UNIVERSIDADE FUMEC

ANÁLISE DE EFICIÊNCIA NA DETECÇÃO DE VULNERABILIDADES EM AMBIENTES WEB COM O USO DE FERRAMENTAS *OPEN SOURCE*

MARCOS FLÁVIO ARAÚJO ASSUNÇÃO

Belo Horizonte - MG

2014

MARCOS FLÁVIO ARAÚJO ASSUNÇÃO

ANÁLISE DE EFICIÊNCIA NA DETECÇÃO DE VULNERABILIDADES EM AMBIENTES WEB COM O USO DE FERRAMENTAS *OPEN SOURCE*

Projeto de pesquisa apresentado ao Curso de Mestrado em Sistemas de Informação e Gestão da Informação da Universidade Fumec como parte dos requisitos para a obtenção do título de Mestre em Sistemas de Informação.

Orientador: Prof. Rodrigo Moreno Marques

Coorientador: Prof. Luiz Cláudio Gomes Maia

Belo Horizonte – MG

2014

RESUMO

As aplicações web vêm se popularizando cada vez mais com a expansão da Internet, oferecendo soluções a usuários e instituições, sejam do setor público ou privado. Entretanto, nem sempre a segurança acompanha a evolução dessas aplicações o que pode levar ao comprometimento de um sistema web, causando prejuízos a empresas e clientes. Entre os problemas decorrentes de uma invasão podemos citar: vazamento de informações, espionagem comercial, utilização do servidor para ataque a terceiros, entre diversos outros. Realizar uma varredura no ambiente web, à procura de vulnerabilidades, é importante em qualquer instituição. Em muitos casos as empresas não possuem verba para adquirir ferramentas comerciais de *Penetration Test*, então os responsáveis pela Segurança muitas vezes utilizam-se de soluções gratuitas. Desta forma, esse trabalho analisa a eficácia das ferramentas *opensource* de análise de vulnerabilidades, baseando-se nas dez principais falhas de ambientes web de acordo com o relatório *Top Ten* da organização Open Web Application Security Project (OWASP).

Palavras-chave: OWASP (Open Web Application Security Project); Ambiente Web; Aplicações Web; *PenetrationTest* ; Segurança da Informação

SUMÁRIO

1. INTRODUÇÃO

Devido à expansão da Internet, há cada vez mais a tendência de migração de aplicações de ambientes de *desktop* locais para versões online. Muitas empresas hoje já oferecem um site da *web*, seja para apresentar algum produto ou serviço, ou para oferecer uma solução de comércio eletrônico que atenda às demandas do usuário final. Além das empresas, vários órgãos governamentais também migraram seus serviços para o ambiente virtual, permitindo a prestação de diversos serviços por meio da Internet, como por exemplo, emissão de segunda via de documentos, inscrição em concursos, entre outras atividades.

Estas soluções são algumas das facilidades que foram possibilitadas pela popularização da *world wide web* nos últimos anos. Segundo Kurose (2005), a *web* é o conjunto das páginas de hipertexto, ou sites, que podem ser acessados por um usuário através de um navegador de Internet. Entretanto, o grande crescimento desse ambiente, nem sempre foi acompanhado por uma preocupação adequada com a segurança das soluções desenvolvidas.

Ceron et al (2008, p. 2) expõem que:

> O avanço das tecnologias voltadas para a web e a falta da devida preocupação com requisitos de segurança tornam a Internet um ambiente repleto de vulnerabilidades e alvo de frequentes ataques. O problema torna-se ainda mais grave com a utilização dos mecanismos de busca como uma ferramenta para localizar sites vulneráveis. (CERON et al. 2008)

Tanto Ceron et al (2008) quanto Holz (2006) concordam que uma solução web é normalmente mais vulnerável a falhas do que um software tradicional de desktop, já que há diversas camadas de soluções atuando em conjunto criando pontos de vulnerabilidade na integração de todo o ambiente. A atuação dos mecanismos de busca, permitem inclusive, a localização de determinadas falhas em web sites, tornando assim um facilitador para o invasor.

Essa visão também é corroborada por Macedo, Queiroz e Damasceno (2010) quando expõem que devido ao crescimento exponencial do mercado de web, muitas empresas têm recorrido a soluções de frameworks terceirizados, que podem conter diversas vulnerabilidades.

Garantir a proteção da informação como um todo é importante para "garantir a continuidade do negócio, minimizar o risco ao negócio, maximizar o retorno sobre os investimentos e as oportunidades de negócio." (ISO 1 7799, 2005). Os impactos causados por sistemas web inseguros podem gerar enormes prejuízos para os envolvidos tanto do âmbito financeiro, quanto em âmbito intelectual.

2. PROBLEMA DE PESQUISA

Os sistemas baseados em web, cada vez mais complexos, trazem facilidade e usabilidade para a vida do usuário. Porém, por serem mais complexos, tornam-se mais vulneráveis a invasões. Para combater esses riscos, surgem as ferramentas para identificação de vulnerabilidades, que examinam toda a estrutura de um site para identificar potenciais brechas de segurança. Entretanto, a maioria das ferramentas de análise de vulnerabilidades são proprietárias e possuem um custo muito alto para o uso de uma pequena ou média empresa.

Com o advento do software livre, muitas empresas agora podem ter seus próprios serviços de Internet de forma acessível e com custos decrescentes e utilizar ferramentas com código-fonte aberto para identificar vulnerabilidades em seus sistemas. A partir desses pressupostos, surgem as perguntas que norteiam a presente pesquisa: quais seriam os softwares de código-aberto mais eficientes para serem utilizados na identificação de vulnerabilidades e qual a eficiência das ferramentas de análise de vulnerabilidade em ambientes web?

3. JUSTIFICATIVA E RELEVÂNCIA DO TEMA

Ao contrário de softwares, que normalmente são projetados para ser executado em apenas uma plataforma computacional, uma aplicação web é criada de modo que possa ser executado em qualquer dispositivo – *tablets*, computadores pessoais ou *smartphones*. Esse fator de complexidade aumenta o risco de exploração de falhas, pois nem sempre o software é desenvolvido tendo a segurança como um dos focos principais.

Uma pesquisa realizada pela *Web Application Security Consortium* (WASC)[1] em 2008, sugere que 49% das aplicações web testadas, possuem vulnerabilidades de alto risco (urgentes e criticas), que podem ser detectadas por ferramentas de varredura. Rocha, Kreutz e Turchetti (2012, p.1) complementam dizendo que “o número de vulnerabilidades e ataques é cada vez maior e mais frequente em serviços e sistemas Web”.

Torna-se cada vez mais necessário o aprimoramento de novas metodologias de detecção e mitigação de riscos, vulnerabilidades e falhas decorrentes destas, tanto em âmbito privado quanto público, uma vez que atualmente empresas e governos de todo mundo tendem a levar seus sistemas para a Internet. Como consequência, caso estes sistemas sejam explorados por um invasor, o banco de dados pode ser comprometido e as informações armazenadas podem ser acessadas por pessoas não autorizadas.

Portanto, esse trabalho justifica-se pela importância de se discutir e identificar as principais vulnerabilidades de aplicações web através do uso de softwares livres. O estudo pretende levantar as principais ferramentas utilizadas para a varredura de brechas e analisar a sua capacidade de detectar falhas em um ambiente web.

A pesquisa realizada segue a linha de pesquisa de Sistemas de Informação, baseando seu conteúdo também na área de Ciências da Computação. São discutidas temáticas pertencentes ao campo das redes de computadores, segurança da informação e gestão da informação.

[1]Disponível em http://projects.webappsec.org/w/page/13246989/Web%20Application%20Security%20Statistics.

4. OBJETIVOS

4.1 Objetivo Geral

O objetivo geral desse trabalho é identificar as principais vulnerabilidades em aplicações web e medir o desempenho de ferramentas livres de análise de falhas.

4.2 Objetivos Específicos

Os objetivos específicos estabelecidos para a pesquisa são:

- Identificar as principais vulnerabilidades que podem impactar no funcionamento de um ambiente web;
- Realizar um levantamento das principais ferramentas com código-fonte aberto utilizadas para varredura de vulnerabilidades em ambientes web.
- Testar as ferramentas selecionadas em um ambiente virtual com falhas simuladas, comparando-as em relação à eficiência na detecção de vulnerabilidades catalogadas.

5. FUNDAMENTAÇÃO TEÓRICA

O presente trabalho apresenta sua fundamentação teórica, dividindo-a em quatro seções, que estão correlacionadas pela sequência do conteúdo apresentado: primeiramente, será apresentado o ambiente web e, especialmente, as tecnologias que o compõem, como os servidores web e seus frameworks. A seguir, será abordada a segurança da informação, com destaque para alguns conceitos utilizados na pesquisa, como intrusão e vulnerabilidade. Na terceira seção, voltada para discussão do *penetration test*, serão expostas as etapas e processos existentes em um procedimento de análise de vulnerabilidades. Por último, as vulnerabilidades da web serão abordadas por meio da discussão das principais falhas de um ambiente web, do projeto OWASP e das principais ferramentas empregadas para verificação das vulnerabilidades.

A divisão em quatro seções visa tornar mais clara a exposição da fundamentação teórica que sustenta o trabalho de pesquisa, buscando discutir o que é a web, suas vulnerabilidades e as ferramentas que podem ser potencialmente utilizadas para detectá-las.

5.1 Ambientes Web

Com o uso cada vez mais frequente de recursos voltados para a *web*, as empresas têm cada vez mais levado seus produtos e serviços para a esfera da Internet. Por essa razão, soluções do tipo *Business to Business* (B2B) e *Business to Client* (B2C) utilizam sistemas de gerenciamento de bancos de dados que são parcialmente ou totalmente integrados à *World Wide Web*.

De acordo com Lima (2003,p.290), “a *World Wide Web* nasceu em 1989 no CERN, advinda da necessidade de fazer com que grupos de pesquisadores de diferentes nacionalidades pudessem colaborar uns com os outros”. Lima (2003) expõe, portanto, que a *web* nasceu como uma ferramenta de colaboração científica, e desde então trouxe a possibilidade de obtenção de vantagens competitivas para as empresas que a utilizam para divulgar, vender ou prestar serviços e produtos.

Existem atualmente dezenas de diferentes padrões e tecnologias que podem ser utilizados para a construção de um website, ou um portal – uma versão mais complexa de um site web que integra normalmente um grande número de recursos para acesso do cliente.

A maioria dos websites integra a criação de recursos visuais (*webdesign*), o uso de scripts de programação e a integração destes com um banco de dados. Entretanto, segundo Stallings (2005) independente dos padrões utilizados para essa integração, os protocolos de acesso utilizados pelo cliente são o *HTTP* e o *HTTPS*.

5.1.1 – Protocolo HTTP (*Hyper Text Transfer Protocol*)

O HTTP, abreviação de *Hyper Text Transfer Protocol*, é um dos principais protocolos de aplicação utilizado na web. Os protocolos de aplicação são usados em uma rede de computadores para permitir a comunicação entre o software usado no sistema do usuário final e o servidor que hospeda o recurso que será acessado.

Kurose (2005, p.7) expõe que “um protocolo define o formato e a ordem das mensagens trocadas entre duas ou mais entidades comunicantes, bem como as ações realizadas na transmissão e/ou no recebimento da uma mensagem ou outro evento”.

Desde a criação da Web, o HTTP é utilizado pelos navegadores de Internet para permitir aos clientes realizarem o acesso à sites da Internet, baixando e processando os recursos gráficos das páginas (como imagens e vídeos) para a visualização dos usuários finais. De acordo com Assunção (2014, p.38) “o HTTP é quem define como as páginas são formatadas e transmitidas e que ações os servidores web e browsers devem tomar ao responder a certos métodos”.

No entanto, com a rápida expansão da web como meio para troca de informações, tornou-se necessário que em determinados acessos, houvesse um meio a garantir o sigilo e a identidade do remetente e do receptor.

5.1.2 – Protocolo HTTPS (*Hyper Text Transfer Protocol Secure*)

O protocolo HTTPS é resultado da utilização do HTTP em conjunto com protocolos que oferecem recursos de criptografia, provendo uma camada de novas funções. Atualmente, os dois protocolos de criptografia mais usados para esse fim são o SSL e o TLS.

Os protocolos SSL (*Secure Socket Layer*), e seu equivalente, TLS (*Transport Layer Security*) utilizam o padrão de certificados X.509 (ITU-T, 2001) e criptografia assimétrica para proteger os dados que trafegam na Internet. Eles foram desenvolvidos com o objetivo de tornar a comunicação na Internet mais segura, já que muitos protocolos da camada de

aplicação não possuem nenhum tipo de criptografia nativa, o que poderia levar a diversos tipos de ataques.

De acordo com Stallings (2008):

> Três protocolos de camada superior são definidos como parte do SSL: o Protocolo de Estabelecimento de Sessão (*HandshakeProtocol*), o Protocolo de Mudança de Especificação de Cifra (*ChangeCipherSpecProtocol*) e o Protocolo de Alerta (*AlertProtocol*). Esses protocolos são usados no gerenciamento de trocas SSL.(STALLINGS, 2008, p. 185)

O Secure Socket Layer se baseia em dois conceitos complementares, a conexão SSL e a sessão SSL. As seções SSL são criadas pelo Protocolo de Estabelecimento de Sessão e funcionam como uma associação entre um servidor e seus clientes. As sessões são usadas para evitar a renegociação de novos parâmetros seguros a cada nova conexão, definindo assim um conjunto de parâmetros criptográficos que podem ser compartilhados entre múltiplas conexões.

Assunção (2014) expõe que mesmo o processo o protocolo HTTPS possuir um processo de criptografia para proteção dos dados, ele ainda pode ser passível a ataques de interceptação de tráfego.

5.1.3– Certificados digitais

Como o SSL e o TLS trabalham com criptografia assimétrica, que exige uma chave pública e uma privada para completar uma transação, é necessária uma infraestrutura para distribuição e uso de certificados digitais. O certificado é documento digital utilizado como uma maneira de distribuir a chave pública através de uma estrutura PKI (*Public Key Infrastructure*). Muitas instituições hoje possuem sua própria infraestrutura de chaves públicas e podem gerar certificados digitais para terceiros, seja de forma gratuita ou cobrando por esse serviço. Essas empresas são chamadas de autoridades de certificação.

5.1.4 – Servidores Web

Servidores Web são servidores responsáveis por aceitar requisições HTTP de clientes web e entregar respostas deste protocolo, geralmente em forma de páginas web contendo conteúdos estáticos (como textos e imagens) e dinâmicos (scripts) (LIMA, 2003). O conteúdo

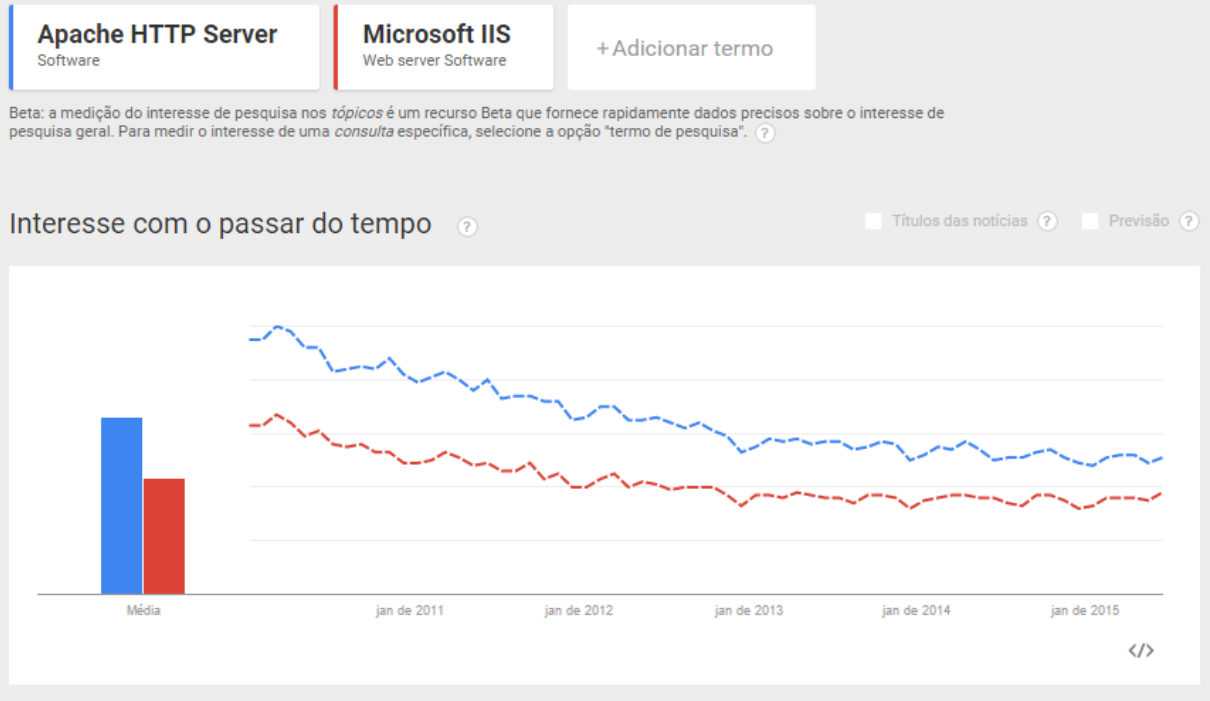
disponibilizado pelo servidor é visto pelo cliente através de navegadores tais como: Internet Explorer, Netscape, Google Chrome.

O servidor que provê o serviço *web*, ou seja, hospeda o conteúdo para consulta, executa um software específico para realizar essa função tais como o IIS (*Internet Information Services*), que é proprietário da Microsoft, e o Apache, que é uma das soluções de código-fonte livre, que é executado em diversas plataformas.

5.1.4.1 – Apache

O Apache é um servidor web de alta-performance, leve e robusto, usado tanto por pequenas empresas quanto largas corporações. Possui código-fonte aberto e oferece uma gama de recursos como: autenticação, controle de acesso, logs de acesso e extensões de scripts. Por ser um software livre, novas funcionalidades podem ser adicionadas por através de sua API (*Application Programming Interface*).

Figura 1. Comparativo de popularidade entre os servidores



Disponível em: http://goo.gl/mKfuQk. Acesso em 09 de Junho de 2015.

A figura acima demostra que durante cinco anos, o Apache é o servidor mais buscado pelo mecanismo de busca Google e consequentemente, bastante utilizado. Assunção (2009)

corrobora esta afirmação alegando que o Apache em conjunto com a linguagem PHP *(Hypertext Preprocessor*) é a combinação de soluções mais comum em servidores da Internet.

5.1.5 – Linguagens de desenvolvimento web

Para produzir conteúdo dinâmico, é preciso utilizar linguagens de programação que permitam maior flexibilidade que o HTTP consegue executar. Nesse campo, surge linguagens como o Java, PHP, Javascript, Python, Ruby, entre várias outras.

O termo “java” geralmente é utilizado para designar uma plataforma, composta pela linguagem de programação, a máquina virtual (Java Virtual Machine) e muitas APIs de controle e desenvolvimento.

Schildt (2013) classifica a linguagem de programação como o responsável por definir os elementos específicos do código-fonte do programa, e dentro desse contexto o Java é considerado uma linguagem de programação de alto nível, orientada a objeto, tornando o processo de criação do software mais rápida, fácil e confiável.

A linguagem Java, que é descendente direta de linguagens como C e C++, ganhou forte terreno com a expansão da Internet, pois permite a criação de programas portáveis, os applets, aplicações baixadas sob demanda, que são executados em navegadores compatíveis com Java. Os applets são possíveis devido à JVM (Java Virtual Machine) que atua como uma camada de hardware simulado que interpreta bytecodes, que pode ser definido como um conjunto de instruções altamente otimizado.

Utilizar a JVM garante a portabilidade, que precisa existir dentro do ambiente Web, devido à grande variedade de arquitetura computacional, e garante um controle de segurança, já que a própria JVM pode inclusive reter o programa, impedindo efeitos colaterais no sistema. Mas muitas vezes, a linguagem orientada a objeto pode ser complementada por scripts, para automatizar tarefas como consultas simples a banco de dados.

5.1.5.1 – Scripts

Scripts podem definidos como um conjunto de instruções que realizam tarefas geralmente repetitivas. Costa (2010), afirma que não há um consenso quanto a essa classificação. O uso da linguagem de scripts evoluiu ao ponto desse conceito aplicar-se também a programas de computador interpretados.

As linguagens baseadas em scripts apresentam características que as diferem das outras linguagens de programação convencionais, como a não geração de um programa executável, à partir de um código fonte.

Em ambientes Web, os scripts podem ser utilizados como uma forma de acrescentar movimento e comportamento às páginas. Uma delas é o PHP é uma linguagem de programação de software-livre, baseada em scripts com múltiplas funções, que foi concebida originalmente por Rasmus Lerdorf no ano de 1995.

Ao contrário do HTML, que é executado pelo browser quando uma página é aberta, PHP é pré-processado pelo servidor. Assim, todo o código PHP incluso num arquivo é processado pelo servidor antes de enviar algo para o cliente, através do browser. Por esse motivo, muitas vezes é referido como uma linguagem *server-side*.

De acordo com Sica e Real (2007, p. 6), “ao acessar alguma página desenvolvida em PHP, o servidor HTTP direciona a requisição para o interpretador PHP, que por sua vez, executa o script contido na página solicitada e devolve ao cliente.”

Um outro exemplo de linguagem script é o Javascript. Costa (2010) afirma que essa linguagem é atualmente a principal linguagem para programação *client-side* em navegadores web. Ela não possui classes, mas possui construtores que são capazes de fazer o que as classes fazem, incluindo agir como contêineres para classes de variáveis e métodos.

5.1.6 – Banco de dados

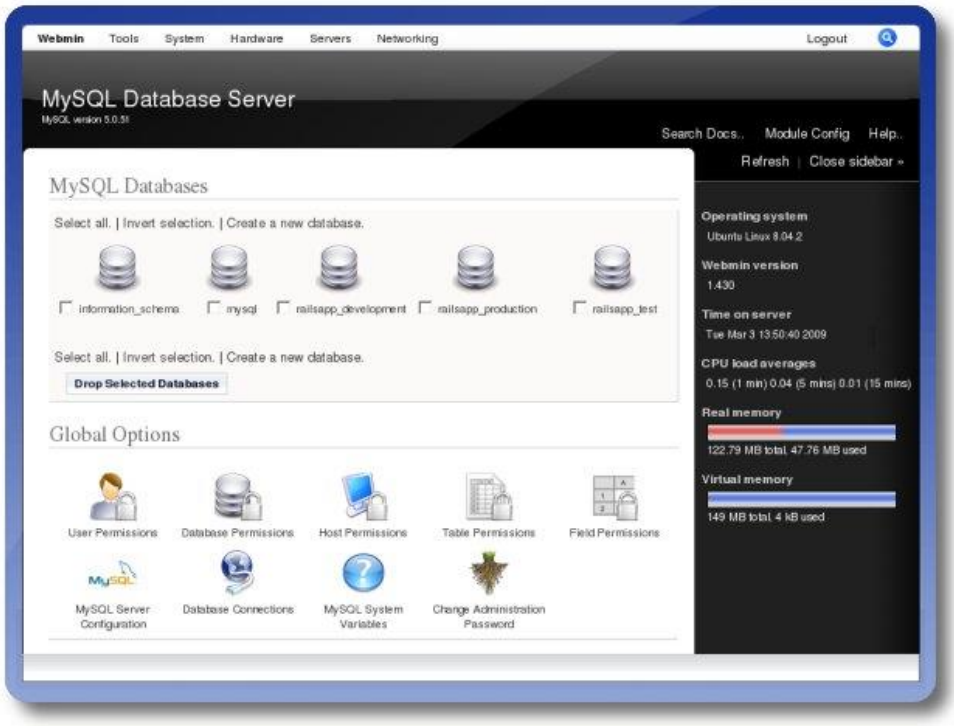
Um banco de dados (sua abreviatura é BD, em inglês DB, *database*) é uma entidade na qual é possível armazenar dados de maneira estruturada e com a menor redundância possível. Estes dados poderão ser então acessados e utilizados por usuários distintos, através de softwares específicos. A linguagem mais comumente utilizada para a comunicação entre os usuários e o banco é a SQL (*Structured Query Language*). (LIMA, 2003).

Para poder controlar os dados e os usuários, utiliza-se um sistema chamado de SGBD (sistema de gestão de bancos de dados) ou em inglês DBMS (*Database Management System*). O SGBD é um conjunto de serviços (aplicações software) que permitem gerenciar os bancos de dados, isto é:

- Permitir o acesso aos dados de maneira simples;
- Autorizar um acesso às informações a múltiplos usuários;
- Manipular os dados presentes no banco de dados (inserção, supressão, modificação);

Existem atualmente diversas soluções para gerenciamento e acesso a banco de dados, como: Microsoft SQL Server, MySQL, PostgreSQL e diversas outras. De acordo com Welling e Thompson (2004, p.25), “o MySQL é um sistema de gerenciamento de dados relacional, isto é, suporta bancos de dados que consistem em um conjunto de relacionamentos”.

Figura 2: Gerenciador de banco de dados MySQL



Disponível em: http://commons.wikimedia.org/wiki/File:Webmin-mysql-fw.png. Acesso em março de 2015.

Welling e Thompson (2004) concordam com Pessoa (2007) ao afirmarem que um banco bem planejado e estruturado pode evitar problemas de redundância de dados, que podem ocasionar perda ou corrompimento dos mesmos. Muitos dos ataques a ambientes web visam como objetivo alcançar os dados no banco, portanto a proteção desse sistema é vital para uma solução segura.

5.2 Segurança da Informação

A informação é o principal bem de qualquer empresa ou instituição, seja pública ou privada. A Segurança da Informação é a área do conhecimento responsável por proteger os dados e os ativos de uma instituição contra vários tipos de ameaças, como o vazamento de informações internas.

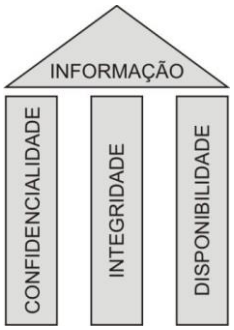
Um dos principais papéis da área de Segurança da Informação é proteger todo o conhecimento institucional gerado a partir do uso das informações. Campos (2014) afirma que "a informação é o elemento essencial para todos os processos de negócio da organização, sendo, portanto, um bem ou ativo de grande valor".

Os ativos de uma empresa são os elementos físicos, humanos ou tecnológicos que fazem parte do fluxo de informações dentro da companhia. A Segurança da Informação utiliza-se de padrões, processos, e regras para controlar e monitorar o uso destes ativos, a fim de proteger o conhecimento organizacional.

5.2.1 – Pilares da segurança da informação

Um sistema de segurança da informação baseia-se em três princípios, ou pilares, básicos: Confidencialidade, Integridade e Disponibilidade. Somente alcançamos um nível de segurança satisfatório se todos os pilares foram corretamente implementados.

Figura 3: Pilares da Segurança da Informação



Disponível em: http://www.helviojunior.com.br/it/security/clareza-e-produtividade-na-gestao-do-firewall-aker-6-1/. Acesso em março de 2015.

Stallings (2010, p.10) deixa explícita a ideia de que “a confidencialidade é a proteção dos dados transmitidos contra ataques passivos”. Esse pilar da segurança é responsável pela codificação dos dados com o uso de técnicas criptográficas, de modo que somente o receptor possa decodificar corretamente a mensagem enviada.

O segundo pilar é o que se refere à integridade. Campos (2014) explica que o pilar da integridade refere-se à proteção dos dados em trânsito em uma rede de computadores, ou outro sistema de transmissão, contra alterações impróprias. Outra função da integridade é prevenir que os dados sejam corrompidos durante o processo de envio da mensagem ao receptor final.

O pilar da disponibilidade é responsável por manter os dados sempre à disposição quando forem requisitados. “A RFC 2828 define a disponibilidade como sendo a propriedade de um sistema ou de um recurso do sistema ser acessível e utilizável sobre demanda por uma entidade autorizada do sistema.” (STALLINGS,2010, p.10)

5.2.2 - Vulnerabilidade

A Vulnerabilidade é o fator que, quando não devidamente detectado e corrigido, pode levar à intrusão e ao comprometimento do sistema. De acordo com Campos (2014, p.23), “os ativos de informação, que suportam os processos de negócio, possuem vulnerabilidades. É importante destacar que essas vulnerabilidades estão presentes nos próprios ativos, ou seja, que são inerentes a eles, e não de origem externa”.

5.2.3 – Intrusão

O processo de intrusão é o comprometimento do sistema pelo invasor, quando este tem acesso a dados privilegiados que não estão disponíveis ao público. Campos (2014) e Stallings (2010) alegam que a intrusão é uma consequência da descoberta e exploração de uma vulnerabilidade, seja ela proveniente de um problema técnico ou falha humana. SOUZA (2002) também aborda o comprometimento de um sistema ao expor que a intrusão ou ataque podem ser definidos como um conjunto de ações que visam comprometer a disponibilidade, integridade ou confidencialidade de um recurso computacional.

Muitos ataques envolvem várias etapas ou fases. Seguem abaixo as principais:

1. Coleta de informações: levantamento não intrusivo de informações do sistema alvo, baseando-se normalmente em fontes públicas de informação.

2. Varredura: pesquisa acerca do sistema alvo para descoberta de características que possam auxiliar na intrusão.

3. Penetração: obtenção de acesso privilegiado não autorizado, através de uma brecha ou vulnerabilidade do sistema.

4. Negação de serviço: comprometimento ou interrupções no funcionamento do sistema ou de partes dele.

5. Remoção de rastros: remoção dos registros que possam identificar o ataque.

6. Criação de entradas: instalação de programas ocultos (*backdoors*) que permitam uma nova invasão ao sistema.

As etapas descritas acima não são regras para criação de um plano de ataque. Podem existir todas ou apenas algumas delas. Cabe destacar que as fases de varredura, penetração e negação de serviço ocorrem na maioria dos ataques.

5.3 Penetration Test

Penetration Test é uma técnica em que são utilizadas ferramentas de hackers para realizar a análise de um sistema ou rede em busca de falhas de segurança. Também chamado de Teste de Invasão ou análise de vulnerabilidades, o Penetration Test Penetration Test é realizado por indivíduos conhecidos como Hackers Éticos. Assunção (2002) refere-se ao Hacker Ético como um profissional treinado para descobrir, identificar e explorar novas falhas em sistemas.

5.3.1 Planejamento e cronograma do Penetration Test

Um *Penetration Test* exige um planejamento cuidadoso do escopo do que será testado, assim como a duração do processo e os itens contidos no relatório final. No início do planejamento, o hacker ético irá colher as informações necessárias para a execução do teste. Dentre estas informações, podemos citar: o escopo do projeto, seu objetivo, duração, tarefas a serem realizadas, custo do serviço, forma de pagamento e as metas que devem ser alcançadas.

As fases de realização de um teste de invasão são: planejamento, execução e pós-teste.

5.3.1.1 Cronograma de realização

O cronograma da realização de um *Penetration Test varia* de acordo com três fatores: a quantidade de tarefas a serem realizadas, a quantidade de equipamentos que serão testados e o número de horas diárias demandadas para realização do teste.O tempo médio é de duas a quatro semanas para cada uma das três fases, o que resulta em uma duração média total entre seis a doze semanas para a realização de um teste completo do tipo caixa preta (*black-box*).

Assunção (2014, p. 36) discorre sobre as vantagens de um teste caixa preta:

> [...] um teste black-box compreende uma análise completa do sistema alvo partindo do pressuposto que o invasor não conheça nenhuma informação preliminar para ajudar na verificação de vulnerabilidades. É o tipo de teste mais completo, pois exige do Hacker Ético uma extensa pesquisa sobre o objeto de análise. Assunção (2014, p. 36)

5.3.2 – Metodologias de Penetration Test

Assim como ocorre com outras áreas de TI, além da Segurança da Informação, diversas entidades desenvolveram normas para uma maior padronização das tarefas a serem realizadas em um *Penetration Test.* Estes padrões são importantes, pois facilitam a organização de um processo de análise de vulnerabilidades em larga escala.

Os dois padrões mais conhecidos são o OSSTMM (*Open Source Security Testing Methodology Manual)*, desenvolvido pela ISECOM (*Institute for Security and Open Methodologies*)*,* e o padrão ISSAF (*Information Systems Security Assessment Framework*), criado pelo OISSG (*Open Information Systems Security Group*).

5.3.2.1 – OSSTMM (*Open Source Security Testing Methodology Manual*)

O Open Source Security Testing Methodology Manual (ISECOM, 2015), ou simplesmente OSSTMM, é a metodologia mais popular para padronização de um Penetration Test . Os testes padronizados são explicados em um alto nível de detalhes, e incluem também a análise de diversos tipos de tecnologias de redes sem fio como bluetooth, infravermelho e wi-fi.

A documentação do OSSTMM não é considerada um ferramental, portanto, as técnicas sugeridas podem ser utilizadas através de qualquer ferramenta de análise de vulnerabilidades compatível com o teste proposto. Por este motivo, apesar de oferecer uma maior flexibilidade na realização dos processos, o OSSTMM possui uma maior curva de aprendizado.

5.3.2.2 – ISSAF(*Information Systems Security Assessment Framework*)

O Information Systems Security Assessment Framework, ou ISSAF, é um manual de padronização desenvolvido para a realização de checklists de auditoria em sistemas, abordando também técnicas para a análise de vulnerabilidades. Sua documentação é divida em duas partes: Gerenciamento geral do processo (ISSAF0.2.1A) e PenetrationTesting (ISSAF0.2.1B).

Uma característica do ISSAF é o fato de não exigir conhecimento prévio em ferramentas e sistemas operacionais. A segunda parte de seu manual (ISSAF0.2.1B) aborda de forma detalhada, com imagens e exemplos, o uso do ferramental necessário para a realização dos testes.A documentação do ISSAF propõe a divisão do Penetration Test em cinco fases distintas: planejamento, análise, tratamento, resultados e manutenção.

5.3.3 – Tipos de testes

Stuart, Scambray e Kurtz (2014) concordam com Assunção (2009) e Engebretson (2013) que a definição do tipo de teste a ser realizado afeta diretamente os resultados do processo, sendo, portanto, muito importante compreender e escolher o teste correto para cada situação.

Tanto o OSTMM quanto o ISSAF definem três tipos de *Penetration Test* que podem ser realizados. São os testes de caixa preta (Black-box), caixa branca (White-box) e caixa cinza (Greybox). Todos os tipos possuem muitas similaridades, sendo diferenciados por seus objetivos finais e o escopo.

5.3.3.1 – *Black-box*

Utilizado para simular ataques de invasores externos que não têm nenhum conhecimento sobre a rede ou sistema que estão varrendo. O invasor ou aquele que simula a invasão deve obter todo o tipo de informação para conseguir descobrir os pontos fortes e fracos de um sistema. A maior vantagem desse teste é a simulação em modo real da visão de um atacante externo sobre o sistema, e a identificação de possíveis vazamentos de informação. As desvantagens incluem um maior tempo de realização e o fato de mostrar apenas a visão de um invasor externo quando, na realidade, a maioria dos ataques acontece por parte de pessoas que estão dentro do ambiente.

5.3.3.2 – *White-box*

Testes de caixa branca são feitos com total conhecimento preliminar da rede. Seus objetivos são mais específicos, geralmente são utilizados para descobrir vulnerabilidades no sistema e não estão focados na visão de um invasor externo e nem na descoberta de vazamentos de informações.

5.3.3.3 – *Gray-box*

O teste *gray-box* é utilizado para identificar possíveis métodos de ataque que poderiam ser realizados por alguém que está dentro da corporação. O exemplo comum é o de um funcionário que tem o conhecimento parcial da rede na qual ele se encontra e tenta um ataque contra outro departamento. Atualmente, muitos ataques iniciam-se dentro das empresas, o que torna o *Gray-box* uma opção de teste importante para consideração.

5.3.4 – Pós-teste

Após a conclusão da análise das vulnerabilidades, é necessário reunir e organizar todas as informações obtidas durante o processo, como, por exemplo, os tipos de problemas encontrados e o nível de gravidade de cada um. Em geral, utiliza-se uma classificação de acordo com a gravidade da falha encontrada: se é de baixo risco, médio risco ou alto risco. Essas informações devem estar contidas no relatório final.

5.3.4.1 – Relatório final

O relatório final é o documento que será composto das informações e falhas encontradas durante o *Penetration Test.* Deve seguir um modelo claro, de preferência baseado em tópicos, sem conter informações confusas. Assunção (2014) recomenda quatro itens essenciais que devem estar contidos no relatório: introdução, detalhes do trabalho, resultados obtidos e recomendações.

Além do relatório final, Mcclure, Scambray e Kurtz (2014) recomendam a criação de um Termo de responsabilidade que deverá ser assinado pelo prestador do serviço de *Penetration Test,* assim como pela empresa responsável pela contratação do teste. Engebretson (2013) e Assunção (2014) também concordam com a importância deste documento, de forma a proteger legalmente o hacker ético já que o penetration test pode levar ao comprometimento do Sistema e acesso a informações confidenciais.

5.4 Vulnerabilidades no ambiente da web

O ambiente web é composto pelo servidor que hospeda as páginas, a linguagem de programação utilizada, além de gerenciadores de conteúdo como o Wordpress e outras soluções. É um ambiente heterogêneo que precisa trabalhar como uma única entidade.

Doupé (2012) explica que a ocorrência das vulnerabilidades de uma aplicação web pode ser diminuída através de uma melhor educação do desenvolvedor ou pelo uso de ferramentas que permitem a realização de testes de segurança. A OWASP (*Open Web Application Security Project*) possui projetos que visam tanto educar os programadores, quanto ajuda-los no processo de identificação das falhas web.

5.4.1 Projeto OWASP (*Open Web Application Security Project)*

O projeto OWASP é resultado dos esforços da entidade sem fins lucrativos de mesmo nome, e tem como objetivo encontrar e combater a causa da insegurança em aplicativos web. Meucci (2008) expõe que a OWASP não é afiliada com nenhuma empresa de tecnologia, portanto as informações de segurança geradas pela comunidade do projeto são imparciais e práticas.

De acordo com Oliveira (2012, p. 49), “os projetos OWASP são divididos em duas categorias: desenvolvimento e documentação”. Curphey et al (2005) e Meucci (2008) citam alguns dos principais projetos desenvolvidos:

- *Development Guide* é o projeto de documentação que foca na segurança durante o desenvolvimento de aplicações web. Ele demonstra vários aspectos pertinentes à segurança das soluções

- *Testing Guide* é um guia prático norteando as questões: o que testar, porque, quando, onde e como testar as aplicações Web em busca de vulnerabilidades.

- *Code Review Guide* é um guia que mostra boas práticas para a revisão segura de códigos com o objetivo de encontrar vulnerabilidades. É um pouco mais técnico e extenso do que o testing guide.

- *Top Ten* é o guia mais popular da OWASP. O *Top Ten* é basicamente um documento que referência e explica as dez principais vulnerabilidades das aplicações web. Mostra as consequências da exploração de cada uma dessas falhas, assim como algumas técnicas para se proteger contra elas.

Carvalho (2014) expõe que o documento do Top Ten não lista necessariamente as falhas mais críticas de um ambiente web, e sim as mais frequentes. Por esta razão ao longo do tempo a lista vai sofrendo alterações: algumas vulnerabilidades são removidas e outras colocadas em seu lugar.

Apresenta-se a seguir algumas das vulnerabilidades já contempladas pelo documento *Top Ten* e a explicação do significado de cada uma delas.

Quadro 1: Vulnerabilidades Web reportadas pela OWASP. Adaptado do *Top Ten*.

Cross Site Scripting (XSS)	Os furos XSS ocorrem sempre que uma aplicação obtém as informações fornecidas pelo usuário e as envia de volta ao navegador sem realizar validação ou codificação daquele conteúdo. O XSS permite aos atacantes executarem scripts no navegador da vítima, o qual pode roubar sessões de usuário, pichar sites Web, introduzir código malicioso, etc.
Falhas de Injeção	As falhas de injeção, em especial *SQL Injection*, são comuns em aplicações Web. A injeção ocorre quando os dados fornecidos pelo usuário são enviados a um interpretador comparte do comando ou consulta. A informação maliciosa fornecida pelo atacante, engana o interpretador que irá executar comandos mal-intencionados ou manipular informações.
Execução maliciosa de arquivos	Os códigos vulneráveis à inclusão remota de arquivos (RFI) permitem ao atacante incluir código e dados maliciosos, resultando em ataques devastadores, como o comprometimento total do servidor. Os ataques de execução de arquivos maliciosos afetam PHP, XML e todos os frameworks que aceitem nomes de arquivo ou arquivos dos usuários.
Referência Insegura Direta à Objetos	Uma referência direta a um objeto ocorre quando um desenvolvedor expõe a referência a um objeto implementado internamente, como é o caso de arquivos, diretórios, registros da base de dados ou chaves, na forma de uma URL ou parâmetro de formulário. Os atacantes podem manipular estas referências para acessar outros objetos sem autorização.

Cross Site Request Forgery (CSRF)	Um ataque CSRF força o navegador da vítima, que esteja autenticado em uma aplicação, a enviar uma requisição pré-autenticada a um servidor Web vulnerável, que por sua vez força o navegador da vítima a executar uma ação maliciosa em prol do atacante. O CSRF pode ser tão poderoso quanto a aplicação Web que ele ataca.
Vazamento de Informações e Tratamento de Erros Inapropriado	As aplicações podem divulgar informações sobre suas configurações, processos internos ou violar a privacidade por meio de uma série de problemas na aplicação, sem haver qualquer intenção. Os atacantes podem usar esta fragilidade para roubar informações consideradas sensíveis ou conduzir ataques mais estruturados.
Autenticação falha e Gerenciamento de Sessão	As credenciais de sessão não são protegidas apropriadamente com bastante frequência. Atacantes comprometem senhas, chaves ou informações de autenticação de forma a assumir a identidade de outros usuários.
Armazenamento Criptográfico Inseguro	As aplicações Web raramente utilizam funções criptográficas de forma adequada para proteção de informações e credenciais. Os atacantes se aproveitam de informações mal protegidas para realizar roubo de identidade e outros crimes, como fraudes de cartões de crédito.
Comunicações inseguras	As aplicações frequentemente falham em criptografar tráfego de rede quando se faz necessário proteger comunicações críticas/confidenciais.
Falha de Restrição de Acesso à URL	Frequentemente, uma aplicação protege suas funcionalidades críticas somente pela supressão de informações como links ou URLs para usuários não autorizados. Os atacantes podem fazer uso desta fragilidade para acessar e realizar operações não autorizadas por meio do acesso direto às URLs.

Disponível em: https://www.owasp.org/images/9/9c/OWASP_Top_10_2013_PT-BR.pdf. Acesso em março de 2015.

Os riscos associados a essas falhas divulgadas anualmente, através de uma coleta de dados, podem ser verificados através do aumento de incidentes envolvendo roubo de invasões documentado pela CERT. Dos mais de 1 bilhão de incidentes ocorridos em 2014, na tentativa de se obter dados em autorização, quase 3% apresentam apenas tentativas de comprometer especificamente o servidor web.

5.4.2 As dez principais vulnerabilidades da web

Segundo a documentação da OWASP TOP TEM, as dez vulnerabilidades mais frequentes em ambientes web são:

- Falhas de injeção de código (*SQL Injection*, *XPATH Injection* e outras);
- Cross Site Scripting (XSS Refletido/Persistido);
- Quebra do processo de autenticação e gerenciamento de sessão;
- Referência insegura a objetos;
- Cross Site Request Forgery (CSRF);
- Má configuração de processos de segurança;
- Insegurança no armazenamento criptográfico;
- Proteção Insuficiente na camada de transporte;
- Redirecionamento e encaminhamentos inválidos.

De acordo com a OWASP, as cinco primeiras vulnerabilidades ocorrem durante a etapa de desenvolvimento da aplicação. Muniz e Lakhani (2013) opinam que a falha de *Cross Site Scripting* é mais impactante do que o *SQL Injection* pois afeta o usuário final, discordando de Pauli (2014), que corrobora a lista da OWASP ao citar o processo de injeção de SQL como a vulnerabilidade web mais grave.

Essa diferença de opiniões entre autores é principalmente devido ao caráter dinâmico das vulnerabilidades, que podem ganhar maior ou menor ênfase ao longo do tempo. De fato, a lista *Top Ten* da OWASP é alterada a cada um ou dois anos, pois novas vulnerabilidades surgem, algumas outras perdem a sua eficácia ou até mesmo deixam de existir. Na imagem há uma comparação da versão 2010 e 2013 da Lista *Top Ten*.

Figura 4: Comparativo da lista *top ten* dos anos de 2010 e 2013.

OWASP Top 10 – 2010 (Anterior)	OWASP Top 10 – 2013 (Novo)
A1 – Injeção de código	A1 – Injeção de código
A3 – Quebra de autenticação e Gerenciamento de Sessão	A2 – Quebra de autenticação e Gerenciamento de Sessão
A2 – Cross-Site Scripting (XSS)	A3 – Cross-Site Scripting (XSS)
A4 – Referência Insegura e Direta a Objetos	A4 – Referência Insegura e Direta a Objetos
A6 – Configuração Incorreta de Segurança	A5 – Configuração Incorreta de Segurança
A7 – Armazenamento Criptográfico Inseguro – Agrupado com A9 →	A6 – Exposição de Dados Sensíveis
A8 – Falha na Restrição de Acesso a URL – Ampliado para →	A7 – Falta de Função para Controle do Nível de Acesso
A5 – Cross-Site Request Forgery (CSRF)	A8 – Cross-Site Request Forgery (CSRF)
<Removido do A6: Configuração Incorreta de Segurança>	A9 – Utilização de Componentes Vulneráveis Conhecidos
A10 – Redirecionamentos e Encaminhamentos Inválidos	A10 – Redirecionamentos e Encaminhamentos Inválidos
A9 – Proteção Insuficiente no Nível de Transporte	Agrupado com 2010-A7 criando o 2013-A6

Disponível em: www.owasp.org/images/9/9c/OWASP_Top_10_2013_PT-BR.pdf. Acesso em Março de 2015.

Percebe-se que algumas das vulnerabilidades mudaram a sua ordem de classificação, enquanto outras deixaram de existir, como é o caso do antigo A6 – “Configuração Incorreta de Segurança.”. Em seu lugar, temos a falha de “Exposição de dados sensíveis”. Este trabalho utilizará a versão de 2013 como base, que atualmente é o padrão mais atual em vigor.

5.4.3 – Ferramentas de análise de vulnerabilidade Web

Hoje existem no mercado dezenas de ferramentas para análise das vulnerabilidades em ambientes Web. Muitas delas são comerciais, com valores que variam de algumas dezenas a milhares de dólares. Entre algumas das ferramentas disponíveis estão o *Acunetix*, o *Nessus* e o *Burp Suite*. Entretanto, alguns fatores devem ser considerados para a escolha da ferramenta que será utilizada para a verificação das falhas.

De acordo com Assunção (2014), um dos fatores relevantes na escolha de uma ferramenta de análise de vulnerabilidades em ambientes web é a quantidade de falsos positivos e falsos negativos gerados após o processo de varredura ser finalizado. Um falso positivo ocorre quando a falha não existe no sistema, mas o software de detecção a identifica erroneamente. Já o falso negativo é quando a vulnerabilidade existe, mas não é detectada pela ferramenta de análise.

Outro fator a ser considerado é a implementação de um perfil para a verificação das falhas citadas na lista *OWASP Top Ten*. Como ao longo do tempo este documento sofre modificações, em virtude dos estudos que identificam a frequência das vulnerabilidades web, a ferramenta deve oferecer atualizações frequentes em sua relação de falhas detectáveis, para se manter relevante ao longo do tempo.

O site *SecTools.org* mantém um ranking das ferramentas de segurança e análise de vulnerabilidades mais populares, segundo especialistas da área (MARTINELO e BELLEZI, 2014).

6. METODOLOGIA DE PESQUISA

Este trabalho desenvolverá uma pesquisa de cunho qualitativo, utilizando os métodos bibliográfico, documental e exploratório. De acordo com Marconi e Lakatos (2003, p.201), “metodologia de pesquisa é aquela que abrange a maior número de itens, pois responde, a um só tempo, às questões: Como? Com quê? Onde? Quanto?”.

Para Gil (2010, p 27) “se o propósito de uma pesquisa for proporcionar uma maior familiaridade com problema conhecido a fim de torná-lo mais explícito ou construir hipóteses, deve-se aplicar a pesquisa exploratória”.

Em relação ao delineamento da pesquisa, ela possui características de caráter experimental onde serão comparadas cinco ferramentas de análises de vulnerabilidades para conhecer a eficiência de cada uma delas na detecção das dez principais falhas da lista Top Ten OWASP, na versão 2013 do documento. O número de ferramentas foi limitado a cinco para permitir um maior aprofundamento na exploração de cada uma delas, respeitando o cronograma de realização deste trabalho.

A OWASP (*Open Web Application Security Project*) é uma organização sem fins lucrativos que visa informar sobre novas vulnerabilidades em ambientes web e ajudar a melhor a segurança das aplicações online. Ela possui uma lista constantemente atualizada das dez falhas mais frequentes em um servidor web. De acordo com Carvalho (2014, p. 5), a lista de vulnerabilidades *OWASP Top Ten* “é desenvolvida e atualizada a partir de consultas feitas a especialistas em Segurança da Informação e organizações de portes e ramos diversos, na esfera governamental ou privada, em vários países). “. Por esta razão este trabalho visa basear o teste das ferramentas escolhidas de análise de falhas na lista de dez vulnerabilidades, pois a mesma é referência na área de Segurança da Informação como problemas que necessitam de rápida correção quando detectados.

As ferramentas serão selecionadas com base no ranking do site “SecTools.Org”, que faz uma listagem das cento e vinte cinco ferramentas de segurança utilizadas e reconhecidas pela comunidade de Segurança da Informação. O ranking se baseia na opinião de mais de dois mil[2] profissionais da área de Segurança da Informação , que podem conceder uma nota às ferramentas de acordo com seus recursos e popularidade.

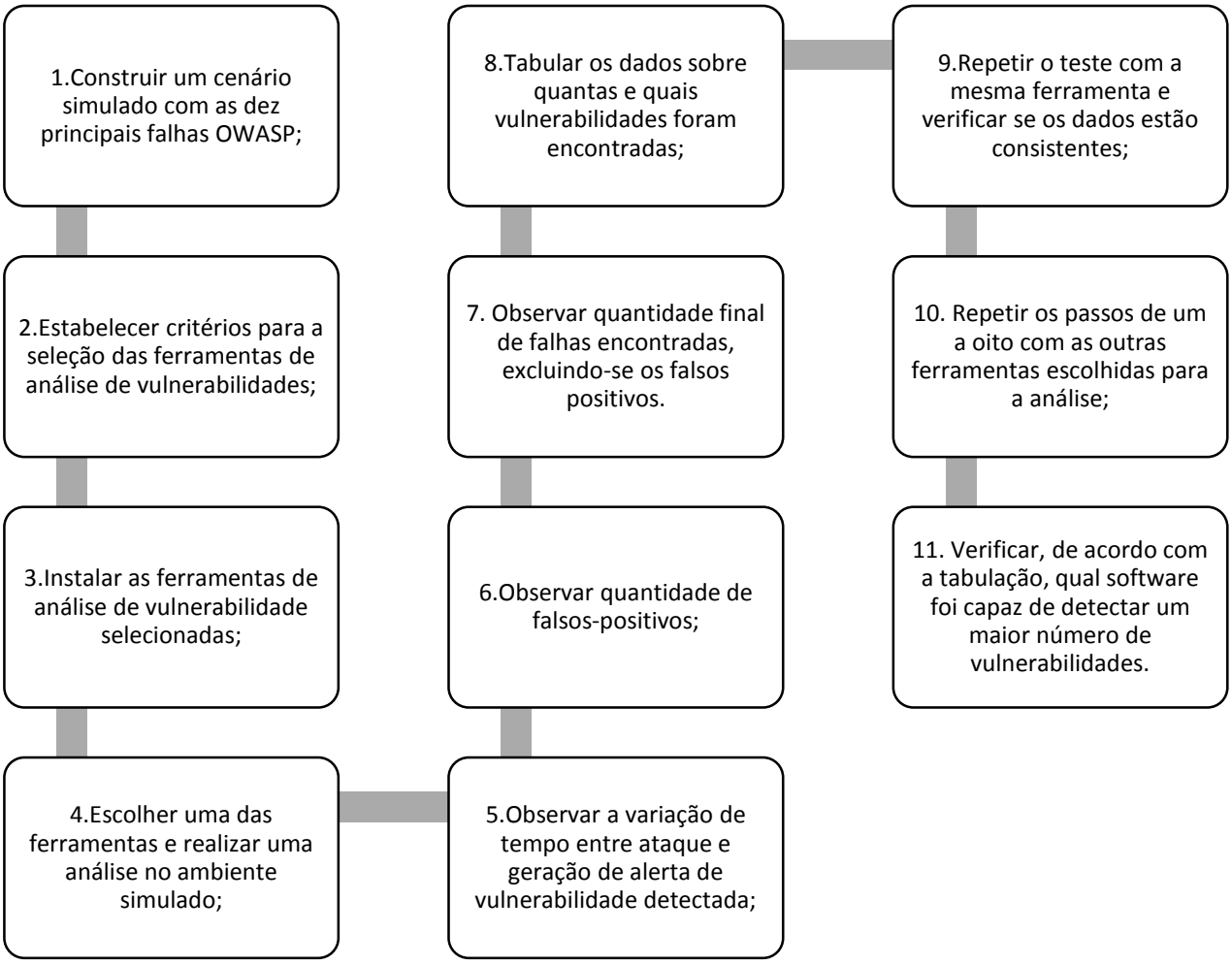
[2] Disponível em http://sectools.org/about

Para a realização da pesquisa, foram escolhidas cinco ferramentas para realizar a análise de vulnerabilidades das falhas do OWASP Top Ten. A seleção levou em consideração apenas as ferramentas *open source* (de código-fonte livre) e o ranking das mesmas no site SecTools. O objetivo da escolha de apenas cinco soluções é permitir uma melhor especificação dos recursos de cada ferramenta, além de uma análise comparativa mais profunda entre a quantidade de falhas detectadas no cenário que será simulado.

Durante o experimento, além dos softwares de teste selecionados, será utilizado o framework Mutillidae[3], que permite simular as vulnerabilidades descritas no documento *OWASP Top Ten* para fins de teste. Também serão coletados os seguintes dados sobre as ferramentas: quantidade de falhas detectadas, número de falsos positivos e velocidade de detecção das falhas. Para fins do trabalho, a ferramenta mais eficiente será considerada aquela que identificar mais vulnerabilidades em um menor tempo. A pesquisa experimental seguirá o seguinte fluxograma:

[3] Disponível em http://sourceforge.net/projects/mutillidae/

Figura 5. Fluxograma de passos para realização da pesquisa



Fonte: Marcos Flávio Araújo Assunção

As conclusões desta pesquisa levarão em consideração tabelas comparativas para levantamento das variáveis analisadas pelo trabalho, como tempo, falhas reais x falsos positivos. Os dados obtidos serão comparados com os trabalhos de outros autores consultados no referencial teórico, e assim permitirão responder à questão inicial e motivadora dessa pesquisa.

7. RESULTADOS

Os resultados ainda serão obtidos e apresentados, de acordo com o cronograma apresentado a seguir.

8. CRONOGRAMA

Figura 6: Cronograma do desenvolvimento da pesquisa para a dissertação

Cronograma do denvolvimento da Dissertação - Marcos Flávio Araújo Assunção									
Atividades	**Fev**	**Mar**	**Abr**	**Mai**	**Jun**	**Jul**	**Ago**	**Set**	**Out**
Desenvolvimento do Pré-Projeto									
Entrega do Pré-Projeto									
Fundamentação e metodologia									
Banca de Qualificação									
Pesquisa para a fundamentação									
Realização de experimentos									
Coleta de resultados									
Defesa da dissertação									

Fonte: Marcos Flávio Araújo Assunção

9. REFERÊNCIAS

ASSUNÇÃO, Marcos F. **Honeypots e Honeynets**. Florianópolis: Visual Books, 2009.

ASSUNÇÃO, Marcos F. **Segredos do Hacker Ético**. 5 ed. Florianópolis: Visual Books, 2014.

CAMPOS, André. **Sistema de Segurança da Informação**. 3 ed. Florianópolis: Visual Books, 2014.

CARVALHO, Alan Henrique Pardo. Segurança de aplicações web e os dez anos do relatório OWASP Top Ten: o que mudou? **Fasci-Tech – Periódico Eletrônico da FATEC-São Caetano do Sul**, São Caetano do Sul, v.1, n. 8, Mar./Set. 2014, p. 6 a 18.

CERON, J; FAGUNDES, Leonardo; LUDWIG, Glauco; TAROUCO. Liane; BERTHOLDO, Leadro. **Vulnerabilidades em Aplicações Web: uma Análise Baseada nos Dados Coletados nos honeypots**. VIII Simposio Brasileiro de Segurança. 2008 Disponível em <http://sbseg2008.inf.ufrgs.br/proceedings/data/pdf/st06_02_resumo.pdf>.

COSTA, D.G. **Administração de Redes com scripts: Bash Script, Python e VBScript**. Rio de Janeiro: Brasport, 2010.

CURPHEY, M.; ENDLER, D.; HAU, W.; TAYLOR, S.; SMITH, T.; RUSSELL,A.; MCKENNA, G.; PARKE, R.; MCLAUGHLIN, K.; TRANTER. A guide to building secure web applications and web services. **The Open Web Application Security Project**, v. 1, 2005.

DOUPÉ, Adam et al. **Enemy of the State: A State-Aware Black-Box Web Vulnerability Scanner**. In: USENIX Security Symposium. 2012. Disponível em < www.usenix.org/system/files/conference/usenixsecurity12/sec12-final225.pdf >. Acesso em 22 de dezembro de 2014.

ENGEBRETSON, Patrick. **Introdução ao Hacking e aos Testes de Invasão**. São Paulo: Novatec, 2013.

GIL, Antonio Carlos. **Como elaborar projetos de pesquisa**. 5. ed. São Paulo: Atlas, 2010.

HOLZ, T.; MARECHAL, S; RAYNAL, F. New threats and attacks on the World Wide Web. **IEEE Security & Privacy Magazine**, v. 4, n. 2, p. 45-50, março. 2006.

ISECOM. **Open Source Security Testing Methodology Manual**. Disponível em < www.isecom.org/osstmm/>. Acesso em 07 de Janeiro de 2015.

ISO 1 7799. **ABNT NBR ISO/IEC 1 7799:2005 – Tecnologia da Informação – Técnicas de segurança – Código de prática para a gestão da segurança da informação**. Associação Brasileira de Normas Técnicas – Rio de Janeiro: ABNT, 2005.

ITU-T. **Recommendation X.509 - Open Systems Interconnection - The directory: authentication framework**. Disponível em <https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.509-199311-S!!PDF-E&type=items>. Acesso em 02 de junho de 2015.

KUROSE, James; ROSS, Keith. **Redes de Computadores e a Internet: uma abordagem top-down**. 3. ed. São Paulo: Pearson, 2005.

LAKATOS, Eva M.; MARCONI, Marina A. **Fundamentos de metodologia científica**. 5. ed. São Paulo: Atlas. 2003.

LIMA, João P. **Administração de Redes Linux**. Goiânia: Terra, 2003.

MACÊDO, Márcio A.; QUEIROZ, Ricardo G.; DAMASCENO, Julio C..jShield: **Uma Solução Open Source para Segurança de Aplicações Web**. Universidade Federal de Pernambuco. 2010.

MARTINELO, Clériston Aparecido Gomes; BELLEZI, Marcos Augusto. **Análise de Vulnerabilidades com OpenVAS e Nessus**. T.I.S. São Carlos, v. 3, n. 1 , p. 34-44, jan-abr 2014

MCCLURE, Stuart; SCAMBRAY, Joel; KURTZ, George. **Hackers Expostos**. 7 ed. São Paulo: Bookman. 2014.

MEUCCI, M. **Owasp testing guide version 3.0**. OWASP Foundation. 2008.

MUNIZ, Joseph; LAKHANI, Aamir. **Web Penetration Testing with Kali Linux**. Birmingham: Packt. 2013.

OISSG. **Information Systems Security Assessment Framework**. Disponível em < www.oissg.org/issaf>. Acesso em 07 de Janeiro de 2015.

OLIVEIRA, Túlio. **Testes de Segurança em Aplicações Web segundo a metodologia OWASP**. Projeto de TCC. Universidade Federal de Lavras. 2012. Disponível em <http://www.bcc.ufla.br/wp-content/uploads/2013/09/TESTES-DE-SEGURAN%C3%87A-EM-APLICA%C3%87%C3%95ES-WEB-SEGUNDO-A.pdf>

OWASP. **10 Top Web Vulnerabilities**. Disponível em <www.owasp.org/index.php/Top_10_2013>. Acesso em 15 de Dezembro de 2014.

PAULI, Josh. **Introdução ao WebHacking**. São Paulo: Novatec, 2014.

PESSOA, Márcio. **Segurança em PHP**. São Paulo: Novatec, 2007.

ROCHA, Douglas; KREUTZ, Diego; TURCHETTI, Rogério. **Uma Ferramenta Livre e Extensível Para Detecção de Vulnerabilidades em Sistemas Web**. Disponível em <article.sapub.org/pdf/10.5923.j.computer.20120001.08.pdf >. Acesso em 21 de dezembro de 2014.

SCHILDT, Hebert; SKRIEN, Dale. (2013). **Programação com Java: Uma Introdução Abrangente**. São Paulo: McGraw-Hill.

SECTOOLS. **Top 125 security tools**. Disponível em<www.sectools.org>.Acesso em 07 de Janeiro de 2015.

SICA, Carlos; REAL, Petter. **Programação Segura utilizando PHP**. São Paulo: Ciência Moderna, 2007.SCHILDT, Hebert; SKRIEN, Dale. Programação com Java: Uma introdução abrangente. São Paulo: McGraw-Hill, 2013.

STALLINGS, William. **Redes e sistemas de comunicação de dados: teorias e aplicações corporativas**. 5 ed. Rio de Janeiro: Elsevier, 2005.

STALLINGS, William. **Criptografia e segurança em redes**. 4.ed. São Paulo: Person Prentice Hall, 2008.

WEB APPLICATION SECURITY CONSORTIUM . Disponível em: <http://www.webappsec.org/>. Acesso em: 01 de junho de 2015.

WELLING, Luke; THOMSON, Laura. **Tutorial MySQL**. Rio de Janeiro: Ciência Moderna, 2004.