RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

DESENVOLVIMENTO SEGURO DE APLICAÇÕES WEB

DEVELOPMENT APPLICATIONS SAFE WEB

Rodrigo Ronner Tertulino da Silva[1], Rommel Wladimir de Lima[2], Cicilia Raquel Maia Leite[3]

Submetido em: 26/03/2021
Aprovado em: 14/04/2021

RESUMO
A escalabilidade, portabilidade e fácil acesso providos pela plataforma *Web* têm popularizado seu uso no desenvolvimento de diversas aplicações. Porém, o crescente número de incidentes relacionados a insegurança levanta preocupações quanto à sua seguridade. Uma parte destes incidentes ocorre da falta de consideração de segurança durante as etapas de desenvolvimento, pois é comum que não sejam utilizadas técnicas para mitigação e prevenção de falhas de segurança no ciclo de vida de desenvolvimento de um software. A segurança da informação e o desenvolvimento de sistemas são áreas que neste trabalho estão integradas no ciclo de desenvolvimento de um *software*. Dessa forma, o presente trabalho tem como objetivo demonstrar uma Metodologia Aplicada ao Desenvolvimento Seguro de Aplicações *Web* (MADS-WEB), por meio da utilização de práticas de segurança de *software* ao longo do ciclo de vida do *software*.

PALAVRAS-CHAVE: Segurança em Computadores. Segurança em Aplicações. *Software* Seguro. Técnicas de Ataque. Plataforma *Web*.

ABSTRACT
The scalability, portability, and easy access provided by the Web platform have popularized its use in developing several applications. However, the growing number of incidents related to insecurity raises concerns about their security. A part of these incidents occurs from the lack of security consideration during the development stages. Commonly, techniques to mitigate and prevent security flaws in the software development lifecycle are not used. Information security and systems development are areas that in this work are integrated into the software development cycle. Thus, the present work aims to demonstrate a Methodology Applied to the Secure Development of Web Applications (MADS-WEB) through software security practices throughout the software life cycle.

KEYWORDS*: Computer security. Security Applications. Software safety. Techniques of attack. Web platform.*

[1] Doutorando em ciências e tecnologias da Informação pela universidade de Coimbra, Portugal. Mestre em Ciência da Computação pela UERN/UFERSA. Possui graduação em Sistemas de Informação, também possui MBA em Gestão de Negócios pela UNP. Atualmente Professor Redes de Computadores no IFRN com dedicação exclusiva. Filiação: Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte - IFRN, Campus Mossoró, Rio Grande do Norte - rodrigo.tertulino@ifrn.edu.br - https://orcid.org/0000-0002-7594-9312

[2] Doutorado em Engenharia Elétrica pela Universidade Federal do Rio Grande do Norte. Atualmente é Professor Adjunto IV na Universidade do Estado do Rio Grande do Norte e Membro permanente no Programa de Pós-Graduação em Ciência da Computação (associação ampla entre UERN/UFERSA) e no POSENSINO. Filiação: Universidade do Estado do Rio Grande do Norte - UERN, Mossoró, Rio Grande do Norte. rommel.lima@gmail.com - https://orcid.org/0000-0002-0309-6322

[3] Doutorado em PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E COMPUTAÇÃO pela Universidade Federal do Rio Grande do Norte. Servidora efetiva da UERN desde 2006, é professora Adjunta IV, lotada no Departamento de Informática, Faculdade de Ciências Exatas e Naturais (FANAT), no Campus Central, onde ministra aulas no curso de graduação em Ciência da Computação. Filiação: Universidade do Estado do Rio Grande do Norte - UERN, Mossoró, Rio Grande do Norte. cicilia.maia@gmail.com - https://orcid.org/0000-0003-1857-6238

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

INTRODUÇÃO

Com a crescente globalização e com a utilização em larga escala da internet, cada vez mais interativa, os usuários se veem cercados de facilidades no mundo digital. Hoje os sites estão cada vez mais dinâmicos e interativos, gerando, assim, uma troca de informações entre servidores e usuários. É nessa troca de informações que *hackers* podem se aproveitar e acessar informações em servidores *Web*. Uma aplicação desprotegida se torna alvo fácil para os *hackers* profissionais (JACOBS, 2011).

Os ataques estão cada vez mais utilizando métodos automatizados de exploração de vulnerabilidades. Segundo dados do Cert.br, de janeiro a dezembro de 2013, 24% dos incidentes reportados no Brasil, tinham como foco tentativas de fraudes, 5% estavam direcionadas a aplicações Web (CERT.BR, 2014).

Neste cenário, as vulnerabilidades acontecem por falha de projeto, implantação ou configuração de *software, e estas* quando exploradas por um atacante, ocasiona violação de segurança. Podem também resultar em roubo de dados confidenciais, quebra de integridade de dados ou afetar a disponibilidade.

Uma forma de mitigar as vulnerabilidades é fazer uso de uma metodologia para construção de *software* que contemple aspectos de segurança durante todo ciclo de desenvolvimento.

Foram identificadas na literatura algumas pesquisas que fundamentam o trabalho:

- Viana (VIANA, 2013) apresenta um *framework* de alto nível fundamentado na necessidade de intensificar as interações entre as equipes de desenvolvimento, arquitetura e administração de dados;
- Aoki (AOKI, 2011) propusera práticas de segurança a serem aplicadas durante o processo de desenvolvimento de *software* Web, tendo como foco o processamento de formulários e sistemas de *login* que minimizem os riscos, aumentando a qualidade e confiabilidade do produto final. Destacando a importância de fazer uso do OWASP *TOP* 10 como linha de base para análise de vulnerabilidades em aplicações Web;
- Floyd (FLOYD, 2012) apresenta algumas vulnerabilidades encontradas no Moodle, tais como: *Session Hijacking Found, Cookie; XSS Injection; Session Management Flaw(s) e Quiz Engine Flaw(s).*

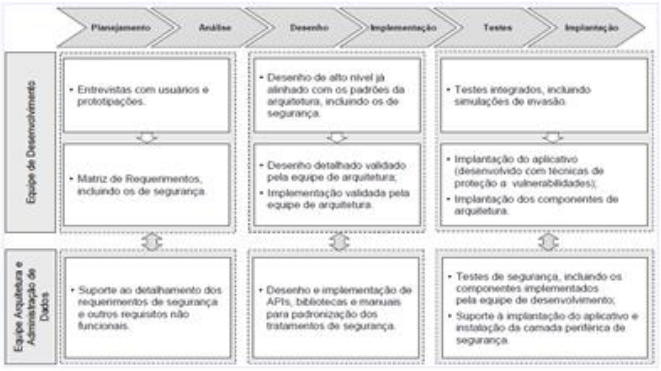
Viana apresenta um *framework* de alto nível para incorporação das atividades de segurança a partir das primeiras fases do ciclo de vida de uma aplicação (*built-in security*) (VIANA, 2013).

A contribuição deste trabalho é um conjunto de recomendações de segurança e uma proposta de um *framework* para o desenvolvimento seguro de aplicações. Este *framework* está fundamentado na necessidade de intensificar as interações entre as equipes de desenvolvimento, arquitetura e administração de dados, reforçando a atenção necessária à segurança desde o início do ciclo de vida de uma aplicação. A figura 1 ilustra o *framework* de alto nível para

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

incorporação das atividades de segurança a partir das primeiras fases do ciclo de vida de uma aplicação.

Figura 1: Framework (*built-in security*)



Fonte: VIANA, 2013.

Este artigo, em relação ao trabalho apresentado (VIANA, 2013), além de apresentar uma metodologia aplicada à construção de sistemas *Web* em que todas as fases e etapas são contempladas com aspectos de segurança, detalha e especifica todas as fases e etapas necessárias para a construção de um sistema seguro, ao contrário do citado artigo que não especifica e nem detalha quais os objetivos de cada etapa e não realiza nenhuma validação a fim de evidenciar sua aplicabilidade em sistemas *Web* existentes. Neste artigo é proposto um conjunto de ferramentas que podem ser utilizadas para a realização de Testes de Vulnerabilidades, o que não foi realizado no artigo em questão.

Aoki *at al* propuseram práticas de segurança a serem aplicadas durante o processo de desenvolvimento de *software Web*, tendo como foco o processamento de formulários e sistemas de *login* que minimizam os riscos, aumentando, assim, a qualidade e confiabilidade do produto final. No artigo são apresentados: conceitos de segurança da informação, as vulnerabilidades mais comuns existentes em *software Web* e algumas práticas que devem ser aplicadas durante o desenvolvimento. O foco no primeiro momento é no processamento de formulários e sistemas de *login*, já que eles são os principais alvos de injeção de códigos e *Cross-site scripting (*XSS*)* (AOKI, 2011).

Neste artigo os autores não avaliaram todas as vulnerabilidades conhecidas em aplicações *Web*, conforme sugere OWASP TOP 10 (OWASP, 2014), eles apenas se detiveram a elencar as vulnerabilidades relacionadas ao *login* dos usuários. Na metodologia proposta neste artigo todas as 10 (dez) vulnerabilidades são avaliadas em um sistema *Web* existente a fim de evidenciar problemas relacionados a todas as vulnerabilidades.

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

Floyd apresenta algumas vulnerabilidades encontradas no Moodle como: *Session Hijacking Found; XSS Injection; Session Management Flaw(s) e Quiz Engine Flaw(s).* Os testes foram aplicados na versão 2.1 (FLOYD, 2012).

Os trabalhos citados nessa seção abordaram algumas iniciativas dos autores em mitigar vulnerabilidades em aplicações *Web*, com o propósito de torná-las mais seguras.

Tais trabalhos contribuíram para o reconhecimento dos desafios enfrentados com a necessidade de prover segurança em *software*. O foco desse artigo é apresentar metodologia MADS-WEB que contemple aspectos de segurança em todas as fases do ciclo de vida de um *software*, não contempladas nos trabalhos anteriores. Nesse sentido, o presente trabalho também procurou evidenciar outras falhas que não foram apresentadas nos trabalhos anteriores, além das já descobertas. Assim, são propostas formas de mitigação de vulnerabilidades em face às falhas que foram descobertas com a realização de técnicas de exploração de vulnerabilidades, haja vista não terem sido contempladas nos trabalhos anteriores.

O artigo está organizado como segue. A Seção 2 apresenta as principais vulnerabilidades conhecidas direcionadas a aplicações *Web*. A Seção 3 apresenta a metodologia aplicada ao desenvolvimento seguro de aplicações *Web*, assim como a arquitetura da metodologia proposta nesse trabalho, neste sentido é feita uma explicação das fases e a interação dessas fases entre si no modelo proposto. Por fim, a Seção 4 apresenta as conclusões e perspectivas futuras deste trabalho.

Vulnerabilidades em Aplicações Web

O *Open Web Application Security Project* (OWASP) é uma entidade sem fins lucrativos e de reconhecimento internacional, que contribui para a melhoria da segurança de softwares aplicativos reunindo informações importantes que permitem avaliar riscos de segurança e combater formas de ataques através da internet (OWASP, 2014).

De acordo com estudos recentes do OWASP é possível destacar o conjunto das dez vulnerabilidades mais exploradas em aplicações Web (HAROLD, 2010). A tabela 1 apresenta o nome das vulnerabilidades ou tipos de ataques, sua respectiva definição e de como pode ser explorada por um *hacker* ou uma pessoa maliciosa.

TABELA 1: Vulnerabilidades Conhecidas em Aplicações Web (Adaptado de [10]).

Casos de Abuso	Risco
Injeção de Código	Ocorrem quando dados não confiáveis são enviados para um interpretador como parte de um comando ou consulta. Os dados manipulados pelo atacante podem

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

	iludir o interpretador para que este execute comandos indesejados ou permita o acesso a dados não autorizados.
Quebra de Autenticação e Gerenciamento de Sessão	Ocorre quando as funções da aplicação relacionadas à autenticação e gerenciamento de sessão são implementadas de forma incorreta, permitindo que os atacantes comprometam senhas, chaves e *tokens* de sessão ou explorem outra falha da implantação para assumir a identidade de outros usuários.
Cross-Site Scripting (XSS)	Falhas XSS permitem aos atacantes executarem *scripts* no navegador da vítima.
Referência Insegura e Direta a Objetos	Ocorre quando um programador expõe uma referência à implantação interna de um objeto, como um arquivo, diretório, ou registro da base de dados.
Configuração Incorreta de Segurança	Uma boa segurança exige a definição de uma configuração segura em todos os níveis.
Exposição de Dados Sensíveis	Ocorre quando as aplicações não protegem devidamente os dados sensíveis, tais como cartões de crédito, IDs fiscais e credenciais de autenticação.
Falta de Função para Controle do Nível de Acesso	Ocorre quando as aplicações não verificam os direitos de acesso, em nível de função, antes de tornar essa funcionalidade visível na interface do usuário.
Cross-Site Request Forgery (CSRF)	Ocorre quando o navegador da vítima é forçado a executar uma ação maliciosa em favor do atacante.
Utilização de Componentes Vulneráveis Conhecidos	Componentes, tais como bibliotecas, *frameworks*, e outros módulos de *software* quase sempre são executados com privilégios elevados.
Redirecionamentos e Encaminhamentos Inválidos	Aplicações frequentemente redirecionam e encaminham usuários para outras páginas ou sites, e usam dados não confiáveis para determinar as páginas de destino.

Para verificar as vulnerabilidades, quando se trata de uma aplicação *Web*, todos os casos de abusos, relacionados na tabela 1, devem ser examinados.

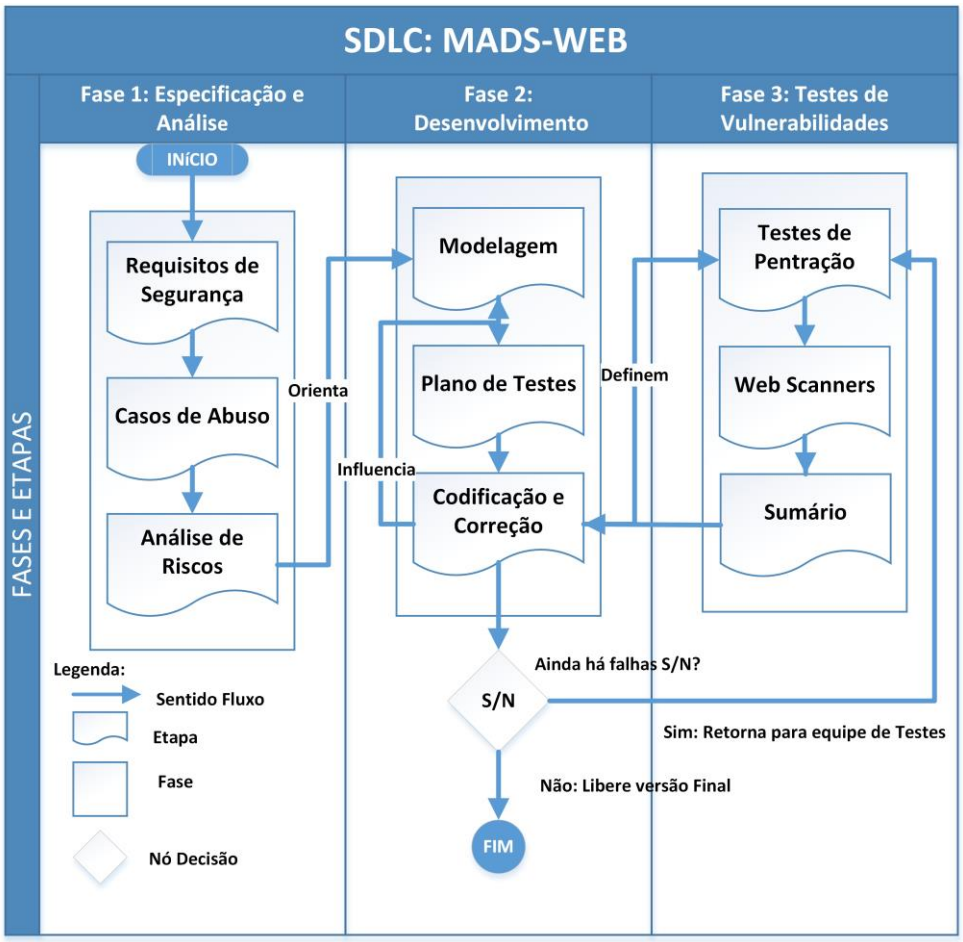
RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

MADS-WEB: Metodologia Aplicada ao Desenvolvimento Seguro de Aplicações *Web*

Embora a OWASP apresente as principais vulnerabilidades que uma aplicação *Web* está sujeita, a mesma não trata as fragilidades no desenvolvimento da aplicação *Web*. Assim, a metodologia MADS-WEB fornece um conjunto de fases e etapas que proporcionam aspectos de segurança no ciclo de desenvolvimento de uma aplicação *Web*. Assim sendo, a segurança passa a ser uma medida indispensável para o desenvolvimento de um sistema.

De forma mais sistemática, na figura 2 é apresentado o modelo MADS-WEB com as etapas de interação, compreendendo todas as suas fases e como se dá essa iteração entre as etapas e as atividades que fazem parte de cada fase.

Figura 2: Metodologia MADS-WEB.



Neste sentido, a MADS-WEB é composta por três fases: 1) Especificação e Análise; 2) Desenvolvimento; 3) Testes de Vulnerabilidades. A Primeira Fase está relacionada ao processo de início de construção do sistema, sendo responsável por estabelecer quais riscos à aplicação

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

está sujeita, com levantamentos dos Requisitos de Segurança e os Casos de Abuso. Ainda nesta fase é realizada a Análise de Risco que determinará quais riscos a aplicação está suscetível.

Na Segunda Fase será o momento em que a aplicação deverá ser desenvolvida e serão criados modelos que expresse como a aplicação se comportará, assim com o Plano de Testes e a Codificação.

Por fim, na Terceira Fase, os Testes de Vulnerabilidades deverão ser realizados com intuito identificar falhas que não foram mitigadas nas etapas anteriores quanto a aspectos de segurança, com a execução de ferramentas elencadas na etapa denominada de *Web Scanners* para realização dos Testes de Penetração. Logo em seguida os resultados serão confeccionados através de Sumários e repassados à equipe de Codificação e Correção na Segunda Fase. Antes de serem liberadas para usuário final, a equipe responsável pela Codificação e Correção, deverá realizar as correções, ou não caso elas não tenham sido detectadas.

3.1 ***Fase 1 - Especificação e Análise***

3.1.1 **Especificação e Análise**

Na primeira fase, Especificação e Análise, inicia-se o desenvolvimento do *software*. Essa fase engloba todas as tarefas que lidam com investigação, definição e escopo para construção de um sistema de atividades, como vulnerabilidades relacionadas à segurança e à confidencialidade que são pertinentes à aplicação. Elas devem ser identificadas para que sejam analisados as perdas e os impactos causados pelo comprometimento da confidencialidade das informações ou roubo das informações restritas a uma organização. Para isso, a primeira fase da MADS-WEB é composta por três etapas: Requisitos de Segurança, Casos de Abuso e Análise de Risco.

3.1.1.1 **Requisitos de Segurança**

A elicitação ou levantamento de requisitos é a fase inicial de qualquer projeto, sendo de extrema importância para seu sucesso, já que antes de iniciar qualquer ideia é preciso entender com clareza todos os objetivos e restrições envolvidos, não apenas para o correto planejamento, estudo de viabilidade ou estimativas de custos, mas para entregar ao usuário final um sistema que atenda às suas expectativas. Desta forma, torna-se fundamental para todos os projetos o entendimento e o controle correto desses requisitos para que os mesmos possam atender ao objetivo determinado.

Na MADS-WEB, o levantamento dos requisitos corresponde à definição dos Requisitos de Segurança que devem ser definidos explicitamente, e precisa corresponder aos objetivos e metas de segurança da organização. Quando os requisitos são apropriadamente definidos e

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

documentados, é possível mensurar o escopo de segurança do projeto, facilitando a implantação e a liberação do *software*.

Como forma de levantamento dos Requisitos de Segurança para aplicações *Web*, nesse trabalho é proposto o uso do *Common Criteria for Information Technology Security Evaluation*, ou *Common Criteria* que foi desenvolvido por diversos órgãos de países distintos, com o objetivo de avaliar a segurança da tecnologia da informação (ISO/IEC, 2009).

A metodologia MADS-WEB faz uso da segunda parte da norma em que são definidos os requisitos funcionais de segurança que devem ser levados em consideração na definição dos Requisitos de Segurança.

Dentro da norma, os requisitos funcionais de segurança são expressos em classes, famílias e componentes. As classes utilizadas na aplicação da metodologia são as seguintes: auditoria de segurança, comunicação, suporte à criptografia, proteção de dados do usuário, política de controle de acesso e identificação e autenticação.

A auditoria de segurança envolve o reconhecimento, gravação, armazenamento e análise de informações relacionadas com a segurança. Os registros de auditoria resultantes podem ser examinados para determinar quais atividades de segurança relevantes ocorreram e quem é o responsável por elas. As seguintes ações devem ser auditáveis e devem ser incluídas pelo menos uma, conforme prevê a norma:

a) Registro de ações que envolvam deleção, alteração e adição do grupo de usuários com permissão de acesso à leitura aos registros de auditoria;
b) Registros dos direitos para ver/modificar os eventos de auditoria;
c) Registros dos parâmetros que controlam o armazenamento de auditoria;
d) Registro de falhas devido a algum problema no armazenamento da auditoria;
e) Serviço de não repúdio;

Os requisitos de comunicação se preocupam essencialmente na troca de dados. Neste requisito o conceito de informação é usado devendo ser interpretado como objetos que estão sendo comunicados, assegurando a identidade do autor da informação transmitida (prova de origem) e a identidade do destinatário das informações transmitidas (comprovante de recebimento). Assim, o autor não pode negar ter enviado a mensagem, nem o destinatário negar o recebimento, ou seja, o não repúdio.

O suporte criptográfico deve fazer uso das funcionalidades de criptografia para satisfazer objetivos de segurança de alto nível. Estes incluem (mas não estão limitados a):

a) Identificação e autenticação;
b) Gerenciamento de segurança;
c) Privacidade;
d) Proteção das funcionalidades de segurança;
e) Utilização dos recursos de chaves públicas;
f) Uso de canais e caminhos confiáveis.

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

Os requisitos de proteção do usuário estão divididos em três partes:

a) Políticas de função do usuário de segurança de proteção de dados:
   1. Política de controle de acesso;
   2. Política de controle de fluxo de informação.
b) Formas de proteção de dados do usuário:
   1. Funções de controle de acesso;
   2. Funções de controle de fluxo de informação;
   3. Residual de proteção de informações;
   4. *Rollback*;
   5. Integridade dos dados armazenados.
c) Armazenamento *off-line*, importação e exportação:
   1. Autenticação de dados;
   2. Exportação dos dados.

A política de controle de acesso define o âmbito do controle das políticas que formam a parte de controle de acesso identificado, gerenciando os atributos usados para tornar o acesso explícito ou negação com base em políticas de segurança.

A identificação e autenticação definem requisitos para funções que estabelecem e verificam a identidade de um usuário em questão e devem ser capaz de:

1. Falhas de Autenticação: definir valores para tentativas sem sucesso de autenticação e ações;
2. Atributos de Usuário: definir requisitos para vinculação de atributos de segurança válidos;
3. Especificação de Senhas: definir requisitos para mecanismos que definem a quantidade da métrica de senhas e formas de satisfazer essa métrica;
4. Identificação do Usuário: define os tipos de autenticação suportados pelos usuários.

3.1.1.2 Casos de Abuso

Após a análise e definição dos Requisitos de Segurança, ou seja, aqueles que especificam os requisitos mínimos de segurança que uma aplicação *Web* deve prover, devem ser elaborados os Casos de Abuso. Os Casos de Abuso ligam vulnerabilidades conhecidas aos riscos que a aplicação possa ser submetida. Um Caso de Abuso é definido a partir de quais explorações uma aplicação pode sofrer, tendo em vista os riscos que ela possui.

Como forma de análise das vulnerabilidades existentes que envolvem aplicações *Web*, a metodologia MADS-WEB faz uso do OWASP *TOP* 10 para definir os riscos pertinentes que uma aplicação *Web* pode sofrer. Todas as 10 (dez) vulnerabilidades elencadas devem ser avaliadas na Terceira Fase, onde serão feitos os Testes de Vulnerabilidades, conforme define a metodologia em questão. Em seguida, a Análise de Risco determinará os riscos aos quais a aplicação é suscetível.

3.1.1.3 Análise de Risco

A terceira etapa da fase de Especificação e Análise, denominada Análise de Risco, envolve identificar perdas e impactos causados pelo comprometimento da confidencialidade das

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

informações ou roubo das informações restritas a uma organização. Os riscos podem estar relacionados à distribuição de dados de importância crítica e à perda de integridade das informações. Essa etapa da metodologia estará diretamente ligada à modelagem do sistema, pois a partir desta etapa ele será definido ou modelado de acordo com as necessidades de segurança para cada organização.

Na MADS-WEB, a Análise de Risco é formada por quatro etapas, seguindo a proposta de (PRESSMAN, 2006):

- Etapa 1 - Identificação dos riscos: a identificação dos riscos inclui determinar quais os riscos podem afetar o projeto e documentar suas características;
- Etapa 2 - Projeção/Estimativas dos riscos: a projeção ou estimativa dos riscos busca classificar cada ameaça, com a probabilidade dela acontecer, prevendo, assim, as consequências do seu acontecimento;
- Etapa 3 - Administração dos riscos: desenvolvimento de opções e ações para aumentar as oportunidades e reduzir as vulnerabilidades encontradas na aplicação;
- Etapa 4 - Monitoramento dos Riscos: acompanhamento dos riscos identificados, monitoramento dos riscos residuais, identificação dos novos riscos, execução de planos de respostas a riscos e avaliação da sua eficácia durante todo o ciclo de vida da aplicação.

Quando uma organização trabalha com uma infraestrutura na internet, se faz necessário tomar uma decisão de como os riscos devem ser tratados. É praticamente impossível eliminar todas as ameaças pelas quais uma organização pode passar. Neste sentido, faz-se necessário mitigar os riscos aos quais a aplicação está suscetível. Logo em seguida a Análise de Risco, inicia-se a fase de Desenvolvimento.

3.2 ***Fase 2 – Desenvolvimento***

3.2.1 **Desenvolvimento**

A Segunda Fase, Desenvolvimento, é o momento onde parte das questões levantadas na fase de Especificação e Análise devem ser avaliadas e mitigadas. A Análise de Riscos, Requisitos de Segurança e os Casos de Abusos levantados na primeira fase, influenciam a etapa de Modelagem, assim como o Plano de Testes. Esta etapa, também chamada de implantação, é a mais central do processo de engenharia de software, pois é quando o *software* é efetivamente construído. Para isso, a Segunda Fase da MADS-WEB também é composta por três etapas: Modelagem, Plano de Testes e Codificação e Correção.

3.2.1.1 **Modelagem**

Um modelo é uma simplificação da realidade, ele é criado para facilitar o entendimento de sistemas complexos. Estes modelos podem abranger planos detalhados e também planos mais gerais com uma visão panorâmica do sistema (PRESSMAN, 2006).

Todos os sistemas podem ser descritos sob diferentes aspectos, com a utilização de modelos distintos, onde cada modelo será, portanto, uma abstração específica do sistema. Os modelos podem ser estruturais, dando ênfase à organização do sistema, ou podem ser comportamentais,

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

dando ênfase à dinâmica do sistema. Há quatro objetivos principais para se criar modelos (SOMMERVILLE, 2011):

1. Ajudam a visualizar o sistema como ele é ou como deseja que ele seja;
2. Permitem especificar a estrutura ou o comportamento de um sistema;
3. Proporcionam um guia para a análise do sistema;
4. Documentam as decisões tomadas no projeto.

Através dos modelos, é possível obter múltiplas visões do sistema, particionando a complexidade do sistema para facilitar sua compreensão, e atuando como meio de comunicação entre os participantes do projeto. Portanto, uma linguagem de modelagem padronizada, tal como a UML[1], é fundamental para a construção e o entendimento de bons modelos.

No contexto da Modelagem MADS-WEB uma das múltiplas visões a serem oferecidas é a modelagem de ameaças. Para isso, o primeiro passo do processo de modelo de ameaças da metodologia é desenvolver uma representação visual das ameaças sob a forma de um diagrama de fluxo[2].

Para evitar erros é importante fornecer um modelo estruturado para modelagem, mas é importante compreender que esta etapa representa o fluxo de dados e não a codificação. Este é um erro muito comum por desenvolvedores, porque eles só pensam em escrever, sem antes modelar para primeiro entender a complexidade do *software* e as suas ameaças (RANSOME, 2013). Os principais passos envolvidos na modelagem de ameaças são:

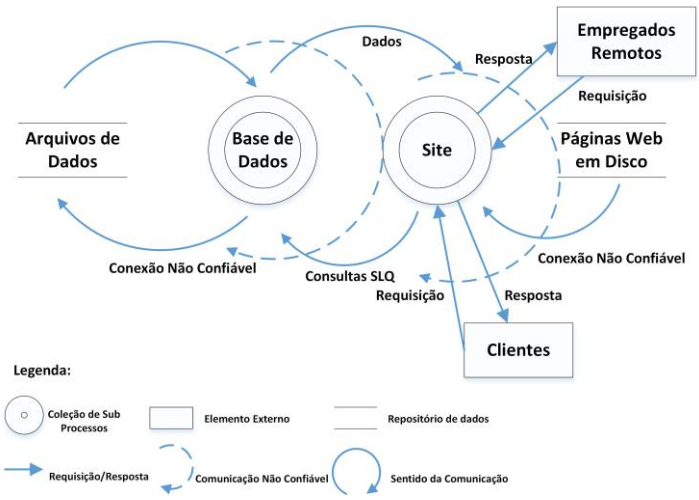
1. Dividir a arquitetura de *software* utilizando diagramas de fluxo de dados;
2. Criar categorias com ameaças, identificando quais ameaças são aplicáveis para cada elemento em seu diagrama de fluxo de dados;
3. Mapear todas as ameaças com vulnerabilidades pertinentes aplicáveis no contexto do cenário de uso;
4. Criar um *ranking* de ameaças atribuindo uma classificação de risco para cada ameaça e vulnerabilidade de forma a compreender o impacto e definir prioridades;
5. Definir um plano de mitigação com contramedidas para cada uma das vulnerabilidades identificadas;
6. Corrigir as vulnerabilidades que não são aceitáveis para o negócio.

O Diagrama de Fluxo de Dados (DFD) na figura 3 apresenta um modelo de ameaças ao qual uma aplicação Web está suscetível. O DFD proporciona uma visão mais ampla das ameaças. Neste sentido, não é possível fornecer um modelo padrão na MADS-WEB para todas as ameaças, visto que cada aplicação tem suas particularidades e as ameaças que estão suscetíveis dependem das funcionalidades que a mesma possui.

Figura 3: Exemplo de Diagrama de Fluxo de Dados para a Modelagem de Ameaças

[1] Permite que desenvolvedores visualizem os produtos de seus trabalhos em diagramas padronizados.
[2] É uma representação gráfica do "fluxo" de dados através de um sistema de informação, modelando seus aspectos de processo.

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218



FONTE: RANSOME, 2013

Criar um DFD é a chave para obter um modelo de ameaça. É importante garantir que todas as peças do sistema estejam representadas. Cada um dos elementos (processos, arquivos de dados, fluxos de dados e usuários), conforme demonstrado na figura 3, ela modela as ameaças que são pertinentes ao *software* que se deseja desenvolver. Uma vez que o DFD seja concluído, será proporcionada uma visão geral de como os dados são processados pelo *software*, incluindo a forma como ele se move. É o que acontece dentro do aplicativo e outros que podem estar associados ao *software*.

Na MADS-WEB a modelagem pode ser modificada, caso seja necessário realizar alguma correção no código, devido à descoberta de algum problema, por exemplo, através da realização dos Testes de Penetração poderão ser detectadas falhas em face aos resultados contidos nos sumários, tornando-se necessário que se realize a remodelação da aplicação para atender às correções.

3.2.1.2 Plano de Testes

A realização do Plano de Testes é uma das atividades do processo de desenvolvimento de sistema de *software* que visa executar um conjunto de ações de modo sistemático com o objetivo de encontrar falhas no processo de desenvolvimento do software.

O Plano de Teste é o conjunto de tarefas que são realizadas no processo de construção de uma aplicação. Ele proporciona a realização de testes que por sua vez validam a segurança de uma aplicação, ao mesmo tempo em que reduz a probabilidade de erros quanto à segurança sobre o produto que está sendo desenvolvido, antes que as falhas de segurança sejam descobertas por clientes ou usuários mal-intencionados.

Neste sentido, se faz necessária a realização de testes em face aos Requisitos de Segurança pertinentes à aplicação a qual a mesma está suscetível, conforme levantamento realizado na

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

etapa de Análise de Risco na primeira fase da metodologia. Exemplos de requisitos a serem testados são: desempenho; segurança; interface de usuário; controle de acesso e funcionalidades. Esses requisitos procuram garantir a competência da aplicação a partir de uma perspectiva de segurança, demonstrada através dos testes e seus artefatos, relatórios e ferramentas.

O objetivo da realização de testes não é testar a insegurança, mas sim, validar a robustez e a segurança da aplicação antes de tornar o produto disponível para os clientes.

O Plano de Teste pode ser elaborado pelo gerente de projeto ou gerente de testes. Esse plano visa planejar as atividades a ser realizadas, definir os métodos a ser empregados, planejar a capacidade necessária, estabelecer métricas e formas de acompanhamento do processo. Nesse sentido, ele deve conter:

1. Os itens a serem testados: o escopo e objetivos do plano devem ser estabelecidos no início do projeto;
2. Atividades e recursos a serem empregados: as estratégias de testes e recursos utilizados devem ser definidas, bem como toda e qualquer restrição imposta sobre as atividades e/ou recursos;
3. Os tipos de testes a serem realizados e ferramentas empregadas: os tipos de testes e a ordem cronológica de sua ocorrência são estabelecidos no plano;
4. Critérios para avaliar os resultados obtidos: métricas devem ser definidas para acompanhar os resultados alcançados.

O planejamento é necessário a fim de antecipar o que pode ocorrer e, portanto, provisionar os recursos necessários nos momentos adequados. Isto significa coordenar o processo de teste de modo a perseguir a meta de qualidade do produto (sistema de *software*).

A tabela 2 apresenta uma relação dos itens considerados imprescindíveis para um Plano de Teste.

TABELA 2. Relação de Itens de um Plano de Teste.

Itens de um Plano de Teste	Conteúdo
1. Introdução	Contém uma identificação do projeto, descrição dos objetivos do documento, o público ao qual ele se destina e escopo do projeto a ser desenvolvido. Pode adicionalmente conter termos e abreviações usadas, além de informar como o plano deve evoluir.
2. Requisitos a serem testados	Descreve em linhas gerais o conjunto de requisitos a serem testados no projeto a ser desenvolvido, comunicando o que deve ser verificado.
3. Estratégias e ferramentas de teste	Apresenta um conjunto de tipos de testes a serem realizados, respectivas técnicas empregadas e critério de finalização de teste. Além disso, é listado o conjunto de ferramentas utilizadas.

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

4. Equipe e infraestrutura	Contém descrição da equipe e da infraestrutura utilizada para o desenvolvimento das atividades de testes, incluindo: pessoal, equipamentos, software de apoio, materiais, dentre outros. Visa garantir uma estrutura adequada para a execução das atividades de testes previstas no plano.
5. Cronograma de atividades	Contém uma descrição de marcos importantes das atividades (incluindo as datas de início e fim da atividade).
6. Documentação complementar	Apresenta-se uma relação dos documentos pertinentes ao projeto.

O Plano de Testes é imprescindível no processo de construção de um sistema, sendo que na metodologia MADS-WEB os testes têm como propósito a validação das funcionalidades pertinentes à aplicação. Não sendo dispensada a realização dos Testes de Penetração a serem realizados na Terceira Fase da metodologia, que tem como objetivo avaliar se todas as medidas de segurança realizadas foram cumpridas e se os riscos foram mitigados, evidenciando os problemas, caso existam.

3.2.1.3 Codificação

Na etapa da Codificação e Correção uma equipe de programadores é designada para trabalhar no *software*. Diagramas gerados na etapa de Modelagem são passados para equipe de desenvolvimento que irá transformá-los em código de uma linguagem de programação qualquer. Normalmente, as pessoas envolvidas nesse processo são divididas em dois grupos, um responsável pela implantação da interface gráfica e outro pela implantação da lógica da aplicação (RANSOME, 2013). Além disso, boa parte da estrutura básica da aplicação pode ser exportada diretamente dos diagramas em código, dependendo das ferramentas que estiverem sendo usadas pela equipe.

Na metodologia MADS-WEB não há como definir a especificação de como deverá ser realizada a codificação, visto que dependendo das funcionalidades pertinentes da aplicação a ser desenvolvida, poderá contar com uma ou mais linguagens de programação diferentes, de modo que cada aplicação possui particularidades que somente a equipe de programação definirá após a modelagem do sistema.

Portanto, a escolha fica a cargo da equipe que realizará a codificação do sistema. Ainda neste sentido, é importante ressaltar que o domínio da linguagem escolhida é um fator determinante para o sucesso da aplicação quanto à segurança do sistema, quanto mais domínio sobre uma determinada linguagem a equipe possuir, melhor será desenvolvido o *software*. Problemas como estouro de pilhas, inteiros e formato de *strings* são vulnerabilidades extremamente sérias para programas escritos em linguagem C ou C++ (RANSOME, 2013).

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

Na metodologia MADS-WEB como medida preventiva algumas recomendações são sugeridas como práticas gerais de codificação:

- Utilizar sempre código testado, gerenciado e aprovado em vez de criar código novo, não gerenciado, para tarefas comuns;
- Utilizar *Application Programming Interface* ou Interface de Programação de Aplicações (API) que insiram tarefas específicas para ser realizadas através do sistema operacional. Não permitir que a aplicação execute comandos diretamente no sistema operacional, especialmente através da utilização de *shells* de comando iniciados pela aplicação;
- Fazer uso de mecanismo de verificação de integridade por *checksum* ou *hash* para verificar a integridade do código interpretado, bibliotecas, arquivos executáveis e arquivos de configuração;
- Evitar requisições simultâneas para a aplicação ou utilizar um mecanismo de sincronização para evitar condições de disputa (*race conditions*), fazendo uso de mecanismos de *lock*;
- Proteger as variáveis compartilhadas e recursos contra acessos concorrentes inapropriados;
- Restringir os usuários de gerar um novo código ou alterar o código existente;
- Revisar todas as aplicações secundárias, códigos e bibliotecas de terceiros para determinar a necessidade do negócio e validar as funcionalidades de segurança, uma vez que estas podem introduzir novas vulnerabilidades;
- Identificar todas as fontes de dados e classificar as fontes como confiável/não confiável. Em seguida, validar os dados provenientes de fontes não confiáveis;
- As rotinas de validação de dados de entrada devem ser centralizadas na aplicação;
- Especificar um conjunto de caracteres apropriados, como UTF-8, para todas as fontes de entrada de dados;
- Codificar os dados para um conjunto de caracteres comuns antes da validação;
- Quando há falha de validação a aplicação deve rejeitar os dados fornecidos;
- Determinar se o sistema suporta conjuntos de caracteres estendidos UTF-8 e, em caso afirmativo, validar após efetuar a decodificação UTF-8;
- Validar todos os dados provenientes dos clientes antes do processamento, incluindo todos os parâmetros, campos de formulário, conteúdo das URLs e cabeçalhos HTTP, por exemplo: nomes e valores dos *cookies*. Certificar-se também de incluir automaticamente mecanismos de *postback*[3] nos trechos de código *JavaScript*, *Flash* ou qualquer outro código incorporado.

Após o *software* ser codificado deverá ser disponibilizada uma versão do sistema para que na Terceira Fase a equipe responsável pelos Testes de Penetração possa realizar análise de vulnerabilidade com base nos Casos de Abuso reportados na Primeira Fase.

Assim, a equipe de desenvolvimento de posse dos Sumários dos Testes de Penetração deverá realizar as correções, caso tenham sido sugeridas, retornando para que sejam realizados novos

[3] Medida tomada por uma página interativa, quando a página inteira e seus conteúdos são enviados para o servidor para processamento de algumas informações e, em seguida, o servidor mostra a mesma página de volta ao seu navegador.

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

testes a fim de descobrir se a aplicação ainda continua com a vulnerabilidade, caso não haja mais problema, deve-se liberar a versão final para usuário.

A correção como atividade incluída nesta etapa se faz necessária à medida que a equipe de responsáveis pelos Testes de Penetração reporta os erros através dos sumários, logo em seguida, a equipe de desenvolvimento deve realizar as correções necessárias.

Neste sentido, como citado nas subseções anteriores, caso as correções impliquem em alterações críticas do ponto de vista da equipe, deverá ser realizada a remodelação da aplicação, necessitando que a equipe responsável realize as alterações no modelo para que logo em seguida possam ser realizadas as correções e os novos Testes de Penetração.

3.3 ***Fase 3 – Desenvolvimento***

3.3.1 **Testes de Vulnerabilidade**

A Terceira Fase se torna essencial para avaliar se a aplicação contém vulnerabilidades, pois serão realizados Testes de Vulnerabilidades com o intuito de identificar quais falhas a aplicação possui. Essa fase é composta por três etapas: Teste de Penetração, *Web Scanners* e Sumário. A partir dos Casos de Abusos, levantados na Primeira Fase, serão definidos os Testes de Penetração que serão utilizados para burlar o sistema. Após definição, os Testes de Penetração serão efetivamente aplicados através de *Web Scanners*. Com o resultado da análise dos *Web Scanners*, serão desenvolvidos sumários com o resultado dos testes.

3.3.1.1 **Teste de Penetração**

Os Testes de Penetração são tentativas legais e autorizadas de localizar e explorar sistemas de computadores de forma bem-sucedida com o intuito de tornar esses sistemas mais seguros. Esse processo inclui analisar as vulnerabilidades, bem como oferecer ataques que funcionem como prova de conceito para demonstrar que eles são reais. Os Testes de Penetração adequados sempre terminam com recomendações específicas para endereçar e corrigir os problemas descobertos durante o teste.

A ideia geral consiste em identificar problemas de segurança usando as mesmas ferramentas e técnicas usadas por um invasor. É possível então atenuar os riscos identificados por essas descobertas antes que um *hacker* de verdade as explore (ENGEBRETSON, 2013).

A MADS-WEB faz uso da metodologia para técnicas de penetração que segundo (ENGEBRETSON, 2013), é dividida em quatro fases: Reconhecimento, *Scanning*, Exploração de falhas (*exploitation*) e Pós-Exploração (ou Preservação do Acesso). Compreender a sequência adequada em que esses passos são executados é fundamental para realizar um teste de penetração abrangente e realista. A figura 4 apresenta essas atividades.

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

Figura 4: A metodologia Hacking para Testes de Penetração.

FONTE: ENGEBRETSON, 2013

O primeiro passo em qualquer Teste de Penetração é o reconhecimento considerando a atividade que cuida da coleta de informações sobre o alvo. Quanto mais informações coletar sobre o seu alvo, mais possibilidade haverá de ser bem-sucedido nos passos subsequentes.

O segundo passo nessa metodologia pode ser dividido em duas atividades distintas. A primeira atividade a ser realizada é o *scanning* de portas. Após concluir o *scanning* de portas, tem-se uma lista de portas abertas e de serviços em potencial sendo executados em cada um dos alvos. A segunda atividade da fase de *scanning* é o *scanning* de vulnerabilidades que corresponde ao processo de localizar e de identificar pontos fracos específicos nos *softwares* e nos serviços presentes no alvo.

Com os resultados obtidos no passo 2, o próximo passo será a fase de exploração de falhas (*exploitation*). Depois da descoberta exata de que as portas estão abertas, informar quais serviços estão executando nessas portas e quais vulnerabilidades estão associadas a esses serviços. Assim, a equipe responsável pelos Testes de Penetração pode começar a atacar o alvo. A exploração pode envolver diversas técnicas, *Web scanners* e códigos diferentes. O objetivo final da exploração consiste em identificar brechas de segurança.

A fase final a ser analisada é a de pós-exploração e preservação do acesso. Com frequência, os *payloads*[4] enviados na fase de exploração de falhas permitem apenas um acesso temporário ao sistema. Como a maior parte dos *payloads* não é persistente deve-se prosseguir rapidamente

[4] É a parte dos dados transmitidos, que é o objetivo fundamental da transmissão, excluindo as informações enviadas com ela (como cabeçalhos ou *metadados*, também conhecido como dados complementares, que podem conter, dentre outras informações, a identificação da fonte e do destino dos dados) apenas para facilitar a entrega.

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

para a fase de pós-exploração para criar uma porta dos fundos (*backdoor*[5]) mais permanente para o sistema. Esse processo permite que o acesso de administrador sobreviva quando os programas forem encerrados e até mesmo quando houver uma reinicialização do sistema.

Na MADS-WEB a etapa de pós-exploração se torna desnecessária devido não haver necessidade de manter acesso para futuras invasões, como sugere a própria metodologia de penetração, o propósito é apenas descobrir a quais falhas a aplicação está suscetível.

Neste sentido, se faz necessário o uso de ferramentas específicas para realização dos Testes de Penetração que são denominadas *Web scanners*.

3.3.1.2 Web Scanners

São conjuntos de ferramentas capazes de detectar se uma aplicação contém vulnerabilidades. Estas ferramentas têm como objetivo facilitar e automatizar a busca por vulnerabilidades em uma aplicação, bem como mapear a estrutura do site, ou seja, possibilitam ter uma visão completa do estado de segurança da aplicação e ajudam a automatizar os testes, pois fazem uma varredura no site de destino que se pretende realizar análise quanto à segurança.

As ferramentas propostas na MADS-WEB para realização de Análise de Vulnerabilidade são todas de código fonte aberto e foram utilizadas como forma de realização de Testes de Penetração em uma aplicação Web, tais ferramentas fazem parte do *Kali Linux*[6]. A primeira coluna da tabela 3 apresenta o nome das ferramentas que foram utilizadas. A segunda coluna apresenta a sua respectiva descrição e o propósito de cada uma na realização de um Teste de Penetração.

TABELA 3. Apresenta as Ferramentas Utilizadas para Realização de Testes de Penetração.

Ferramenta	Descrição
Metasploit Framework (MSF)	Tem como objetivo realizar análise de vulnerabilidades de segurança e facilitar testes de penetração. Utilizada nos testes de Quebra de Autenticação.
Nikto	Trata-se de um scanner que realiza testes abrangentes contra servidores Web, incluindo mais de 6.700 arquivos potencialmente perigosos. Também verifica quais são os itens de configuração do servidor, tais como a presença de arquivos de índice, opções do servidor HTTP, tentando identificar os servidores Web instalados e quais *softwares* fazem parte. Itens de digitalização e *plugins* são atualizados com frequência e podem ser atualizados automaticamente. Utilizada nos testes de Referência Insegura e Direta a Objetos.

[5] *Backdoor* é um recurso utilizado por diversos *malwares* para garantir acesso remoto ao sistema ou à rede infectada, explorando falhas críticas não documentadas existentes em programas instalados, *softwares* desatualizados e do *firewall* para abrir portas do roteador.
[6] *Kali Linux* é uma distribuição Linux baseada em Debian para realização de testes de penetração.

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

Web Application Attack and Audit Framework (W3af)	É um *scanner* de segurança de aplicações Web de código aberto. Fornece informações sobre vulnerabilidades de segurança e ajuda no esforço de teste de penetração. Utilizada nos testes de Injeção de Código *Sql.*
Spidering	São principalmente utilizados para criar uma cópia de todas as páginas visitadas para um pós-processamento por um motor de busca que irá indexar as páginas baixadas para prover buscas mais rápidas. *Crawlers* também podem ser usados para tarefas de manutenção automatizadas em um *website*, como checar os links ou validar o código HTML. Utilizada nos testes de Utilização de Componentes Vulneráveis Conhecidos.
WebScarab	Ferramenta de teste de aplicações de segurança Web. Ele serve como um *proxy* que intercepta e permite que as pessoas alterem pedidos feitos a navegadores Web (HTTP e HTTPS) e respostas dos servidores Web. Utilizada nos testes de *Cross-Site Request forgery (CSRF).*
OWASP-ZAP	É uma ferramenta utilizada em testes de penetração desenvolvida para encontrar vulnerabilidades em aplicações *Web.* Fornece *scanners* automatizados, bem como um conjunto de ferramentas que permitem encontrar vulnerabilidades de segurança manualmente. Utilizada nos testes Redirecionamentos e Encaminhamentos Inválidos, Configurações Incorretas de Segurança, Exposição de Dados Sensíveis, Falta de Função para Controle do Nível de Acesso e *Cross-Site Scripting* (XSS).

3.3.1.3 Sumários

Refere-se ao documento formal onde devem constar todos os testes feitos e as variáveis que foram levadas em consideração, assim como os resultados e as soluções propostas para que a equipe responsável por fazer as correções do código possa utilizar como referência, como forma de agilizar as correções a serem feitas.

O propósito do sumário é oferecer uma visão geral simples, não técnica, de uma a duas páginas, relativas às descobertas. Esse relatório deve ressaltar e sintetizar os problemas mais graves descobertos pela equipe de teste. É de suma importância que ele possa ser lido e compreendido tanto pelo pessoal técnico quanto pelo não técnico. No relatório deve conter as medidas para mitigar os problemas encontrados e as soluções devem ser apresentadas para a equipe de desenvolvimento. Exemplos de sumários são demonstrados nos apêndices desse trabalho, onde constam resultados dos testes de penetração realizados em uma aplicação *Web*.

4 CONCLUSÕES E TRABALHOS FUTUROS

Nos últimos anos, houve mais ênfase no desenvolvimento de *softwares* seguros e, como consequência, as aplicações *Web* atuais são muito mais seguras do que as versões anteriores. Houve uma forte pressão no sentido de incluir a segurança nos estágios iniciais do ciclo de vida

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

do desenvolvimento de *software* e de formalizar a especificação dos Requisitos de Segurança de forma padronizada. Também houve um aumento enorme na organização de diversas comunidades dedicadas à segurança de aplicações Web. Empresas de grande porte como a Microsoft, IBM *e* Google, são extremamente preocupadas com segurança, visto que suas aplicações são utilizadas por pessoas em todo mundo, podendo colocar em risco sua reputação, caso falhas sejam descobertas e exploradas.

Entre as inúmeras vulnerabilidades, estudos recentes realizados pela (OWASP, 2014) indicam as dez mais recorrentes e mais exploradas pelos invasores. Muitas destas vulnerabilidades identificadas pela simples realização de atividades de testes de penetração, como é sugerido na fase três da metodologia apresentada neste artigo MADS-WEB.

A implantação de um *software* seguro exige a observância das características de segurança já nas fases iniciais, em conjunto com o entendimento do processo de negócio, ao invés de adicionar segurança apenas no final do ciclo de vida.

Neste sentido, a metodologia apresentada neste trabalho favorece uma integração forte entre a equipe que faz levantamento dos Requisitos de Segurança, Casos de Abuso e Análise de Risco, na fase um, como a equipe de Modelagem, Plano de Testes e Codificação, na fase dois e a equipe de Testes de Penetração que engloba *Web scanners* e os Sumários, na fase três, visto que toda equipe está envolvida com aspectos quanto à segurança do sistema em todo seu ciclo de vida. De modo que, beneficia as equipes de desenvolvimentos, inserindo elementos de segurança no ciclo de vida do *software*.

De forma geral, este trabalho contribui para que equipes de desenvolvimento possam fazer uso da metodologia apresentada neste trabalho para a construção de aplicações mais seguras em função da utilização de aspectos de segurança no ciclo de vida de desenvolvimento de um *software*. O que torna fundamental o abandono da cultura de que segurança é algo a ser acoplado ao final do projeto de desenvolvimento de software. No que diz respeito aos trabalhos futuros, sugere-se:

1. A utilização da metodologia em outros projetos de *software* para desenvolvimento de sistemas e estabelecimento de métricas de tempo, custo e retorno do investimento;
2. O estabelecimento de um modelo de maturidade de segurança de *software* voltado à construção do *software*;
3. O estabelecimento de um modelo de levantamento de requisitos de segurança de software;
4. Desenvolvimento de um *framework* especializado que realize os testes de vulnerabilidade de forma dinâmica em uma única aplicação.

referências bibliográficas

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

ASSOSIAÇÃO BRASILEIRA DE NORMA TECNICAS. **ABNT ISO/IEC 17799/2005**: Tecnologia da informação – Técnicas de segurança – Código de prática para a gestão da segurança da informação. Rio de Janeiro, 2005.

ASSOSIAÇÃO BRASILEIRA DE NORMA TECNICAS. **ABNT NBR ISO/IEC 27001:2013**: Tecnologia da informação — Técnicas de segurança — Sistemas de gestão da segurança da informação — Requisitos. Rio de Janeiro, 2013.

ASSOSIAÇÃO BRASILEIRA DE NORMA TECNICAS. **ABNT NBR ISO/IEC 27002:2013**: Tecnologia da informação — Técnicas de segurança — Código de prática para controles de segurança da informação. Rio de Janeiro, 2013.

ACUNETIX. **Web Vulnerability**, 2014. Disponível em http://www.acunetix.com. Acesso em: 24 ago. 2014.

AL-AJLAN, A. S. A comparative study between e-learning features, methodologies, tools and new developments for e-learning, dr. elvis pontes (ed.). *In.*: **Methodologies, Tools and New Developments for E-Learning. INTECH**, 2012. p. 25. Qassim University Kingdom of Saudi Arabia. Disponível em: http://www.intechopen.com/books/methodologies-tools-and-new-developments-for-e-learning/a-comparative-study-between-e-learning-features. Acesso em: 15 set. 2014.

AOKI, Eric Komiyama; CARVALHO, Alan Henrique Pardo. Práticas de segurança para o desenvolvimento de sistemas Web. Fasci-Tech – **Periódico Eletrônico da FATEC-São Caetano do Sul**, São Caetano do Sul, v. 1, n. 5, p. 56 a 66, Out/Dez. 2011.

CERT.BR - **Centro de Estudos, Respostas e Tratamentos de Incidentes de Segurança no Brasil**. 2014. Disponível em: http://www.cert.br/stats/incidentes/2013-jan-dec/tipos-ataque.html. Acesso em: 15 set. 2014.

ENGEBRETSON, Patrick. **Basics of hacking and penetration testing, the ethical hacking and penetration testing made easy**. Syngress, 2013.

FLOYD, Colton. Schultz, Tyler And Fulton, Steven. Security Vulnerabilities in the open source Moodle eLearning System. *In.*: **Proceedings of the 16th Colloquium for Information Systems Security Education**. Lake Buena Vista, Florida June 11-13, 2012.

HAROLD, F. **Official (Isc)2 Guide to the SSCP Cbk**. 2nd ed. Boston, MA, USA: Auerbach Publications, 2010.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO/IEC 15408-1:2009**: Information technology - Security techniques - Evaluation criteria for IT security - Part 1: Introduction and general model. Geneva, 2009.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO/IEC 15408-1:2009** Information technology - Security techniques - Evaluation criteria for IT security - Part 2: Introduction and general model. Geneva, 2009.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO/IEC 21827/2008:** Systems Security Engineering - Capability Maturity Model. Geneva, 2008.

JACOBS, Stuart. **Engineering information security**: The application of systems engineering concepts to achieve information assurance. Wiley-IEEE Press, 2011. ISBN 978-0-470-56512-4 (hardback).

KARAPANOS, Nikolaos. CAPKUN, Srdjan. On the effective prevention of TLS man-in-the-middle attacks in web applications. *In.*: **Proceedings of the 23rd USENIX conference on Security Symposium (SEC'14).** Berkeley, CA, USA: USENIX Association. p. 671-686.

KUMAR, S.; GANKOTIYA, Ak.; DUTTA, K., "A comparative study of moodle with other e-learning systems," Electronics Computer Technology (ICECT), 2011. *In.*: **3rd International Conference on**, v. 5, p. 414-418, 8-10 April 2011. DOI: 10.1109/ICECTECH.2011.5942032 Disponivel em: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5942032&isnumber=5941942 Acesso em: 30 dez. 2014.

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

OWASP. **Top Ten – 2013 The Ten Most Critical Web Application Security Risks**. Disponível em: https://www.owasp.org/index.php/Top_10_2013-Top_10. Acesso em: 30 dez. 2014.

PONTES, Elvis; SILVA, Anderson; GUELFI, Adilson; KOFUJI, Takeo. (Org.). **Methodologies, Tools and New Developments for E-Learning**. Croatia: InTech Janeza Trdine 9, 51000 Rijeka. ISBN 978-953-51-0029-4.

PRESSMAN, Roger S. **Engenharia de Software**. 6. Ed. São Paulo: Ed. McGrawHill, 2006.

RANSOME, James; MISRA, Anmol. **Core Software Security: Security at the Source**. Boston, MA, USA: Auerbach Publications, 2013.

SOMMERVILLE, Ian. **Software engineering**. 9. ed. São Paulo: Pearson, 2011.

TSOUTSOS, Nektarios Georgios; MANIATAKOS, Michail. Trust No One: Thwarting "heartbleed" Attacks Using Privacy-Preserving Computation. *In.*: **Proceedings of the 2014 IEEE Computer Society Annual Symposium on VLSI (ISVLSI '14)**. Washington, DC, USA: IEEE Computer Society. p. 59-64. Disponivel em: DOI: http://dx.doi.org/10.1109/ISVLSI.2014.86 Acesso em: 30 dez. 2014.

VIANA, Sidney; SILVA, Richard F. Centro, Judith Pavón. Laine, Jean M. Segurança no Desenvolvimento de Aplicações Web com a Qualidade dos Dados. **Revista de Sistemas e Computação**, Salvador, v. 3, n. 2, p. 93-104, jul./dez. 2013. Disponivel em: http://www.revistas.unifacs.br/index.php/rsc/article/view/2745 Acesso em: 30 dez. 2014.