (WBPM).

de Holanda, M. T. and Fernandes, J. H. C. (2009). Segurança no desenvolvimento de aplicações. *Gestão da Segurança da Informação e Comunicações - CEGSIC2009-2011*.

Dhillon, G. and Backhouse, J. (2000). Technical opinion: Information system security management in the new millennium. *Communications of the ACM*, 43(7):125–128.

Fenz, S. and Ekelhart, A. (2009). Formalizing information security knowledge. In *Proceedings of the 4th international Symposium on information, Computer, and Communications Security*, pages 183–194. ACM.

Gasevic, D., Djuric, D., Devedzic, V., and Selic, B. (2006). *Model driven architecture and ontology development*, volume 1. Springer.

Gordon, L. A. and Loeb, M. P. (2002). The economics of information security investment. *ACM Transactions on Information and System Security (TISSEC)*, 5(4):438–457.

Gregoire, J., Buyens, K., Win, B. D., Scandariato, R., and Joosen, W. (2007). On the secure software development process: Clasp and sdl compared. In *Proceedings of the Third International Workshop on Software Engineering for Secure Systems*, page 1. IEEE Computer Society.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220.

Grüninger, M. and Fox, M. S. (1995). Methodology for the design and evaluation of ontologies.

Guarino, N. (1998). *Formal ontology in information systems: Proceedings of the first international conference (FOIS'98), June 6-8, Trento, Italy*, volume 46. IOS press.

Guarino, N. and Welty, C. (2002). Evaluating ontological decisions with ontoclean. *Communications of the ACM*, 45(2):61–65.

Haav, H.-M. and Lubi, T.-L. (2001). A survey of concept-based information retrieval tools on the web. In *Proceedings of the 5th East-European Conference ADBIS*, volume 2, pages 29–41.

Hernan, S., Lambert, S., Ostwald, T., and Shostack, A. (2006). Threat modeling-uncover security design flaws using the stride approach. *MSDN Magazine-Louisville*, pages 68–75.

Higgins, S. (2009). Information security management: The iso 27000 (iso 27k) series.

Jasper, R., Uschold, M., et al. (1999). A framework for understanding and classifying ontology applications. Proceedings of the IJCAI-99 Ontology Workshop.

Kawano, V. J. (2009). Desenvolvimento de uma ontologia para gerenciamento de projetos. *Monograph Degree in Computing. Exact Science Institute. Department of Computer Science. University of Brasilia, Distrito Federal*.

Kerr, D. (2013). Cyberattacks account. `http://www.pwc.com/en_US/us/increasing-it-effectiveness/publications/assets/us-state-of-cybercrime.pdf`. [Online; accessado 10-Junho-2014].

Klöti, R. (2013). Openflow: A security analysis. *Proc. Wkshp on Secure Network Protocols (NPSec). IEEE*.

Lacey, D. (2011). *Managing the Human Factor in Information Security: How to win over staff and influence business managers*. John Wiley & Sons.

Laszka, A., Johnson, B., Schöttle, P., Grossklags, J., and Böhme, R. (2013). Managing the weakest link. In *Computer Security–ESORICS 2013*, pages 273–290. Springer.

Lenat, D. B. and Guha, R. V. (1989). *Building large knowledge-based systems; representation and inference in the Cyc Project*. Addison-Wesley Longman Publishing Co., Inc.

Lipner, S. (2004). The trustworthy computing security development lifecycle. In *Computer Security Applications Conference, 2004. 20th Annual*, pages 2–13. IEEE.

López, M. F., Gómez-Pérez, A., Sierra, J. P., and Sierra, A. P. (1999). Building a chemical ontology using methontology and the ontology design environment. *IEEE intelligent Systems*, 14(1):37–46.

Lorens, E. M. (2009). Aspectos normativos da segurança da informação: um modelo de cadeia de regulamentação.

Marques, M. M. and Ralha, C. G. (2014). An ontological approach to mitigate risk in web applications. *XIV Simpósio Brasileiro em Segurança da Informação e Sistemas Computacionais - SBSEG*.

Martimiano, L. A. and Moreira, E. (2005). Using ontologies to assist security management. In *Proceedings of the 8th International Protégé Conference*.

Martins, A. B. and Santos, C. A. S. (2005). Uma metodologia para implantação de um sistema de gestão de segurança da informação. *Journal of Information Systems and Technology Management*, 2(2):121–136.

McCumber, J. (2004). *Assessing and managing security risk in IT systems: A structured methodology*. CRC Press.

Microsoft (2010). Security development lifecycle. http://www.microsoft.com/security/sdl/default.aspx. [Online; accessado 30-Julho-2014].

MITRE (2010). About cwe. http://cwe.mitre.org/about/index.html. [Online; accessado 19-Julho-2014].

MITRE/SANS (2011). Top 25. http://cwe.mitre.org/top25/index.html. [Online; accessado 20-Julho-2014].

Mizoguchi, R., Vanwelkenhuysen, J., and Ikeda, M. (1995). Task ontology for reuse of problem solving knowledge. *Towards Very Large Knowledge Bases: Knowledge Building & Knowledge Sharing*, pages 46–59.

Novello, T. C. (2002). Ontologias, sistemas baseados em conhecimento e modelos de banco de dados. *Universidade Federal do Rio Grande do Sul*.

Noy, N. and McGuinness, D. L. (2001). Ontology development 101. *Knowledge Systems Laboratory, Stanford University*.

NSA (2007). Defense in depth: A practical strategy for achieving information assurance in todays highly networked environments. http://www.nsa.gov/ia/_files/support/defenseindepth.pdf. [Online; accessado 20-Julho-2014].

Obrst, L., Chase, P., and Markeloff, R. (2012). Developing an ontology of the cyber security domain. In *STIDS*, pages 49–56.

OWASP (2011a). About owasp. https://www.owasp.org/index.php/About_OWASP. [Online; accessado 13-Julho-2014].

OWASP (2011b). Owasp clasp project. https://www.owasp.org/index.php/Category:OWASP_CLASP_Project. [Online; accessado 30-Julho-2014].

OWASP (2013). Owasp top 10. https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project. [Online; accessado 13-Julho-2014].

OWASP (2014). Owasp asvs. https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project. [Online; accessado 14-Agosto-2014].

Parker, D. B. (2002). Toward a new framework for information security. *FLY*, page 501.

Peltier, T. R. (2013). *Information Security Policies, Procedures, and Standards: guidelines for effective information security management*. CRC Press.

Pérez, J., Arenas, M., and Gutierrez, C. (2006). Semantics and complexity of sparql. In *The Semantic Web-ISWC 2006*, pages 30–43. Springer.

Pinto, H. S. and Martins, J. P. (2004). Ontologies: How can they be built? *Knowledge and Information Systems*, 6(4):441–464.

Pressman, R. S. (2011). *Engenharia de software*. McGraw Hill Brasil.

PRNewswire (2005). Only 10%. http://www.prnewswire.com/news-releases/only-10-of-web-applications-are-secured-against-common-hacking-techniques-58703902.html. [Online; accessado 10-Junho-2014].

Prud, E., Seaborne, A., et al. (2008). Sparql query language for rdf. *W3C recommendation*, 15.

PWC (2013). Key findings. http://www.pwc.com/en_US/us/increasing-it-effectiveness/publications/assets/us-state-of-cybercrime.pdf. [Online; accessado 10-Junho-2014].

PWC (2014). Uma defesa ultrapassada. http://www.pwc.com.br/pt/publicacoes/servicos/consultoria-negocios/pesquisa-global-seguranca-informacao-14.jhtml. [Online; accessado 10-Outubro-2014].

Raskin, V., Hempelmann, C. F., Triezenberg, K. E., and Nirenburg, S. (2001). Ontology in information security: a useful theoretical foundation and methodological tool. In *Proceedings of the 2001 workshop on New security paradigms*, pages 53–59. ACM.

Razzaq, A., Ahmed, H., Hur, A., and Haider, N. (2009). Ontology based application level intrusion detection system by using bayesian filter. In *Computer, Control and Communication, 2009. IC4 2009. 2nd International Conference on*, pages 1–6. IEEE.

Razzaq, A., Anwar, Z., Ahmad, H. F., Latif, K., and Munir, F. (2014). Ontology for attack detection: An intelligent approach to web application security. *computers & security*, 45:124–146.

Razzaq, A., Hur, A., Masood, M., Latif, K., Ahmad, H. F., and Takahashi, H. (2011). Foundation of semantic rule engine to protect web application attacks. In *Autonomous Decentralized Systems (ISADS), 2011 10th International Symposium on*, pages 95–102. IEEE.

Rosa, T. M., Santin, A. O., and Malucelli, A. (2009). Uma ontologia para mitigar xml injection. *XI Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg), p. 1-14.*

Sasse, M. A., Brostoff, S., and Weirich, D. (2001). Transforming the weakest link a human computer interaction approach to usable and effective security. *BT technology journal*, 19(3):122–131.

Sêmola, M. et al. (2003). *Gestão da Segurança da informação*, volume 1. Elsevier Brasil.

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., and Katz, Y. (2007). Pellet: A practical owl-dl reasoner. *Web Semantics: science, services and agents on the World Wide Web*, 5(2):51–53.

Sommerville, I., Melnikoff, S. S. S., Arakaki, R., and de Andrade Barbosa, E. (2011). *Engenharia de software*, volume 9. Addison Wesley.

Staab, S., Studer, R., Schnurr, H.-P., and Sure, Y. (2001). Knowledge processes and ontologies. *IEEE Intelligent systems*, 16(1):26–34.

Stytz, M. R. (2004). Considering defense in depth for software applications. *Security & Privacy, IEEE*, 2(1):72–75.

Swartout, B., Patil, R., Knight, K., and Russ, T. (1996). Toward distributed use of large-scale ontologies. In *Proc. of the Tenth Workshop on Knowledge Acquisition for Knowledge-Based Systems*, pages 138–148.

Uschold, M. and Gruninger, M. (1996). Ontologies: Principles, methods and applications. *The knowledge engineering review*, 11(02):93–136.

Uschold, M. and King, M. (1995). *Towards a methodology for building ontologies*, volume 1. Citeseer.

Uto, N. and Melo, S. (2009). Vulnerabilidades em aplicações web e mecanismos de proteção. *Minicursos SBSeg*.

Van Heijst, G., Schreiber, A. T., and Wielinga, B. J. (1997). Using explicit ontologies in kbs development. *International journal of human-computer studies*, 46(2):183–292.

Viega, J. (2005). Building security requirements with clasp. *ACM SIGSOFT Software Engineering Notes*, 30(4):1–7.

Vorobiev, A. and Han, J. (2006). Security attack ontology for web services. In *Semantics, Knowledge and Grid, 2006. SKG'06. Second International Conference on*, pages 42–42. IEEE.

Vrancianu, M. and Popa, L. A. (2010). Considerations regarding the security and protection of e-banking services consumers and interests. *The Amfiteatru Economic Journal*, 12(28):388–403.

Weske, M. (2007). *Concepts, Languages, Architectures*, volume 14. Springer.

Whitman, M. and Mattord, H. (2011). *Principles of information security*. Cengage Learning.

Wichers, D. (2010). The strengths of combining code review with application penetration testing. `https://www.owasp.org/index.php/TheStrengthsofCombiningCodeReviewwithApplicationPenetrationTesting`. [Online; accessado 01-Setembro-2014].

Wiesmann, A., van der Stock, A., Curphey, M., and Stirbei, R. (2005). A guide to building secure web applications and web services. *The Open Web Application Security Project*.

Yadav, R., Rawal, S. K., and Goyal, A. (2014). Web application security. *International Journal of Computer Science and Mobile Computing*, 3:3494–355.

Ye, J., Coyle, L., Dobson, S., and Nixon, P. (2007). Ontology-based models in pervasive computing systems. *The Knowledge Engineering Review*, 22(04):315–347.

Apêndice A

Questionário do Desenvolvedor

1. System use queries in databases by means of technologies like SQL, LDAP, Xpath or NoSQL
2. SMTP protocol is used in at least one of the system's features
3. System contains information in transit coded in XML language
4. User require to authenticate with a login and password to access the system
5. Authentication credentials are stored in a system database
6. User can recover login information by using 'forgot password/forgot login' feature
7. Requirements demand that users can recover password by answering secret questions
8. User can click a 'remember me' box so they don't have to re-authenticate
9. User system access relies on session controle using timeout and logout features
10. URLs are used to send information from one web page to another
11. Web pages use script technology to validate forms or control behaviour (vbscript, javascript,...)
12. Users are allowed to interact with system trough tools like comments, foruns or surveys
13. AJAX is used to dinamically update webpages
14. There are different levels of permissions within the system
15. There is sensitivie data (PII) stored in the system's database
16. Data transmission is sent in clear text at some point of the system
17. Encryption alghoritms are used to store data in the system

18. The system relies on the use of browser cookies for users to navigate
19. User is redirect to diferent pages according to system's normal flow
20. System uses builtin frameworks or libraries from vendor or open-source