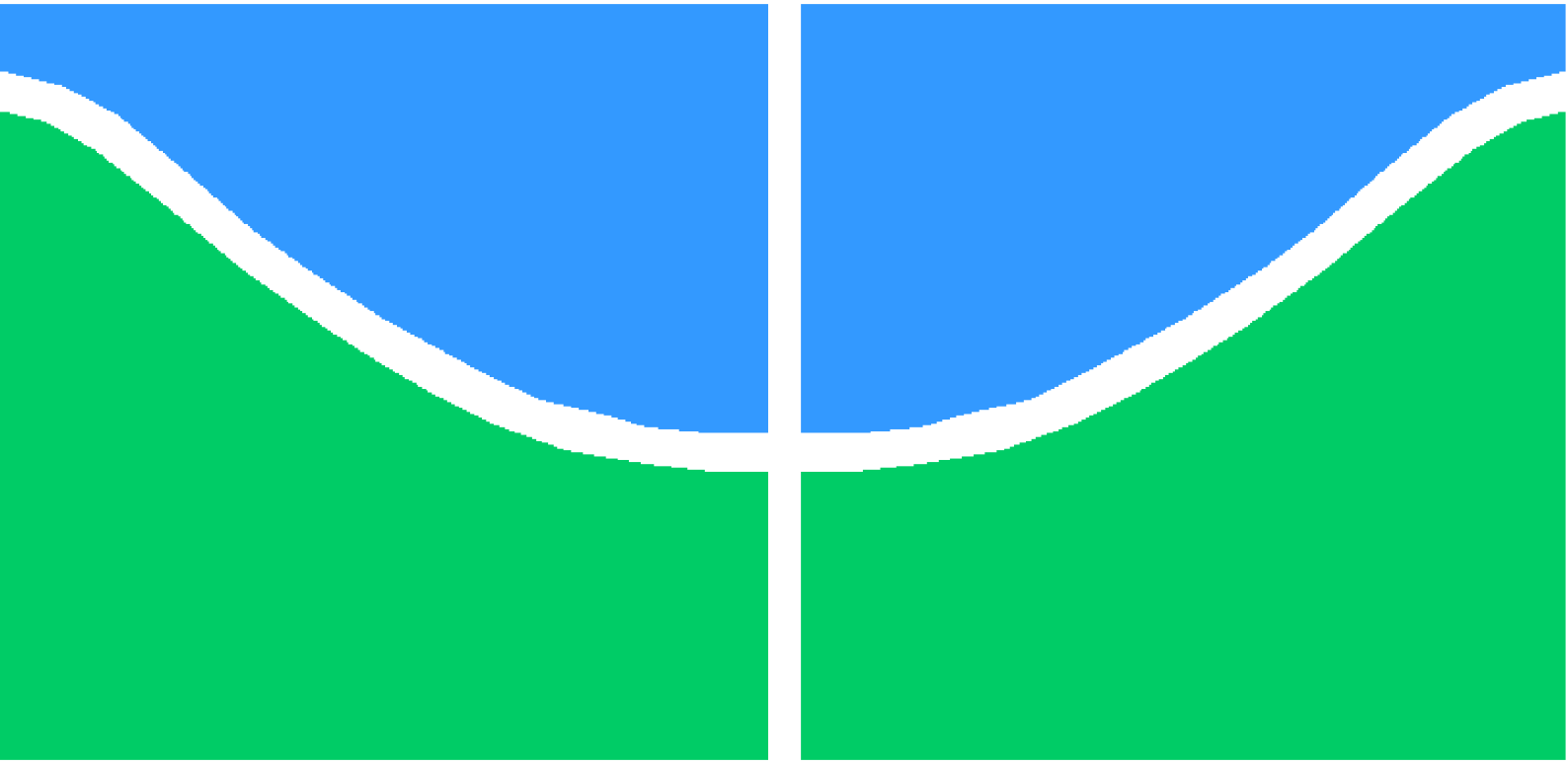

Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Bacharel em Engenharia de Software

Identificação das lacunas ferramental para dar suporte ao programa de segurança de software

Autor: Matheus da Silva Freire
Orientador: Prof. Drº Fabricio Braz

Brasília, DF
2013

Matheus da Silva Freire

Identificação das lacunas ferramental para dar suporte ao programa de segurança de software

Monografia submetida ao curso de graduação em Bacharel em Engenharia de Softwareda Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Bacharel em Engenharia de Software.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. Drº Fabricio Braz

Brasília, DF

2013

Matheus da Silva Freire

Identificação das lacunas ferramental para dar suporte ao programa de segurança de software/ Matheus da Silva Freire. – Brasília, DF, 2013-

38 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Drº Fabricio Braz

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2013.

1. Segurança. 2. BSIMM. I. Prof. Drº Fabricio Braz. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Identificação das lacunas ferramental para dar suporte ao programa de segurança de software

CDU 02:141:005.10

Matheus da Silva Freire

Identificação das lacunas ferramental para dar suporte ao programa de segurança de software

Monografia submetida ao curso de graduação em Bacharel em Engenharia de Softwareda Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 12 de dezembro de 2013:

Prof. Drº Fabricio Braz
Orientador

Drº Paulo Roberto Miranda Meirelles,D.Sc.
Convidado 1

Drº Luiz Augusto Fontes Laranjeira, D.Sc.
Convidado 2

Brasília, DF
2013

Resumo

O resumo deve ressaltar o objetivo, o método, os resultados e as conclusões do documento. A ordem e a extensão destes itens dependem do tipo de resumo (informativo ou indicativo) e do tratamento que cada item recebe no documento original. O resumo deve ser precedido da referência do documento, com exceção do resumo inserido no próprio documento. (. . .) As palavras-chave devem figurar logo abaixo do resumo, antecedidas da expressão Palavras-chave:, separadas entre si por ponto e finalizadas também por ponto. O texto pode conter no mínimo 150 e no máximo 500 palavras, é aconselhável que sejam utilizadas 200 palavras. E não se separa o texto do resumo em parágrafos.

Palavras-chaves: latex. abntex. editoração de texto.

Abstract

This abstract is a brief of conclusion work of course in University of Brasilia, by the student Matheus da Silva Freire and who is oriented by Professor Fabricio Braz.

Key-words: latex. abntex. text editoration.

Lista de ilustrações

Lista de tabelas

OWASP Open Web Application Security Project

BSIMM Building Security In Maturity Model

CR *Code Review*

PT *Penetration Testing*

CMVM *Configuration Management and Vulnerability Management*

SSG *Software Security Group*

SDLC *Software Development Life Cycle*

Sumário

1 Introdução

A segurança da informação e, em especial o aspecto da privacidade, tem sido assunto em destaque nos meios de comunicação. O noticiário tem relatado com muita frequência ocorrências negativas relacionadas a esse tema, revelando uma constante relativização da ética por todos os países/organizações com capacidade técnica para executar ataques, espionagem, fraudes e outras ações intrusivas que comprometam o sigilo, a integridade ou disponibilidade da informação.

Na tentativa de impor maior dificuldade aos adversários em conseguir alcançar os seus objetivos escusos, a segurança da informação como um todo necessita ser reforçada. Uma parte desse todo é a segurança de software, tema com o qual este trabalho de conclusão de curso procurou contribuir, envidando esforços para incrementar o programa de segurança de software com ferramental que suporte a sua execução.

1.1 Contextualização

A segurança da informação é um campo repleto de desafios. Um grande desafio para os defensores, pessoas com a função de garantir as propriedades de segurança de seu sistema, é o fato dos atacantes necessitarem de apenas uma brecha ou vulnerabilidade para comprometer o sistema. Por mais que o defensor eleve o nível de segurança de seu sistema, se uma única falha for deixada de lado, ela poderá ser a porta de entrada usada pelo atacante para abusar do sistema.

Esse desafio se torna ainda maior, quando se compara o recurso disponível para uma organização se defender, face ao efetivo de atacantes focado na exploração de seu sistema. Tais adversários podem ser motivados por recompensas intangíveis, como, por exemplo, a notoriedade pela execução do ataque; pelo retorno financeiro decorrente da comercialização da informação como, por exemplo, pela negociação de dados de cartões de crédito; ou pela vantagem comercial em função da revelação de estratégias de negócio de uma empresa/pais.

Toda essa adversidade impõe à organização a obrigação de lidar estrategicamente com os recursos disponíveis para a segurança da informação, com o intuito de maximizar o resultados. Desta forma, é imperativo que aconteça um fortalecimento dos elos que compõe a segurança da informação como, por exemplo, a segurança de rede, a segurança de estação (*host*), e segurança de software.

O elo da segurança de software demorou a receber atenção, quando se comparado com os outros mencionados. Enquanto as iniciativas em antivírus e filtro de pacotes estão

em evidência há mais de 30 anos, a segurança de software começou a receber maior atenção a partir do ano 2000, com a apla divulgação do programa *Trustworthy Computing Initiative* da *Microsoft*, pelo qual a empresa elevou segurança de software como a sua prioridade estratégica (VIEGA, 2011).

A obtenção de software seguro é uma atividade altamente complexa, que envolve vários fatores a serem considerados e práticas a serem executadas (ALLEN et al., 2008). Essa diversidade se apresenta como um dificultador para as organizações interessadas no assunto, fato que induziu ao desenvolvimento de alguns processos como, por exemplo o SDL da *Microsoft*(MCGRAW, 2012); de organizações como a Open Web Application Security Project (OWASP), além de modelos para profissionalizar um programa de segurança como, por exemplo, OpenSAMM (*Software Assurance Maturity Model* da OWASP e o Building Security In Maturity Model (BSIMM) da Cigital.

Apesar dos modelos OpenSAMM e BSIMM serem pautados na melhoria da maturidade da organização, a natureza deles é distinta, uma vez que o primeiro se coloca como um modelo prescritivo, ou seja, ele enumera ações que, na opinião de alguns especialistas devem ser empreendidas para uma organização produzir software seguro. Já o BSIMM se propõe a ser um modelo descritivo, ou seja, nele encontram-se consolidadas observações realizadas em campo. Essa diferenciação confere ao modelo BSIMM um caráter científico ausente no OpenSAMM, fato que justifica a preferência por ele no desenvolvimento deste trabalho de conclusão de curso.

O BSIMM é um modelo de maturidade que considera preocupações do interesse de um programa de segurança de software. Essas preocupações estão representadas em 109 atividades, distribuídas em 12 práticas, classificadas nos quatro domínios ilustrados pela Figura 1.

The Software Security Framework (SSF)			
Governance	**Intelligence**	**SSDL Touchpoints**	**Deployment**
Strategy and Metrics	Attack Models	Architecture Analysis	Penetration Testing
Compliance and Policy	Security Features and Design	Code Review	Software Environment
Training	Standards and Requirements	Security Testing	Configuration Management and Vulnerability Management

Figura 1 – Domínios e Práticas do BSSIM. McGraw, Migues e West (2013)

Mais detalhes sobre o domínio são dados a seguir, no qual uma breve explanação

do significado de cada um é apresentada:

- Governança: práticas que auxilima o gerenciamento do grupo de segurança(gerenciamento e medição da segurança).
- Inteligência: práticas que auxiliam a organização a melhorar o programa de segurança da informação, repassando o conhecimento obtido em cada iniciativa de segurança de software.
- SSDL Touchpoints: práticas que estão relacionadas com artefatos e processos sobre segurança.Também está relacionada com boas práticas adotadas durante o SDLC (*Software Development Life Cycle*).
- Deployment: práticas relacionadas com a entrega do software ao interessado. (MCGRAW; MIGUES; WEST, 2013)

O BSIMM é resultado de pesquisa que observou vários programas de segurança de software com o intuito de reconhecer e consolidar atividades comuns executadas por diversas organizações. As atividades reconhecidas são categorizadas em três níveis, conforme a sua ocorrência nas organizações. Ou seja, se uma atividade foi percebida em todas as organizações, ela é categorizada como de primeiro nível. À medida que a sua ocorrência diminui, é concedido um nível mais elevado para a atividade.

A Figura 2 apresenta uma visão consolidada da ocorrência de cada uma das práticas nas 67[1] organizações alvo da pesquisa, que envolve participantes dos mais variados ramos (empresas de energia, telefonia, saúde, etc.).

Nesta Figura, o modelo dá a visão de como a atividade foi observada na pesquisa do BSIMM V. Por exemplo, a prática de PT é dividida em 3 níveis, contendo certas atividades por níveis. A atividade 1, PT1.1, foi observada em 62 das 67 organizações participantes.

Essa grande quantidade de práticas não significa que para ter um software seguro é necessário a adoção de todas as práticas em todos os níveis, deve ser feito uma adaptação de acordo com o(s) objetivo(s) proposto(s) pela organização e quais práticas poderão auxiliar a alcançá-lo(s).

Para a construção de software seguro, as melhores empresas investem em apoio ferramental para a execução de práticas. A segurança no *Software Development Life Cycle* (SDLC) deve ser acompanhada por ferramentas e recursos para que possam auxiliar a gestão a medir e monitorar a iniciativa de segurança (BRINK, 2010a). O apoio ferramental é de suma importância uma vez que somente ter uma equipe ou apenas um membro de segurança pode significar em perda de informação oriunda das medições realizadas.

[1] Estes participantes estão disponíveis em: http://bsimm.com/community/

1.2 Problema

Este trabalho tem como maior motivação ou problema a ser solucionado é suportar de ferramentas para facilitar a execução de um programa de segurança de software.

Um programa de suporte visa identificar possíveis métodos, ferramentas que possam auxiliar desenvolvimento de software seguro.A construção de software seguro envolve processos, pessoas, métodos e ferramentas para desenvolver uma iniciativa de forma a auxiliar ao programa de segurança de informação. O processo pautado somente em equipe especializada perde muita informação devido ao tempo que as suas atividades são executadas, uma vez que para executar a atividade de forma a levantar informações do estado da segurança é necessário uma agilidade além do recurso humano.

O apoio ferramental é uma incremento em agilidade e produtividade de membro(s) ou uma equipe especializada em segurança de software, conforme o BSIMM constatou que uma equipe designada SSG (*Software Security Group*) é visualizado em grande quantidade nas organizações participantes.

Acredita-se que algumas atividades de um programa possam ser automatizadas em total ou parcialmente, entretanto uma lacuna na oferta das mesmas é uma situação que as organizações passam a ter quando decidem pela automatização, e isso acarreta em perda de tempo e perda de objetivo de negócio. As ferramentas que suportam a iniciativa de segurança podem incluir em atividades que possam auxiliar uma atividade como pode ser uma ferramenta que engloba toda as atividades de uma prática.

Um suporte ferramental um auxílio a organizações que queiram adotar a automatização durante o programa de segurança para estes tem um meio de auxílio a identificar possíveis ferramentas candidatas de acordo com a atividade preterida. A gestão ferramental dentro do SDLC pode agilizar que informações sejam resgatadas das atividades realizadas.

1.3 Objetivo

Esta seção do trabalho apresenta o objetivo que se deseja conquistar, para tal objetivo é necessário etapas para que estes possa ser realizado, e estes são chamados de objetivos específicos.

1.3.1 Geral

O objetivo de trabalho é oferecer suporte de ferramentas para facilitar a execução de um programa de software.

1.3.2 Específico

- Analisar áreas do modelo de maturidade escolhido;
- Identificar áreas alvo no programa de segurança de software;
- Levantar estado atual do apoio ferramental das áreas escolhidas;
- Elaborar uma arquitetura para a ferramenta;
- Desenvolver um protótipo de ferramenta que dá apoio a execução do programa de segurança de software

1.4 Justificativa

As práticas de um programa de segurança podem ser incluídas em diferentes momentos do SDLC, porém deve dar suporte para que as tomadas de decisões possam ter tempo hábil de fazerem efeito. Essa inclusão de práticas podem levar a uma gama de atividades a serem executadas, conforme é visto na pesquisa do próprio BSIMM.

Para que a prática possa levantar dados que auxiliem nos rumos da segurança da software é necessário que esta seja executada com maior agilidade e que os dados possam ser altamente utilizáveis para alcançar seu propósito. Para tal levantamento, a utilização apenas de capital humano pode ser onerosa e até mesmo dispendiosa se tratando de tempo.

A utilização de ferramentas para a iniciativa de segurança não exclui a utilização de uma boa equipe de consultores para análise de dados. O apoio ferramental tem o intuito de fazer parte, juntamente com o papel humano, de um processo dentro do SDLC somente preocupado em segurança de software.

A utilização de ferramentas é apenas para auxiliar o desenvolvimento, o apoio ferramental não deve ser encarado como substituto de pessoas e boas habilidades que elas possuem (HOWARD, 2004).

Empresas classificadas como melhores, conforme Brink (2010a) realizou em sua pesquisa, utilizam no ciclo de desenvolvimento ferramentas que auxiliem a segurança de software, isso envolve ferramentas de testes, análise de código, ferramentas para gestão de vulnerabilidade, etc. Equipes efetivas utilizam das mais variadas ferramentas relacionadas à iniciativa de segurança durante o SDLC (CHESS; ARKIN, 2011).

1.5 Metodologia

1.5.1 Etapa 1

A primeira etapa consiste no esclarecimento do apoio ferramental que suporta a escolha execução da prática. Esta etapa mostra como está a situação da prática em termos das ferramentas disponíveis e lacunas necessárias para facilitar /viabilizar a execução de atividades de práticas do BSIMM, a partir de revisão bibliográfica.

O esclarecimento de como está o estado atual da prática, é necessário olhá-la por dois lados dentro do contexto de segurança de software. O contexto de gestão da prática e contexto de engenharia.

A análise será orientada pelas seguintes questões que tratam de separadamente detalhes mais afetos a engenharia e gestão

Para responder as perguntas foi utilizado o referencial teórico. As palavras-chaves que conduziram o levantamento bibligráfico foram separadas por práticas do modelo de segurança, e são mostradas na tabela abaixo.

Gestão

Quais indicadores a execução da atividade proporciona?

A geração desses indicadores é passível de automação (geração ou parte dela)?

Se sim, quais ferramentas estão disponíveis?

Engenharia

A realização da prática em si exige apoio ferramental?

Se sim, quais ferramentas estão disponíveis?

Para responder as perguntas, acima citadas, foi utilizada a bibliografia encontrada As palavras-chaves que conduziram o levantamento bibligráfico foram separadas por áreas da segurança, e são mostradas na Tab (1.5.1).

Code Review	***Penetration Testing***	***Configuration Management and Vulnerability Management***
Code review	*penetration testing*	*configuration management in software security*
static analysis	*dynamic analysis*	*vulnerability management in software security*
tools for static analysis	*pen test*	*process for management security*
code scanner	*penetration test in software security*	*vulnerability management*
process of code review in software security	*intrusion tests*	*carrying the vulnerabilities in software security*
code review in software security	*penetration test in software security*	*managing software security*

Tabela 1 – Palavras-chave por prática do BSIMM

1.5.2 Etapa 2

Após o levantamento de toda a situação da prática, é necessário o levantamento de ferramentas candidatas que possam suportar a prática dentro da duas perspectivas (gestão e engenharia). As ferramentas candidatas foram pesquisadas em repositórios de ferramentas com código aberto e também em sítios de pesquisas da *web* para que fossem identificadas até mesmo ferramentas proprietárias, mas que que atendesse o escopo da prática.

A utilização da ferramenta ficará a escolha da organização que está tentando obter um apoio de quais ferramentas eles podem ter para auxiliá-lo. Essas ferramentas são dividas de acordo com a prática e também são divididas em licença das mesmas.

Ferramentas proprietárias eram mais pesquisadas pelo texto que o fabricante disponibiliza na página da internet para explicar um pouco do objetivo daquela ferramenta. As ferramentas *opensource* eram buscadas e quando achado seus repositórios era lido a *wiki*[2].

As ferramentas *opensource* são ferramentas colaborativas no qual quem queira contribuir para o projeto, deve seguir mínimas formalidades para este virar um colaborador da ferramenta. Por isso, algumas ferramentas eram descartadas quando a data de atualização da sua base de dados já passava algum tempo.

1.5.3 Etapa 3

1.5.4 Etapa 4

1.5.5 Etapa 5

1.6 Estrutura do Trabalho

[2] Wiki é uma ferramenta de edição de conteúdo, aonde usuários proprietários ou não podem descrever o conteúdo para futuros usuários

Governance		Intelligence		SSDL Touchpoints		Deployment	
Activity	Observed	Activity	Observed	Activity	Observed	Activity	Observed
[SM1.1]	44	[AM1.1]	21	[AA1.1]	56	[PT1.1]	62
[SM1.2]	34	[AM1.2]	43	[AA1.2]	35	[PT1.2]	51
[SM1.3]	34	[AM1.3]	30	[AA1.3]	24	[PT1.3]	43
[SM1.4]	57	[AM1.4]	12	[AA1.4]	42	[PT2.2]	24
[SM1.6]	36	[AM1.5]	42	[AA2.1]	10	[PT2.3]	27
[SM2.1]	26	[AM1.6]	16	[AA2.2]	8	[PT3.1]	13
[SM2.2]	31	[AM2.1]	7	[AA2.3]	20	[PT3.2]	8
[SM2.3]	27	[AM2.2]	11	[AA3.1]	11		
[SM2.5]	20	[AM3.1]	4	[AA3.2]	4		
[SM3.1]	16	[AM3.2]	6				
[SM3.2]	6						
[CP1.1]	43	[SFD1.1]	54	[CR1.1]	24	[SE1.1]	34
[CP1.2]	52	[SFD1.2]	53	[CR1.2]	34	[SE1.2]	61
[CP1.3]	45	[SFD2.1]	26	[CR1.4]	50	[SE2.2]	31
[CP2.1]	24	[SFD2.2]	29	[CR1.5]	23	[SE2.4]	25
[CP2.2]	28	[SFD2.3]	9	[CR1.6]	25	[SE3.2]	10
[CP2.3]	29	[SFD3.1]	13	[CR2.2]	10	[SE3.3]	9
[CP2.4]	25	[SFD3.2]	9	[CR2.5]	15		
[CP2.5]	35			[CR3.1]	18		
[CP3.1]	14			[CR3.2]	4		
[CP3.2]	11			[CR3.3]	6		
[CP3.3]	8			[CR3.4]	1		
[T1.1]	50	[SR1.1]	48	[ST1.1]	51	[CMVM1.1]	59
[T1.5]	29	[SR1.2]	43	[ST1.3]	55	[CMVM1.2]	59
[T1.6]	23	[SR1.3]	45	[ST2.1]	27	[CMVM2.1]	50
[T1.7]	33	[SR1.4]	27	[ST2.3]	13	[CMVM2.2]	44
[T2.5]	9	[SR2.1]	23	[ST2.4]	11	[CMVM2.3]	30
[T2.6]	13	[SR2.2]	19	[ST3.1]	8	[CMVM3.1]	6
[T2.7]	9	[SR2.3]	19	[ST3.2]	6	[CMVM3.2]	6
[T3.1]	4	[SR2.4]	22	[ST3.3]	5	[CMVM3.3]	2
[T3.2]	4	[SR2.5]	8	[ST3.4]	7		
[T3.3]	8	[SR3.1]	12				
[T3.4]	9						
[T3.5]	5						

Figura 2 – *Scorecard* do BSIMM com as atividades levantadas. (MCGRAW; MIGUES; WEST, 2013)

2 Desenvolvimento

2.1 Etapa 1

Conforme dito na seção 1.5.1, a etapa 1 consiste no esclarecimento do apoio ferramental. Entretanto, este é somente um dos passos a serem seguidos por tal etapa para determinar quais áreas foram escolhidas para serem investigadas. Porém, as áreas foram retiradas do modelo de maturidade escolhido, no qual o BSIMM (MCGRAW; MIGUES; WEST, 2012) foi o modelo adotado para ser o guia deste trabalho.

2.1.1 O Modelo

Para tal trabalho, foi escolhido o BSIMM dentre os demais processos, isso se deve porquê, o BSIMM pode ser usado para identificar como a segurança do software se encontra por meio de medidas objetivas (MCGRAW, 2012). O objetivo do modelo é ajudar as organizações a planejar, cumprir e medir a iniciativa de segurança McGraw, Migues e West (2013).

2.1.2 Escolha das Práticas

Após uma análise das práticas do BSIMM foi levantado práticas alvos para o desenvolvimento do trabalho. Entretanto, devido ao tempo deste trabalho foram escolhidos apenas 3, que são áreas que necessitem de mais apoio ferramental para serem executadas. As áreas são:

- CR ou revisão de código para segurança, ocupa um nível elevado entre as melhores práticas de segurança de software (CHESS; MCGRAW, 2004). A revisão de código verifica o código antes de sua entrega ao cliente, ou seja, uma verificação durante o desenvolvimento do software.

- PT ou teste de penetração são testes de segurança, que focam em identificar vulnerabilidades nas aplicações durante a fase de teste/validação (AGARWWAL et al., 2008). O teste de penetração resulta em analisar possíveis vulnerabilidades que os atacantes possam explorar nas aplicações.

- CMVM ou Gestão da Configuração e Vulnerabilidade é uma área mais voltada para gestão da segurança da informação durante o ciclo de desenvolvimento. Gestão de configuração segue um caminho sequencial, similar ao desenvolvimento da aplicação, e gira em torno do controle e liberação de versões diferentes de produtos (MEYER,

2007). VM, como gestão de vulnerabilidades é chamado, se faz necessário o uso de ferramentas de segurança especializadas e fluxos de trabalho que possam contribuir ativamente para eliminar riscos exploráveis. (WILEY, 2008)

2.1.3 Pesquisa

Para identificar como se encontra a prática, foi necessário fazer uma pesquisa como se encontra o estado da arte da prática. Para tal finalidade, utilizou-se as palavras da Tab. (1.5.1). Essas palavras foram utilizadas no levantamento de suporte ferramental nas referências bibliográficas evidenciada em artigos científicos. Para a pesquisa dos referenciais teóricos para o levantamento foram realizados em sítios como:

1. IEEE - *Institute of Electrical and Electronics Engineers* ou Instituto de Engenheiros Eletricistas e Eletrônicos
2. ACM - *Association for Computing Machinery*
3. *Science Direct*
4. OWASP ou Projeto Aberto de Segurança de Aplicação WEB

Para definição de quais artigos poderiam auxiliar no desenvolvimento deste trabalho, alguns critérios foram adotados para eliminação. Os artigos encontrados eram lidos a seção de introdução, disponibilizado antes mesmo de abrir o documento para se ter uma visão de todo o documento. Outro critério de exclusão é por período, segundo (WAZLAWICK, 2009) no ramo de computação, há utilização de fontes mais antigos diminui a credibilidade do trabalho desenvolvido.

Para pesquisa dos textos científicos foi utilizado sítios de pesquisa de texto científicos como *http://scholar.google.com.br/*, *ieeexplore.ieee.org*,*http://www.acm.org/* e além da ferramenta do CAPES que indexa fontes bibliográficas dos mais variados sítios da internet que é o *www.periodicos.capes.gov.br*.

As bases bibliográficas foram selecionadas levando em consideração que poderiam dar resultados de artigos publicados em eventos, pois eventos contem passos até a publicação que somente os melhores artigos são escolhidos (WAZLAWICK, 2009), porém isso não significa que artigos publicados fora de eventos foram totalmente desconsiderados, porém a relevância de artigos presentes em anais de eventos era mais considerada.

2.1.4 Questões de Pesquisa

Para não ser uma pesquisa que seja somente levantar referencias teóricos, foram criados 2 vertentes para serem respondidas. Essas duas áreas ou caminhos que a prática

pode ter são caminhos que possam auxiliar a prática dentro do contexto de segurança de software.

A prática vista pela gestão é ter a visão da ocorrência da prática dentro do programa de segurança. Uma análise gerencial de cada prática pode entregar subsídios para determinar possíveis ações e analisar o que está sendo feito. A gestão da prática deve ser encarada como uma subprática, aonde é necessário uma adaptação para cada organização possa executar de acordo com seus objetivos estabelecidos.

A visão da engenharia trata da realização da prática para atingir os objetivos que a gestão estabeleceu para o programa de segurança. Os pontos que desejam-se alcançar são importantes para determinar quais o níveis tecnológico envolvidos para determinar se há o apoio ferramental, seja por ferramentas que possam auxiliar aquela tecnologia utilizada pela organização, seja por tecnologia usada que uma ferramenta emprega que possa dar um melhor resultado da gestão.

Para vislumbrar como está a prática destes dois pontos, é necessário que certas questões sejas respondidas observado o referencial teórico e o próprio modelo utilizado. As 2 visões tem perguntas que devem ser respondida ao longo do tempo deste trabalho e com isso pode-se ter um olhar aprofundado de qual prática do modelo devemos escolher. As perguntas de cada questão são:

Gestão

- Quais indicadores a execução da atividade proporciona?
- A geração desses indicadores é passível de automação (geração ou parte dela)?
- Se sim, quais ferramentas estão disponíveis?

Engenharia

- A realização da prática em si exige apoio ferramental?
- Se sim, quais ferramentas estão disponíveis?

Tais perguntas foram respondidas por prática, conforme o levantamento de palavras-chave também foi dividido por prática, e com isso pode-se observar o estado atual do apoio ferramental da prática.

2.1.5 Pesquisa Ferramental

Para a determinação do como está o estado atual do apoio ferramental que a prática possui, levou-se em conta os textos dos referenciais teóricos. Textos com tal finalidade possui um estudo de caso sobre a utilização de uma ferramenta, como também poderiam

apresentar as ferramentas que atendiam certas necessidades evidencias pelas práticas. A pesquisa era guiada pelas palavras-chave, conforme falado na Tabela 1.5.1

Este não foi o único meio de se levantar ferramentas que auxilie as práticas, a pesquisa também envolvia pesquisas em sítios específicos de aplicações, como por exemplo:

- Github - aplicação *web* que prover uma base de repositório de ferramentas *open-source*. Disponível em: *https://github.com/*
- Sourceforge - aplicação que contém repossitório dos mais variados tipos de ferramentas. Disponível em: *http://sourceforge.net/*
- Google Code - repositório da empresa Google no qual os usuários podem utilizar para versionamento e controle de suas aplicações. Disponível em: *http://code.google.com*

2.1.6 Práticas Alvo

Conforme falado no item 2.1.2, as práticas escolhidas foram *Code review*, *penetration testing* e *Configuration Management and Vulnerability Management*. Nesta seção é explicado o que é cada prática do ponto de vista de segurança da informação.

2.1.6.1 CR

é uma prática que é utilizada durante o desenvolvimento da aplicação. Essa prática visa identificar possíveis erros ainda na fase de codificação do sistema a fim de posteriormente este erro não aparecer como uma possível vulnerabilidade. Alguns erros de segurança, podem ser remediados se forem descobertos na codificação (MCGRAW, 2008). Esta prática é encarada como uma necessidade que deve ser realizado para alcançar um nível maior de segurança. Negligenciá-lo não é uma opção. Howard (2006)

A utilização de ferramentas para análise de código é estudada há algum tempo. Utilizando um maior número de ferramentas de revisão de código, também conhecida como análise estática, a capacidade de detectar um grande número de vulnerabilidades é maior do que a utilização de somente uma única ferramenta (TEIXEIRA, 2007). Portanto, a prática possuindo um maior número de apoio ferramental, deixará passar um menor número de vulnerabilidades.

2.1.6.2 PT

é um teste realizado antes da entrega da aplicação ao cliente, é categorizado como um teste dinâmico, pois a aplicação já está em execução. No *pentest*, como é conhecido, os testes tem como função parecer como um atacante para detectar vulnerabilidades de segurança (AGARWWAL et al., 2008). *Penetration testing* tem o intuito, também, de dimensionar a capacidade da segurança (NAIK, 2009).

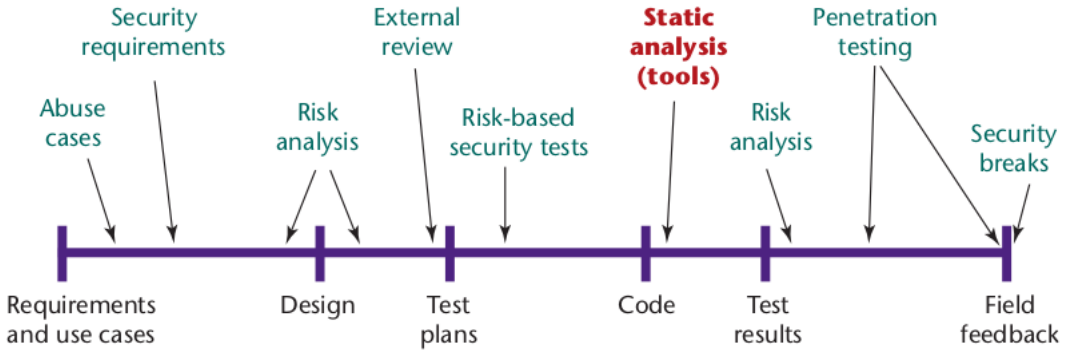

Figura 3 – *Code Review* no SDLC. Chess e McGraw (2004)

O momento mais indicado para execução do *pentest* são os momentos finais do ciclo de desenvolvimento,conforme Fig. 4, porém isso não é tido como algo obrigatório. *hackear* atualmente tem uma maior facilidade pela disponibilidade de boas ferramentas para melhorar seu ofício. (ENGEBRETSON, 2011). Por isso ter uma boa base de dados sobre ferramentas disponíveis pode auxiliar as organizações a conquistar níveis de segurança de software. De acordo com Brink (2010b), em sua pesquisa, as melhores empresas são mais diferenciadas por usar *penetration testing*.

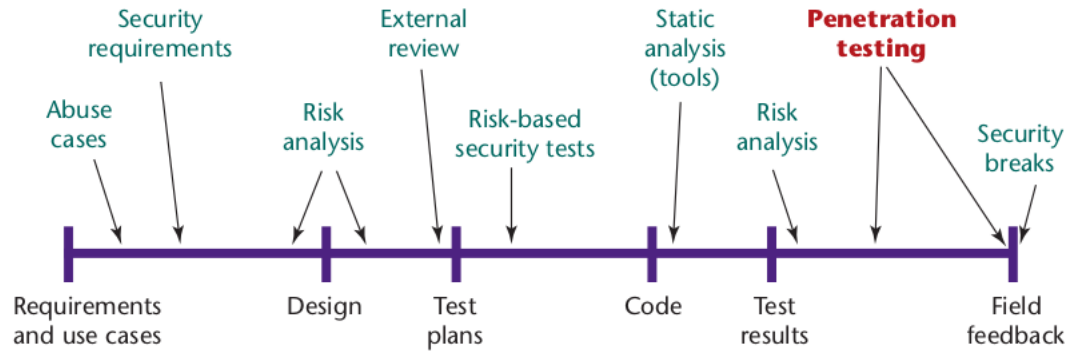

Figura 4 – *Penetration testing* no SDLC. Arkin, Stender e McGraw (2005)

Conforme (ENGEBRETSON, 2011), teste de penetração também é conhecido como:

- *Pen Testing*
- PT
- *Ethical Hacking*

A utilização do PT não exclui a prática de *code review*, 2.2.1, as duas práticas são efetivas em levantar dados diferentes, enquanto o PT levanta vulnerabilidades que os atacantes possam utilizar em uma aplicação já em estado de execução, *code review*, pode,

utilizar o código fonte para levantar possíveis pontos de vulnerabilidades que possam existir. A grande diferença entre as duas é que a última necessita de se ter acesso ao código fonte, já a primeira não necessita (ANTUNES; VIEIRA, 2009).

A vantagem de se utilizar o *pentest* é que ele nos da um bom entedimento do da segurança de software com sua real execução McGraw (2012).

2.1.6.3 CMVM

A gestão de configuração e gestão de vulnerabilidades tem a intenção de ser um prática com o foco em dar um suporte à gerência. O gerenciamento de vulnerabilidades é ciclo que gira em torno de vulnerabilidades (FOREMAN, 2010). Este ciclo é um processo de captura, avaliação e correção de vulnerabilidades, além de utilizar as informações para posteriores iniciativas.

A gestão de vulnerabilidades está relacionado como a organização trata as vulnerabilidades descobertas. Este tratamento envolve o uso de ferramentas, fluxos das vulnerabilidades que irão auxiliar a equipe responsável a eliminar estes riscos exploráveis (WILEY, 2008).

A gestão de configuração, no contexto de segurança de software, está mais ligada a gestão de ativos que a organização possui e quais os relacionamentos elas possuem.

2.2 Etapa 2

A etapa dois consiste no levantamento ferramental para cada prática. As práticas foram avaliadas pela 2 perspectivas para levantamento ferramental. A perspectiva da gestão foi dado enfoque maior em ferramentas que pudessem auxiliar a gerência a ter um apoio para controle da prática. A perspectiva da engenharia ou execução dá o enfoque de ferramentas em que executem o levantamento de dados.

As práticas foram respondidas, levando em consideração as questões, já citadas no item 2.1.4.

2.2.1 *Code review*

Para detectar o suporte ferramental do *code review*, foi respondido as seguintes perguntas:

2.2.1.1 Gestão

Quais indicadores a execução da atividade proporciona?

A prática proporciona indicadores sobre o código, indicadores no qual é possível visualizar o estado do desenvolvimento da aplicação do ponto de vista da segurança. Após a revisão de código é possível uma análise de vulnerabilidade, no qual é possível saber qual trecho ou se o código completo tem algum ponto que o atacante pode explorá-lo.

Os indicadores levantadas através das atividades da prática são:

- # de defeitos no código;
- porcentagem de código coberto;
- # vulnerabilidades por trecho de código;
- # pontos exploráveis;
- porcentagem de código não utilizado.

A geração desses indicadores é passível de automação (geração ou parte dela)?

As atividades proposta da prática é passível de suporte ferramental,em sua maioria, principalmente quando se fala das atividades de análise do código (com o intuito de detectar erros ou para detectar códigos maliciosos), porém como a própria prática cita mesmo sendo automatizada ela requer um bom julgamento do SSG para poder aprimorar o quesito segurança da organização (MCGRAW; MIGUES; WEST, 2012).

Se sim, quais ferramentas estão disponíveis.

2.2.1.2 Engenharia

A realização da prática em sí exige apoio ferramental?

A utilização de ferramentas para a geração de métricas sobre o código pode auxiliar o julgamento dos auditores, porém é importante que se tenha ideia que somente uma ferramenta pode não ser necessário para levantar os dados viáveis. Por isso é necessário que os "auditores"tenham em mente o que querem com a revisão de código e o levantamento de métricas é um processo mais rápido utilizando ferramentas de análise de código (MCGRAW, 2008).

Utilizando a revisão de código tem benefícios, além de levantar erros no código antes da entrega ao cliente, a prática levantar possíveis vulnerabilidades no início, e quanto mais tempo passa, mais caro fica para corrigí-la (MCGRAW, 2008)

Se sim, quais ferramentas estão disponíveis.

Ferramenta	Licenciamento	Propósito	Linguagem Suportada	Site
Veracode	Proprietário	Detectar vulnerabilidades associadas a segurança utilizando o code review	Java, JSP, .NET- C#, .NET - VB.NET, ASP.NET, .NET - C++/CLI, C/C++, PHP, ColdFusion, iOS, Android, J2ME, Ruby, Classic ASP, VB6, VBScript, Flash	http://www.veracode.com/security/code-review
Agnitio	Open Source	Auxiliar desenvolvedores a fazer uma revisão do código atrelado a segurança	PHP, Perl, Python, Tcl, Ruby e Shell scripts	http://agnitiotool.sourceforge.net/
Java Code Review	Open Source	Revisão de código	Feita especialmente para java, porém suporta outras linguagens	http://jcodereview.sourceforge.net/
Findbugs	Open Source	Revisão de código para detectar bugs	Java	http://findbugs.sourceforge.net/
phpdepend, phpmessdetector, phpcpd, phpdcd	Open Source	Revisão de código para detectar bugs (código duplicado, código não utilizado)	PHP	http://pdepend.org/, http://phpmd.org/, https://github.com/sebastianbergmann/phpcpd
Qafoo Code Review Tool	Open Source	Revisão de código	PHP	https://github.com/Qafoo/review
Code Analysis	Proprietário	Revisão de código	.Net	http://msdn.microsoft.com/en-us/library/3z0aeatx.aspx
FxCop	Proprietário	Revisão de código quando forem integrado	.Net	http://msdn.microsoft.com/en-us/library/bb429476(v=vs.80).aspx
Analizo	Open Source	Revisão de código	C, C++ e Java	http://analizo.org/
pylint	Open Source	Revisão de código	python	http://www.pylint.org/

Tabela 2 – Ferramentas da Prática de *Code Review*

2.2.2 *Penetration Testing*

Para detectar como é o estado do suporte ferramental, foram respondidas as questões, conforme falado no item 2.1.4

2.2.2.1 Gestão

Quais indicadores a execução da atividade proporciona?

A atividade de pentest pode ter um rastreamento do código. Aonde é possível rastrear o código analisando quesitos de segurança (possibilidades de falhas), vulnerabilidades. Então é possível saber:

- # de vulnerabilidades por funcionalidades;
- # de defeitos por funcionalidades;
- # vulnerabilidades por período;
- # de defeitos por período;
- # vulnerabilidades por trecho de código;
- # defeitos totais;
- # vulnerabilidades totais

A geração desses indicadores é passível de automação (geração ou parte dela)?

O levantamento dos indicadores pode ser feito através de uma conjunção de ferramentas, aonde podemos ter uma ferramenta de gestão de pessoas juntamente com uma ferramenta para realizar penetration test, porém uma ferramenta no mercado encontrada foi a *metasploit*, aonde é possível fazer uma auditoria sobre *penetration test* a fim de gerenciar tudo que o teste levanta (priorização, verificar a eficácia da segurança da equipe, etc).

Se sim, quais ferramentas estão disponíveis.

A ferramenta que possa auxiliar a gestão do *penetration testing* é a *Metasploit*, ferramenta com licenciamento *open source*. Seu código está aberto para comunidade que deseja alterá-lo de acordo com seu objetivo. Site: http://www.metasploit.com/

2.2.2.2 Engenharia

A realização da prática em sí exige apoio ferramental?

A prática da *pentest* é, necessariamente, feita por ferramenta, aonde é inviável um humano ficar tentando encontrar falhas de segurança. O apoio ferramental será para descobrir, antes mesmo do atacante, possíveis falhas de seguranças (vulnerabilidades) e falhas que já estão no código que são passíveis de correção.

A utilização de ferramentas de *pentest* levantar poucas informações em relação a métricas de segurança, porém sua ausência pode não indicar o problema no qual você deve focar (MCGRAW, 2012)

Se sim, quais ferramentas estão disponíveis.

Ferramenta	**Licenciamento**	**Linguagem Suportada**	**Site**
Zaproxy	Open Source	Não informado	https://code.google.com/p/zaproxy/
Veracode Penetration Testing	Proprietário	Não informado	http://www.veracode.com/security/penetration-testing
Panoptic	Open Source	Não Informado	http://websec.ca/blog/view/panoptic

Tabela 3 – Ferramentas da Prática de *Penetration Testing*

2.2.3 *Configuration Management and Vulnerability Management*

Para identificar o estado atual da prática em relação a seu apoio ferramental, foram respondida as questões do item 2.1.4:

2.2.3.1 Gestão

Quais indicadores a execução da atividade proporciona?

A atividade de *configuration management* levar a indicadores dos ativos de software, incluindo histórico de sua evolução e rastreamento com os demais ativos. A atividade de *vulnerability management* é o rastremento das vulnerabilidades encontradas associadas a tais ativos.

Os indicadores são:

- # capacidade de análise de ocorrência da equipe
- # volume de código rastrado
- # de incidentes tratados por período
- # de incidentes causados por uma vulnerabilidade
- # de pessoas envolvidas com a resposta a incidentes por período
- # de incidentes por aplicativo/software por período
- # de aplicativos impactados por incidente
- # de vulnerabilidades reveladas por período, por aplicação, por ativo, por componete, por método de análise (estática ou dinâmica), por risco

A geração desses indicadores é passível de automação (geração ou parte dela)?

Os indicadores citados são sim passíveis de atuomação. Essa automação seria mais uma ferramenta para controlar o número absoluto levantado e quantos já foram remediados, se isso for caso.

Se sim, quais ferramentas estão disponíveis.

A Tabela (2.2.3.1) contem as ferramentas para a atividade de gestão de vulnerabilidades.

Ferramenta	Licenciamento	Site
Tripwire IP360	Proprietário	http://www.tripwire.com/it-security-software/enterprise-vulnerability-management/tripwire-ip360/
Secunia VIM 4.0	Proprietário	http://secunia.com/vulnerability
FortiScan	Proprietário	http://www.fortinet.com/products/fortiscan/
QualysGuard	Proprietário	http://www.qualys.com/enterprises/qualysguard/vulnerability-management/
Nessus	Proprietário	http://www.tenable.com/solutions/vulnerability-management
threadfix	Open Source	https://code.google.com/p/threadfix/

Tabela 4 – Ferramentas para Gestão de Vulnerabilidades

A Tabela (2.2.3.1) contem as ferramentas para a atividade de gestão de configuração.

Ferramenta	Licenciamento	Site
Tripwire Enterprise	Proprietário	http://www.tripwire.com/it-security-software/scm/tripwire-ente
Tripewire CCM	Proprietário	http://www.tripwire.com/it-security-software/scm/ccm/
QualysGuard Policy Compliance	Proprietário	http://www.qualys.com/enterprises/qualysguard/policy-compli
Nessus Configuration & Compliance Auditing	Proprietário	http://www.tenable.com/solutions/configuration-auditing

Tabela 5 – Ferramentas para Gestão da Configuração de Software

2.2.3.2 Engenharia

A realização da prática em sí exige apoio ferramental?

A realização da prática CMVM, pela perspectiva da gestão de configuração, demanda um forte apoio ferramental, uma vez que é inviável rastrear manualmente as características dos ativos de software ao longo de sua evolução, bem como os relacionamentos entre tais ativos. Pela perspectiva da gestão de vulnerabilidades, essa dependência é menor, entretanto a ausência de ferramenta que estabelece o relacionamento entre os ativos e os respectivos incidentes e vulnerabilidades torna-se uma atividade entediante e tendenciosa ao erro e assim ocasionar perda de integridade, em razão da constante demanda de atualização.

Se sim, quais ferramentas estão disponíveis.

Ferramenta	Licenciamento	Site
Bugzilla	Open Source	http://www.bugzilla.org/
MantisBT	Open Source	http://www.mantisbt.org
Trac	Open Source	http://trac.edgewall.org/
Redmine	Open Source	http://www.redmine.org/
Fossil	Open Source	http://www.fossil-scm.org
Jira	Proprietário	https://www.atlassian.com/software/jira

Tabela 6 – Ferramentas para execução da prática CMVM

2.3 Etapa 3

2.4 Etapa 4

2.5 Etapa 5

3 Conclusão

Referências

AGARWWAL, A. et al. Owasp testing guide v3. 0 (2008). 2008. Disponível em: <http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Owasp+testing+guide+2008#1http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Owasp+testing+guide+v3.+0+(2008)#0>. Citado 2 vezes nas páginas 23 e 26.

ALLEN, J. H. et al. *Software Security Engineering: A Guide for Project Managers (The SEI Series in Software Engineering)*. 1. ed. [S.l.]: Addison-Wesley Professional, 2008. ISBN 032150917X, 9780321509178. Citado na página 16.

ANTUNES, N.; VIEIRA, M. Comparing the Effectiveness of Penetration Testing and Static Code Analysis on the Detection of SQL Injection Vulnerabilities in Web Services. *2009 15th IEEE Pacific Rim International Symposium on Dependable Computing*, Ieee, p. 301–306, nov. 2009. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5369093>. Citado na página 28.

ARKIN, B.; STENDER, S.; MCGRAW, G. Software penetration testing. *IEEE Security and Privacy Magazine*, v. 3, n. 1, p. 84–87, jan. 2005. ISSN 1540-7993. Citado 2 vezes nas páginas 9 e 27.

BRINK, D. *Securing Your Applications*. [S.l.], 2010. 26 p. Citado 2 vezes nas páginas 17 e 19.

BRINK, D. Security and the Software Development Lifecycle : Secure at the Source Business Context : Three Ways to Play – Part Three. n. December, p. 14, 2010. Citado na página 27.

CHESS, B.; ARKIN, B. Software Security in Practice. *IEEE Security & Privacy Magazine*, v. 9, n. 2, p. 89–92, mar. 2011. ISSN 1540-7993. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5739646>. Citado na página 19.

CHESS, B.; MCGRAW, G. Static analysis for security. *Security & Privacy, IEEE*, 2004. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1366126>. Citado 3 vezes nas páginas 9, 23 e 27.

ENGEBRETSON, P. *The Basics of Hacking and Penetration Testing: Ethical Hacking and Penetration Testing Made Easy*. Waltham: [s.n.], 2011. 178 p. ISBN 9781597496551. Disponível em: <http://medcontent.metapress.com/index/A65RM03P4874243N.pdf>. Citado na página 27.

FOREMAN, P. *Vulnerability Management*. 1 edition. ed. United Kingdom: Auerbach Publications, 2010. 330 p. ISBN 1439801509. Citado na página 28.

HOWARD, M. Building more secure software with improved development processes. *Security & Privacy, IEEE*, p. 63–65, 2004. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1366122>. Citado na página 19.

HOWARD, M. A. A process for performing security code reviews. *Security & Privacy, IEEE*, 2006. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1668009>. Citado na página 26.

MCGRAW, G. Automated code review tools for security. *Computer*, 2008. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4712512>. Citado 2 vezes nas páginas 26 e 29.

MCGRAW, G. Software Security. *Datenschutz und Datensicherheit - DuD*, v. 36, n. 9, p. 662–665, set. 2012. ISSN 1614-0702. Disponível em: <http://link.springer.com/10.1007/s11623-012-0222-3>. Citado 4 vezes nas páginas 16, 23, 28 e 31.

MCGRAW, G.; MIGUES, S.; WEST, J. *Building Security In Maturity Model*. [S.l.], 2012. Citado 2 vezes nas páginas 23 e 29.

MCGRAW, G.; MIGUES, S.; WEST, J. *Building Security In Maturity Model*. [S.l.], 2013. Disponível em: <http://w.secappdev.org/handouts/2010/GaryMcGraw/bsimm15thingslores.pdf>. Citado 5 vezes nas páginas 9, 16, 17, 22 e 23.

MEYER, A. *Configuration Management in the Security World*. 2007. Disponível em: <http://www.sans.edu/research/security-laboratory/article/meyer-config-manage>. Citado na página 24.

NAIK, N. e. a. Penetration Testing: A Roadmap to Network Security. *Journal of Computing*, v. 1, n. 1, p. 187–190, 2009. Disponível em: <http://arxiv.org/abs/0912.3970>. Citado na página 26.

TEIXEIRA, E. P. L. Ferramenta de análise de código para detecção de vulnerabilidades. 2007. Disponível em: <https://docs.di.fc.ul.pt/jspui/handle/10455/3090>. Citado na página 26.

VIEGA, J. Ten years of trustworthy computing: Lessons learned. *Security Privacy, IEEE*, v. 9, n. 5, p. 3–4, 2011. ISSN 1540-7993. Citado na página 16.

WAZLAWICK, R. *Metodologia de pesquisa para ciência da computação*. 6ª. ed. Rio de Janeiro: [s.n.], 2009. ISBN 978-85-352-3522-7. Disponível em: <http://books.google.com/books?hl=en&lr=&id=ZLbCKKWQ6OQC&oi=fnd&pg=PA2&dq=Metodologia+de+Pesquis+para+Ci%C3%AAncia+da+Computa%C3%A7%C3%A3o&ots=a-f1CWM7hU&sig=WaWru3lr2_NGWKljhZEAptglW2c>. Citado na página 24.

WILEY, J. *Vulnerability Management for Dummies*. Chichester, England: [s.n.], 2008. ISBN 9780470694572. Citado 2 vezes nas páginas 24 e 28.