

**UNIVERSIDADE NOVE DE JULHO**

Marcel Brilha Alves da Silva

Análise de eficiência na detecção de vulnerabilidades em sistemas web com o uso  
de ferramentas gratuitas e de código aberto

São Paulo  
2016

Marcel Brilha Alves da Silva

**Análise de eficiência na detecção de vulnerabilidades em sistemas web com o uso de ferramentas gratuitas e de código aberto**

Monografia apresentada à  
Universidade Nove de Julho – UNINOVE, como requisito  
parcial para obtenção do grau de Especialista em  
Sistemas e Desenvolvimento Web.

Prof. Antonio Carlos de Alcantara Thimoteo, Dr. - Orientador

São Paulo  
2016

## **AGRADECIMENTOS**

Gostaria de agradecer primeiramente a Deus por essa oportunidade de evoluir como profissional, aos meus pais que sempre me apoiaram emocionalmente e financeiramente, aos nobres companheiros de profissão que sempre estavam dispostos a me ensinar algo novo e a minha namorada Dennyse que sempre me apoiou nas horas difíceis.

Para finalizar, gostaria de agradecer humildemente aos queridos mestres que me ensinaram que um bom professor não é necessariamente aquele que detém o maior conhecimento, e sim aquele que ensina apaixonadamente o pouco que sabe e tem a humildade de aceitar que deve aprender um pouco mais a cada dia.

*“Todo o homem que encontro me é superior em alguma coisa. E, nesse particular,  
aprendo com ele.”*

**Ralph Waldo Emerson**

## RESUMO

Atualmente milhares de softwares são desenvolvidos e disponibilizados pela web, mas infelizmente devido à falta de conhecimento e despreparo dos profissionais de tecnologia, muitas aplicações são desenvolvidas com falta de segurança, onde o roubo das informações traz grande impacto nas empresas.

Este trabalho tem como finalidade identificar as principais vulnerabilidades web e avaliar a eficiência das ferramentas gratuitas e de código aberto para cada vulnerabilidade encontrada.

Foram realizados diversos experimentos em um ambiente de teste preparado, contra um sistema web que possui as principais vulnerabilidades encontradas atualmente, de acordo com o documento TOP TEN da OWASP.

As principais vulnerabilidades de acordo com a OWASP foram testadas, sendo elas: injeção de SQL, XSS, CSRF, interceptação de requisições, reconhecimento e ataque em senhas.

Utilizamos os softwares de código aberto disponíveis nas distribuições Kali Linux e BackTrack Linux, sendo eles: SQLMap, JSQL, Vega, ZAP OWASP, W3af, Skip Fish, BEEF, Burp Suite, WebScarab, HTTRACK, Hash Cat e John the Ripper.

Nos testes realizados, tivemos um grande destaque para as ferramentas SQLMap no quesito injeção de SQL, W3af no quesito scanner de vulnerabilidades e Beef no quesito XSS (Cross Site Scripting).

Palavras-chave: Aplicações Web; Segurança da Informação; PenetrationTest; Análise de Vulnerabilidades; Ferramentas Open Source.

## **ABSTRACT**

Currently thousands of software are developed and made available on the Web, but unfortunately due to lack of knowledge and lack of preparation technology professionals, many applications are developed with lack of security, where the theft of information brings great impact on business.

This work aims to identify the main web vulnerabilities and evaluate the efficiency of the free tools and open source for each vulnerability found.

Several experiments were conducted in a prepared test environment from a web system has the main current vulnerabilities found, according to the document OWASP TOP TEN.

The main vulnerabilities according to OWASP were tested, as follows: SQL injection, XSS, CSRF, interception requests, recognition and attack on passwords.

We use open source software available in Kali Linux and BackTrack Linux distributions, as follows: SQLMap, JSQL, Vega, OWASP ZAP, W3af, Skip Fish, BEEF, Burp Suite, WebScarab, HTTrack, Hash Cat and John the Ripper.

In tests, we had a great emphasis on SQLMap tools in the question SQL injection, W3af the vulnerability scanner Question and Beef in the category XSS (Cross Site Scripting).

**Keywords:** Web Applications; Information security; PenetrationTest; Vulnerability Analysis; Open Source tools.

## LISTA DE FIGURAS

Figura 1. Top 10 Vulnerabilidades Web OWASP.....	6
Figura 2. Fluxograma de passos adotados na pesquisa.....	11
Figura 3. DVWA sendo executado na máquina virtual.....	13
Figura 4. Tela principal Kali Linux.....	14
Figura 5. Interface gráfica do JSQL.....	16
Figura 6. Interface gráfica do VEGA.....	17
Figura 7. Interface gráfica BEEF.....	18
Figura 8. Interface Skipfish.....	19
Figura 9. Interface SQLMAP.....	20
Figura 10. Interface gráfica Burp Suite.....	21
Figura 11. Interface HashCat.....	22
Figura 12. Interface gráfica John the ripper.....	23
Figura 13. Interface HTTRACK.....	24
Figura 14. Interface gráfica OWASP Zap.....	25
Figura 15. Interface gráfica W3af.....	26
Figura 16. Interface gráfica WebScarab.....	27

## LISTA DE TABELAS

Tabela 1. Vulnerabilidades do framework DVWA.....	28
Tabela 2. Desempenho das ferramentas.....	29
Tabela 3. Ranking – Especialidade: Scanner de vulnerabilidades.....	30
Tabela 4. Ranking – Especialidade: Injeção de SQL.....	31
Tabela 5. Ranking – Especialidade: Reconhecimento e ataque a senhas.....	31



## SUMÁRIO

<b>1. INTRODUÇÃO .....</b>	<b>1</b>
1.1. Problema .....	1
1.2. Justificativa .....	2
1.3. Objetivo .....	2
 <b>2. REFERENCIAL TEÓRICO .....</b>	 <b>3</b>
2.1. Aplicações Web .....	3
2.1.1. Servidores Web .....	3
2.1.2. Protocolo HTTP .....	4
2.1.3. Linguagens de desenvolvimento Web .....	4
2.1.4. Banco de dados .....	4
2.2. Segurança da informação .....	5
2.2.1. Vulnerabilidades .....	5
2.3. As principais vulnerabilidades web de acordo com a OWASP .....	5
2.3.1. Injeção de código .....	6
2.3.2. XSS (Cross-Site Scripting).....	6
2.3.3. CSRF (Cross Site Request Forgery).....	7
2.3.4. Interceptação de requisições .....	7
2.3.5. Reconhecimento e ataque a senha .....	7
2.4. Pesquisas acadêmicas referente a ferramentas de vulnerabilidades Web..	8
 <b>3. METODOLOGIA .....</b>	 <b>10</b>
 <b>4. RESULTADOS.....</b>	 <b>13</b>
4.1. Construção do ambiente de testes .....	13
4.2. Seleção das ferramentas gratuitas e de código aberto .....	15
4.2.1. JSQL .....	16
4.2.2. VEGA .....	17
4.2.3. BEEF .....	18
4.2.4. Skipfish .....	19
4.2.5. SQLMap .....	20
4.2.6. BurpSuite .....	21
4.2.7. HashCat .....	22

4.2.8. John the Ripper .....	23
4.2.9. HTTRACK .....	24
4.2.10. OWASP Zap .....	25
4.2.11. W3af .....	26
4.2.12. WebScarab .....	27
4.3. Vulnerabilidades existentes no framework DVWA .....	28
4.4. Análise dos dados através das ferramentas selecionadas .....	29
4.5. Ranking das ferramentas mais eficientes .....	30
<b>5. CONSIDERAÇÕES FINAIS .....</b>	<b>32</b>
<b>REFERÊNCIAS .....</b>	<b>33</b>

## **1. INTRODUÇÃO**

Devido à grande utilização da internet, muitas empresas estão deixando de desenvolver suas aplicações para o ambiente desktop e estão adotando o desenvolvimento para as plataformas móveis e para a web.

Além das empresas, órgãos governamentais também disponibilizam seus serviços através da rede mundial de computadores, como inscrição em concursos, emissão de segunda via de documentos, etc.

Estas soluções começaram a ser adotadas devido à grande popularização da WWW (World Wide Web) nos últimos anos, entretanto, o grande desenvolvimento voltado para esse ambiente não veio acompanhado com a preocupação de desenvolvermos softwares seguros.

Assim como os softwares voltados para a plataforma desktop, os sistemas web também apresentam falhas de segurança que devem ser tratados e corrigidos, pois apresentam grandes impactos para as corporações, inclusive uma falha de segurança em um software web sofre agravante, devido ao fácil acesso a informações privilegiadas.

A proteção da informação é extremamente importante para garantirmos a continuidade do negócio, diminuir o impacto e prejuízo para os envolvidos, tanto no âmbito financeiro, quanto no âmbito intelectual.

É nesse contexto que está inserido o problema que se baseia essa pesquisa.

### **1.1. Problema**

As aplicações desenvolvidas para a web estão cada vez mais complexas, e devido ao crescimento acelerado das oportunidades de negócio, os softwares vêm sendo desenvolvidos de uma forma precária, sem preocupação com a segurança da informação, para combater esses riscos, surgem às ferramentas para identificarmos vulnerabilidades, examinando toda a estrutura da aplicação em busca de brechas de segurança.

Entretanto, a maioria dessas ferramentas, destinadas a esse tipo de problema, são proprietárias e tem um custo muito elevado para as pequenas e médias empresas, nesse cenário entram os softwares gratuitos e de código aberto como alternativa, para essas empresas que se preocupam em fazer um software de qualidade e seguro para os seus clientes, com foco nesse tema, surgem algumas questões que norteiam essa pesquisa: esses softwares gratuitos e de código aberto são eficientes na identificação de vulnerabilidades web?

## **1.2. Justificativa**

Devido à velocidade em que as tecnologias vêm morrendo, prazos extremamente curtos para a realização de um projeto, complexidade elevada, faz aumentar a probabilidade de falhas de segurança nos softwares desenvolvidos, tornando cada vez mais necessário o aprimoramento de novas metodologias, ferramentas, técnicas, para detecção e mitigação de riscos e vulnerabilidades.

Portanto, esse trabalho se justifica pela importância de identificar as principais vulnerabilidades web que mais são utilizadas por crackers no roubo de informação, e avaliar a eficiência das ferramentas gratuitas e de código aberto que são disponibilizadas para detecção e correção de falhas de segurança web.

## **1.3. Objetivo**

Encontrar as principais vulnerabilidades em softwares web e avaliar a eficiência de ferramentas gratuitas e de código aberto por tipo de falha encontrada.

## **2. REFERENCIAL TEÓRICO**

Este trabalho apresenta o seu referencial teórico dividindo-o em quatro partes, primeiramente serão apresentadas as informações sobre aplicações web, focando nas tecnologias que a compõem, de forma bem simples e sucinta, para que o leitor consiga ter um entendimento básico sobre o assunto.

A seguir iremos abordar a segurança da informação, com foco nos assuntos de invasão e vulnerabilidades, com destaque para alguns conceitos utilizados na pesquisa.

A terceira parte é voltada para a discussão das principais vulnerabilidades web, de acordo com o documento TOP 10 da OWASP.

Na última parte, falo um pouco da pesquisa acadêmica realizada por Marcos Flávio Araújo Assunção, referente à análise de eficiência de ferramentas open source na detecção de vulnerabilidades web e também falo de outras pesquisas realizadas.

Realizei a separação em partes, para que seja possível tornar mais claro os pontos fundamentais que sustentam a pesquisa.

### **2.1. Aplicações web**

Mas afinal, o que é uma aplicação web?

O termo aplicação web será utilizado ao longo dessa monografia referindo-se a qualquer aplicação (software) baseada na web, que acione funcionalidades (login, cadastrar produtos, etc) de acordo com uma entrada de usuário e que normalmente utilize tecnologias de back-end (PHP, JSP, ASP, etc) e grave as informações manipuladas pelo usuário em um banco de dados.

#### **2.1.1. Servidores web**

O servidor web nada mais é do que uma porção de software executado no sistema operacional de um servidor e que permite conexões para acessar uma aplicação web, responsáveis por aceitar requisições HTTP e HTTPS de clientes web e entregar respostas desses protocolos, normalmente em formas de página, esse conteúdo disponibilizado através da página é visualizado pelo usuário através de um browser (navegador web). Esses servidores possuem estruturas normais de diretório como qualquer outro computador e são esses diretórios que hospedam a aplicação web.

### **2.1.2. Protocolo HTTP**

Para que duas pessoas se comuniquem, elas precisam compreender a mesma língua, no caso de uma aplicação web, para que ela possa se comunicar precisa possuir o mesmo protocolo. O HTTP é um protocolo stateless (sem estados), de modo que toda solicitação de um usuário e a resposta da aplicação web corresponde a um evento totalmente independente, que não possui conhecimento das requisições anteriores.

Basicamente é o método utilizado para enviar e receber informações na web, o usuário através do navegador faz a requisição (request) de um determinado recurso, enviando um pacote de informações, contendo cabeçalhos a uma URL, o servidor recebe essa requisição e envia uma resposta (response), que pode ser um recurso ou outro cabeçalho.

### **2.1.3. Linguagens de desenvolvimento web**

Devido à necessidade atual do mercado de produzir conteúdo dinâmico, acessando banco de dados, consumindo outros sistemas, temos que utilizar linguagens de programação que disponibilizam flexibilidade para construir lógicas computacionais para resolver um problema em específico.

A maioria das linguagens utilizadas atualmente na web é baseada em scripts (PHP, JavaScript, Python, entre outras), que podem ser definidos como um conjunto de instruções que realizam determinadas tarefas, definidas por um desenvolvedor.

### **2.1.4. Banco de dados**

Banco de dados é um local no qual é possível armazenar dados de maneira estruturada. Estes dados poderão ser consultados por usuários distintos, através de permissões e softwares específicos. A linguagem mais utilizada atualmente para a comunicação entre os usuários e o banco de dados é a SQL (Structured Query Language).

## **2.2. Segurança da informação**

A informação é o bem mais precioso que uma corporação possui, a área de segurança da informação é responsável por proteger os dados e ativos da empresa contra vários tipos de ameaças.

Patrick Engebretson (2013) explica que os ativos de uma empresa são os elementos físicos, humanos ou tecnológicos que fazem parte do fluxo de informações da companhia.

O autor continua expondo que um sistema de segurança da informação é baseado em três pilares básicos: confidencialidade, integridade e disponibilidade.

### **2.2.1. Vulnerabilidades**

Josh Pauli (2014), vulnerabilidade é uma fraqueza que permite que um atacante reduza a garantia da informação de um sistema.

Vulnerabilidade é composta de três elementos: uma falha do sistema, acesso do atacante à falha e a capacidade do atacante de explorar a falha.

Uma pesquisa realizada pela Cenzic (WEB APPLICATION SECURITY TRENDS REPORT) no segundo semestre de 2009, 82% das vulnerabilidades registradas estavam em aplicações web e tecnologias relacionadas, com isso vemos a importância dessa monografia.

## **2.3. As principais vulnerabilidades web de acordo com a OWASP**

A OWASP (Web Application Security Project) é uma comunidade aberta, dedicada a capacitar as organizações, a desenvolver, adquirir e manter aplicações confiáveis.

O documento TOP 10 de vulnerabilidades web da OWASP abrange as principais vulnerabilidades atuais, mostra as consequências da exploração de cada uma dessas falhas, assim como algumas técnicas para se proteger contra elas.

A maioria das vulnerabilidades da lista ocorre durante a etapa de desenvolvimento do software, essa lista sofre alterações constantes, devido aos avanços feitos pelos atacantes.

Figura 1. Top 10 Vulnerabilidades Web OWASP

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

Fonte: [www.owasp.org](http://www.owasp.org) (2013).

Com base nesse documento que retirei algumas vulnerabilidades para testar as ferramentas gratuitas, irei falar um pouco mais sobre as principais, sendo elas:

### 2.3.1. Injeção de código

A falha de injeção de código infelizmente ainda é muito comum nas aplicações web, essa vulnerabilidade resulta do processo de envio de informações não confiáveis para a aplicação, geralmente por meio de uma requisição.

A aplicação web por não ter a segurança correta implementada, executa comandos inesperados em razão dos dados maliciosos enviados pelo atacante, fazendo com que o invasor tenha acesso a informações não autorizadas. Os ataques mais comuns de injeção têm como alvo: Comandos no sistema operacional, SQL (Structured Query Language), entre outros.

### 2.3.2. XSS (Cross-Site Scripting)

O XSS ocorre quando uma informação de entrada informada pelo usuário é aceita pela aplicação como parte de uma solicitação e, em seguida, é usada na apresentação da resposta, sem que haja uma codificação adequada de saída para validação e higienização.



Essa vulnerabilidade permite que atacantes executem scripts dentro do navegador da vítima, que pode roubar informações de cookies, sessões, senhas, até destruir o conteúdo apresentado na tela (Layout).

O XSS possui duas versões, o refletido e o armazenado, sendo o segundo o mais grave.

O refletido é considerado menos prejudicial devido ao fato de ser um ataque realizado uma única vez, em que a requisição enviada é válida somente para aquela requisição.

Já o armazenado é mais prejudicial porque persiste ao longo de várias solicitações e vários usuários, podendo ser armazenado em muitas vezes no banco de dados da aplicação.

### **2.3.3. CSRF (Cross Site Request Forgery)**

O CSRF ocorre quando um atacante é capaz de enviar uma solicitação adequadamente composta, para um usuário que está autenticado, essa solicitação possui variáveis que executam determinada função na aplicação, sem que a vítima sequer perceba. Alguns danos causados pelo CSRF são: alterações de senha, criações de novos usuários, entre outras.

### **2.3.4. Interceptação de requisições**

É o tipo de ataque em que as informações trocadas entre o usuário e a aplicação web são de alguma forma, interceptada, registrada e possivelmente alterada pelo atacante sem que a vítima perceba, o atacante pode desde alterar as solicitações, destruindo o conteúdo original submetido, até capturar informações geradas pela aplicação, devido estar entre a requisição do usuário e a aplicação.

### **2.3.5. Reconhecimento e ataque a senha**

É um tipo de ataque que visa de alguma forma reconhecer as senhas capturadas pelo atacante, seja através de força bruta ou pelo reconhecimento da senha através de bancos de dados de senhas criptografadas.

## **2.4. Pesquisas acadêmicas referente a ferramentas de vulnerabilidades web**

Uma pesquisa realizada por Marcos Flávio Araújo Assunção em 2015, com base no documento TOP 10 da OWASP se tornou grande referência para as empresas de segurança, a diferença dessa pesquisa realizada em 2015 para essa monografia, é que os testes realizados na época não consideraram os falsos negativos e falsos positivos para montar o ranking de ferramentas mais eficientes.

O falso negativo é quando a vulnerabilidade existe, mas não é detectado pela ferramenta, o falso positivo é quando a falha não existe, mas é apontada como falha pela ferramenta.

É importante considerar os falsos positivos e negativos, porque é uma falha da ferramenta de detecção de vulnerabilidades, que impacta diretamente nos testes de segurança de aplicações web.

O grande valor disponibilizado por essa pesquisa foi identificar que existem milhares de ferramentas para detecção de vulnerabilidades web, que muitos infelizmente ainda desconhecem, o grande problema é que foi realizada em 2015, com base nas ferramentas open source da distribuição Back Track Linux, atualmente já possuímos outras ferramentas mais específicas para cada vulnerabilidade, uma evolução disponibilizada pelo novo sistema operacional Kali Linux.

Outra pesquisa muito importante foi a de Armando Gonçalves da Silva Junior, baseada em testes nas ferramentas voltadas para a vulnerabilidade XSS (Cross Site Scripting), esse trabalho teve como objetivo o estudo sobre o estado da arte do Cross Site Scripting e o desenvolvimento de um ataque a um serviço de e-banking para comprovar que essa vulnerabilidade pode comprometer seriamente a segurança de qualquer página.

Essa pesquisa também foi um marco no ano de 2015, porque até então, muitos especialistas da área de segurança da informação conheciam a vulnerabilidade, mas não davam muita importância e não conseguiam visualizar os grandes impactos que essa vulnerabilidade pode provocar.

Para demonstrar o perigo dessa vulnerabilidade, Armando Gonçalves preparou um ambiente de teste com um software de e-banking real, onde ele realizou diversos ataques, no material desenvolvido, ele também comenta as maneiras de se prevenir.

A pesquisa realizada por Rogério Aparecido Campanari Xavier em 2010 também serviu como base para essa monografia, sua pesquisa foi voltada para auxiliar os desenvolvedores a verificarem o nível de segurança dos seus códigos,

ele baseou o seu trabalho na ferramenta OpenTracker, que permite a interação do desenvolvedor durante o processo de análise e desenvolvimento.

Para finalizar não poderia deixar de comentar da obra “Introdução ao Web Hacking” do doutor Josh Pauli, onde ele, através de uma linguagem simples, explica as diversas metodologias disponíveis para realizarmos os testes de invasão.

Através do material escrito por ele, conseguimos ter um conhecimento bem amplo sobre ataques web, com a abordagem bastante prática, todos os capítulos demonstram técnicas disponíveis através de ferramentas de segurança atuais.

Tirei desse livro, como criar um ambiente seguro para testar as ferramentas de segurança, descrições das ferramentas e como utilizar de forma correta todas as funcionalidades de cada ferramenta disponível.

### 3. METODOLOGIA

A segurança de aplicações é um tema que infelizmente não tem certa preocupação para entidades governamentais e privadas, devido aos impactos negativos que pode gerar, deveria ser o tema mais difundido dentro das corporações.

Este trabalho utiliza para seus fins, os estudos e pesquisas mais recentes sobre questões de segurança em sistemas web, onde posso destacar:

- **OWASP TOP 10:** Onde retirei as principais vulnerabilidades web encontradas atualmente no mercado.
- **SecTools.org:** onde utilizei para selecionar as principais ferramentas de segurança de código aberto, eles possuem uma listagem das cento e vinte e cinco ferramentas de segurança mais utilizadas e reconhecidas pela comunidade de segurança da informação, se baseando na opinião de mais de cinco mil profissionais da área.
- **2011 CWE/SANS Top 25 Most Dangerous Software Errors:** Lista detalhada demonstrando os principais erros de programação, que podem levar a sérias vulnerabilidades em um software.

A pesquisa realizada teve cunho qualitativo, utilizando os métodos bibliográficos e experimentais. De acordo com Marconi e Lakatos (2003), “metodologia de pesquisa é aquela que abrange o maior número de itens, pois responde, a um só tempo, às questões: Como? Com quê? Onde? Quanto?”.

Após explanação inicial realizada até aqui, apresento o cunho qualitativo utilizando como base pesquisa experimental, comparando as ferramentas selecionadas e verificando sua eficiência na detecção das principais vulnerabilidades web de acordo com a OWASP Top 10.

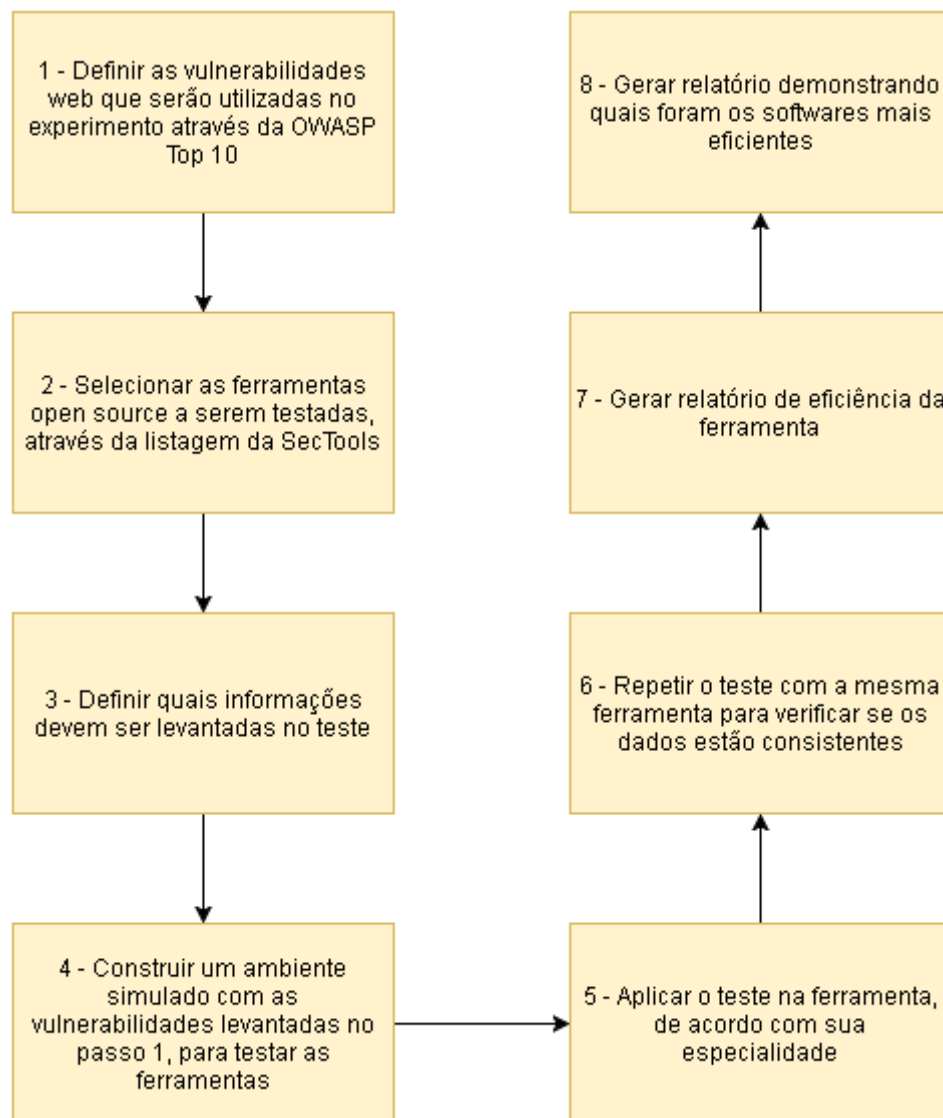
Para o teste, foram escolhidas as principais ferramentas open source do ranking da SecTools, durante o experimento, além dos softwares de teste selecionados, utilizei o DVWA que permite simular as vulnerabilidades encontradas atualmente pela OWASP. O DVWA é uma aplicação web concebida de forma vulnerável para auxiliar os profissionais da área a testarem softwares de segurança.

Através do teste experimental realizado, foram coletados os seguintes dados: quantidade de falhas detectadas, tempo total levado para detectar a falha e quantidade de falsos positivos e negativos.

Para concluir, a ferramenta mais eficiente será aquela que identificar o maior número de vulnerabilidades no ambiente, com a menor taxa de falsos positivos e negativos.

A pesquisa experimental seguiu o fluxograma exposto logo abaixo.

Figura 2. Fluxograma de passos adotados na pesquisa



Fonte: Elaborado pelo autor (2016).

A análise de resultados obtidos na etapa de testes experimentais da pesquisa é criteriosa, com base no total de vulnerabilidades identificadas, os falsos positivos e negativos serão apresentados na proporção encontrada após o teste e serão considerados para o estabelecimento do ranking.

Os dados obtidos foram determinantes para a formulação do ranking e servem como uma resposta para a questão inicial motivadora dessa pesquisa.

## 4. RESULTADOS

Nesta seção apresento todas as informações obtidas após a execução dos testes experimentais no framework DVWA, demonstrando todas as vulnerabilidades existentes no framework e o resultado obtido de cada ferramenta selecionada.

### 4.1. Construção do ambiente de testes

Para a realização dos testes, foi criada uma máquina virtual com o software de gerenciamento VMWARE Player, foi instalado o sistema operacional Windows 7 com o framework DVWA, esse framework fornece um conjunto de páginas com várias vulnerabilidades para que os profissionais de segurança da informação possam adquirir aprendizado prático e realizar testes.

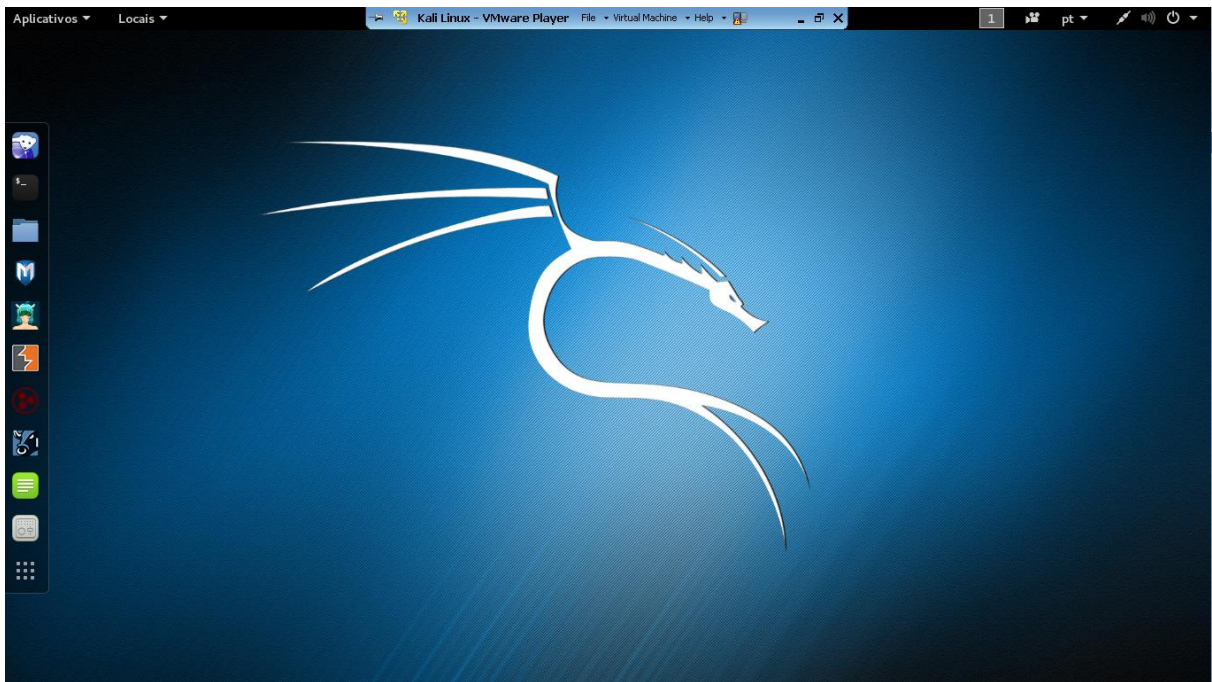
Figura 3. DVWA sendo executado na máquina virtual



Fonte: Print screen gerado pelo autor (2016).

Para realizar os ataques na aplicação DVWA e analisarmos a eficiência das ferramentas open source, foi configurada outra máquina virtual com o sistema operacional Kali Linux, versão 2016.2 com todos os softwares já instalados.

Figura 4. Tela principal Kali Linux.



Fonte: Print screen gerado pelo autor (2016).

Após a instalação e configuração das duas máquinas virtuais, realizei alguns testes para verificar se a máquina atacante (Kali Linux), estava conseguindo ter acesso a máquina vítima (Windows 7), os testes foram positivos e o acesso foi estabelecido corretamente.

Na próxima seção mostro como foi o processo de escolha dos softwares a serem testados, falo um pouco das ferramentas utilizadas no trabalho, apresentando um pouco de suas características básicas e principais recursos.



## **4.2. Seleção das ferramentas gratuitas e de código aberto**

O processo de escolha das ferramentas que iriam participar desse experimento foi feita através dos critérios expostos abaixo:

1. Minhas experiências com a ferramenta no mercado de trabalho;
2. Citação de especialistas;
3. Posição do ranking SecTools;

O primeiro critério foi escolher as ferramentas que eu mais tenho familiaridade e que sei que funcionam, devido às diversas experiências em projetos de segurança, desse levantamento selecionei as ferramentas JSQL para injeção de código, VEGA para scanner de vulnerabilidades e BEEF para Cross Site Scripting.

Do segundo critério, os trabalhos de Josh Pauli (2014) e Engebretson (2013) citam também o uso das ferramentas de código aberto Skipfish e SQLMAP como uma solução recomendada na análise de vulnerabilidades de injeção de código e ataques de Cross Site Scripting, do workshop Ethical Hacking Foundation de Patrick de Brouwer (2016) retirei as ferramentas Burp Suite para interceptação de requisições, Hash Cat e John the ripper para ataque a senhas e HTTRACK para capturar o código da aplicação web.

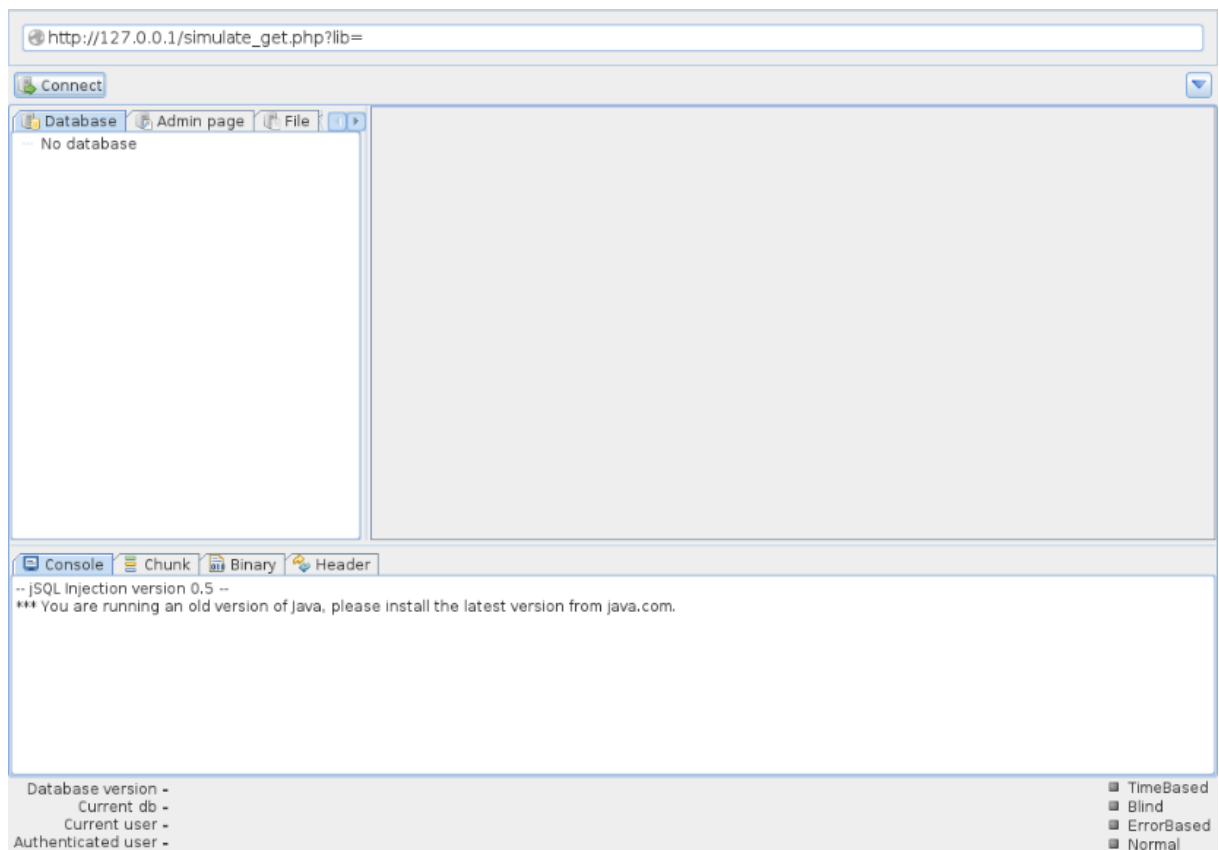
O terceiro e mais importante critério utilizado, foi a verificação da posição das ferramentas através do site SecTools, que faz uma enquete quantitativa para estabelecer um ranking das soluções de software mais populares, retirei desse critério as ferramentas ZAP Owasp, W3af e WebScarab para scanner de vulnerabilidades.

### 4.2.1. JSQL

JSQL é um aplicativo leve usado para encontrar informações de banco de dados em servidores distantes, sua maior funcionalidade é permitir a análise e injeção de SQL.

Possui uma interface amigável para o usuário, onde permite visualizar de uma forma simples os bancos de dados, injeções SQL utilizadas para a invasão, as tabelas e os registros.

Figura 5. Interface gráfica do JSQL



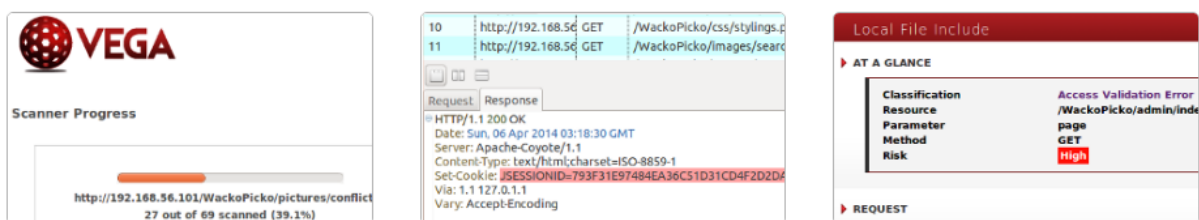
Fonte: Print screen gerado pelo autor (2016).

#### 4.2.2. Vega

Vega é um software de plataforma livre e aberta, sua principal funcionalidade é verificar e testar a segurança de aplicações web.

O VEGA pode ajudar a encontrar e validar vulnerabilidades como Injeção de SQL, Cross Site Scripting, entre outros. Ele é escrito em Java, baseado em GUI, roda em Linux, OSX e Windows.

Figura 6. Interface gráfica do VEGA



Fonte: Montagem gerada pelo autor (2016).

### 4.2.3. BEEF

BEEF é uma ferramenta de testes de penetração que se concentra no navegador web.

Sua principal funcionalidade é sua interface amigável para a realização de testes voltados para a vulnerabilidade Cross Site Scripting, disponível para Linux e Windows, essa ferramenta disponibiliza mais de 100 scripts para testes, incluindo injeção de scripts para roubo de sessões, cookies, entre outros.

Figura 7. Interface gráfica BEEF



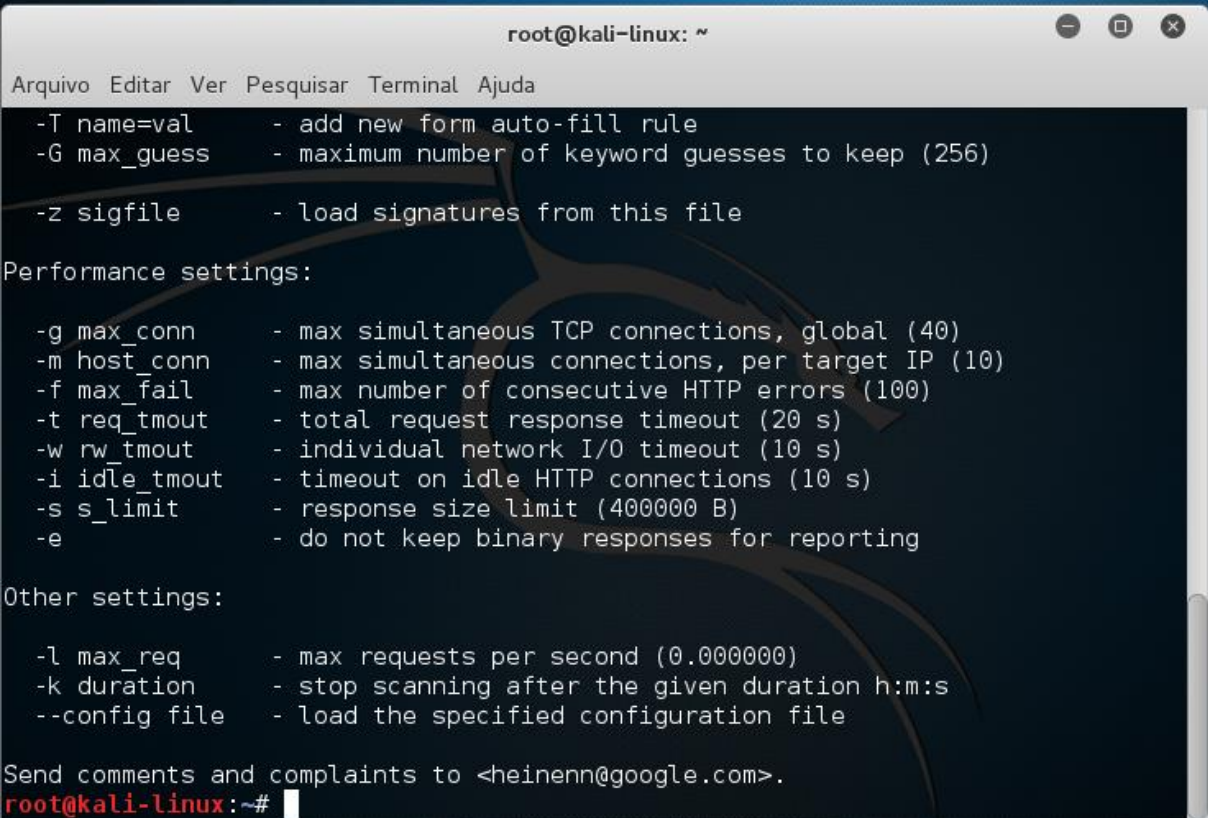
Fonte: Print screen gerado pelo autor (2016).

#### 4.2.4. Skipfish

Skipfish é uma ferramenta open source de análise de vulnerabilidades em ambiente web que permite a identificação de arquivos através de um crawl recursivo e a criação de um mapa completo de todos os recursos encontrados no alvo analisado.

Josh Pauli (2014) expõe que o programa permite a geração de relatórios personalizados com todas as informações descobertas em diversos formatos de arquivos diferentes.

Figura 8. Interface Skipfish

A screenshot of a terminal window titled 'root@kali-linux: ~'. The window contains the Skipfish help text, which lists various command-line options and their functions. The options are grouped into 'Performance settings:' and 'Other settings:'. The terminal text is as follows:

```
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
-T name=val      - add new form auto-fill rule
-G max_guess     - maximum number of keyword guesses to keep (256)
-z sigfile       - load signatures from this file

Performance settings:
-g max_conn      - max simultaneous TCP connections, global (40)
-m host_conn     - max simultaneous connections, per target IP (10)
-f max_fail      - max number of consecutive HTTP errors (100)
-t req_tmout     - total request response timeout (20 s)
-w rw_tmout      - individual network I/O timeout (10 s)
-i idle_tmout    - timeout on idle HTTP connections (10 s)
-s s_limit       - response size limit (4000000 B)
-e              - do not keep binary responses for reporting

Other settings:
-l max_req       - max requests per second (0.000000)
-k duration      - stop scanning after the given duration h:m:s
--config file    - load the specified configuration file

Send comments and complaints to <heinenn@google.com>.
root@kali-linux:~#
```

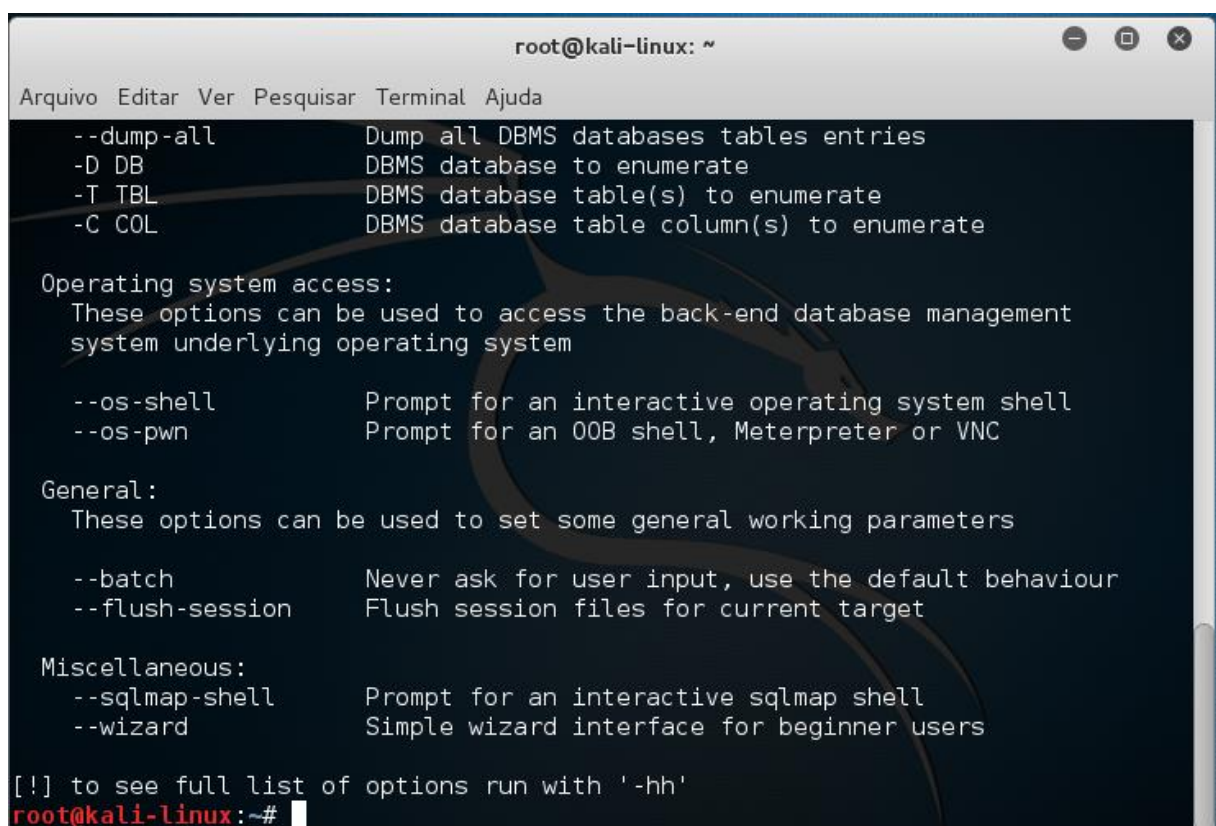
Fonte: Print screen gerado pelo autor (2016).

#### 4.2.5. SQLMap

O SQLMap é uma solução open source, desenvolvida com a linguagem de programação Python, e de acordo com Torres, a ferramenta “automatiza o processo de detecção e exploração de falhas de SQL Injection”.

O objetivo da utilização do software é a obtenção de acesso direto a bancos de dados por meio de aplicações web vulneráveis.

Figura 9. Interface SQLMap



```
root@kali-linux: ~
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda

--dump-all      Dump all DBMS databases tables entries
-D DB          DBMS database to enumerate
-T TBL         DBMS database table(s) to enumerate
-C COL         DBMS database table column(s) to enumerate

Operating system access:
These options can be used to access the back-end database management
system underlying operating system

--os-shell      Prompt for an interactive operating system shell
--os-pwn       Prompt for an OOB shell, Meterpreter or VNC

General:
These options can be used to set some general working parameters

--batch        Never ask for user input, use the default behaviour
--flush-session Flush session files for current target

Miscellaneous:
--sqlmap-shell Prompt for an interactive sqlmap shell
--wizard       Simple wizard interface for beginner users

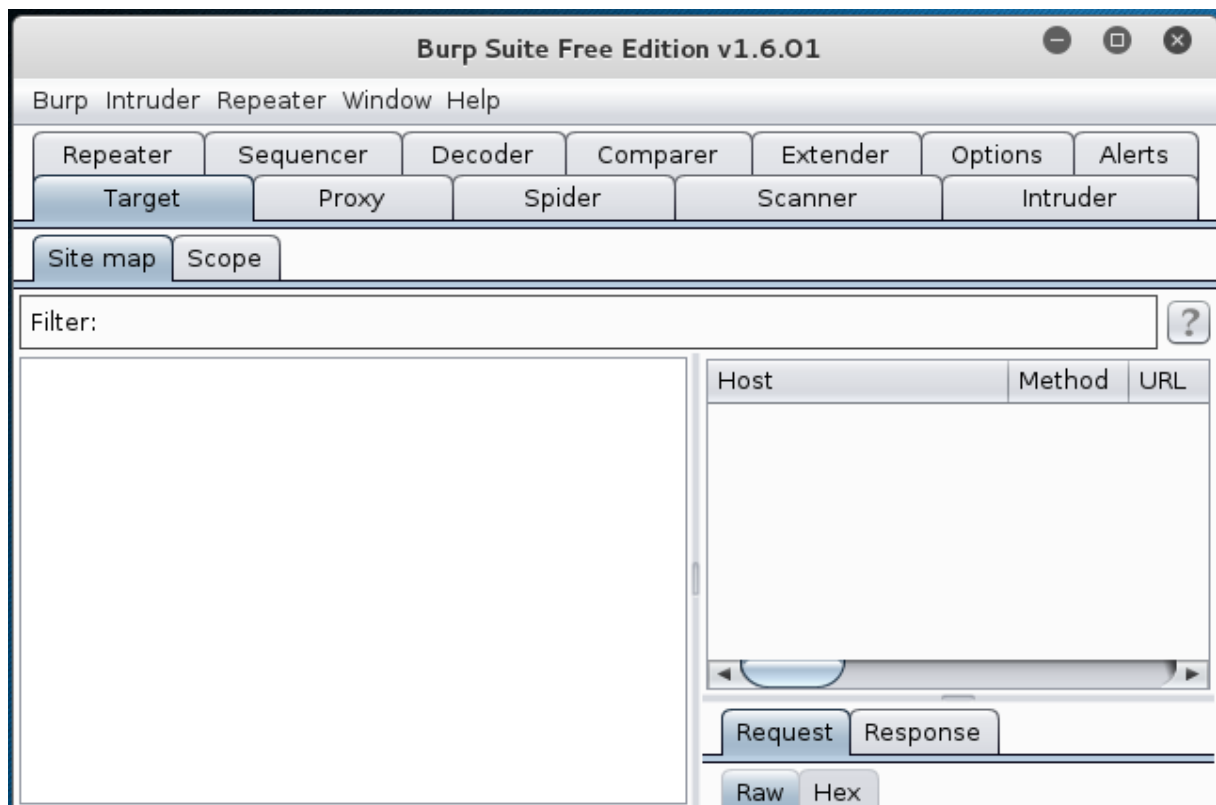
[!] to see full list of options run with '-hh'
root@kali-linux:~#
```

Fonte: Print screen gerado pelo autor (2016).

#### 4.2.6. Burp Suite

Burp Suite é uma ferramenta construída sobre a plataforma de software JAVA, sua principal funcionalidade é a realização de testes de segurança em aplicações web do tipo interceptação de requisições, podendo ser utilizado de forma automatizada ou manual.

Figura 10. Interface gráfica Burp Suite



Fonte: Print screen gerado pelo autor (2016).

#### 4.2.7. HashCat

HashCat é uma ferramenta livre e de código aberto, está disponível para os sistemas operacionais Linux, OSX e Windows.

Sua principal funcionalidade é explorar versões de hash de senhas, muito utilizado quando o cracker descobre a senha criptografada, mas precisa da senha sem o hash.

Figura 11. Interface HashCat

```
Initializing hashcat v0.37 by atom with 8 threads and 32mb segment-size...
NOTE: press enter for status-screen

Added hashes from file C:/HashCat/hashes.txt: 1 (1 salts)
Activating quick-digest mode for single-hash
Charset....: abcdefghijklmnopqrstuvwxyz0123456789
Length....: 1
Index.....: 0/1 (segment), 36 (words), 0 (bytes)
Recovered..: 0/1 hashes, 0/1 salts
Speed/sec..: - plains, - words
Progress...: 36/36 (100.00%)
Running....: ---:---:---
Estimated..: ---:---:---
Charset....: abcdefghijklmnopqrstuvwxyz0123456789
Length....: 2
Index.....: 0/1 (segment), 1296 (words), 0 (bytes)
Recovered..: 0/1 hashes, 0/1 salts
Speed/sec..: - plains, - words
Progress...: 1296/1296 (100.00%)
Running....: ---:---:---
Estimated..: ---:---:---
Charset....: abcdefghijklmnopqrstuvwxyz0123456789
Length....: 3
Index.....: 0/1 (segment), 46656 (words), 0 (bytes)
Recovered..: 0/1 hashes, 0/1 salts
Speed/sec..: - plains, - words
Progress...: 46656/46656 (100.00%)
Running....: ---:---:---
Estimated..: ---:---:---
dcb8e94ac7d0aad8a81d9c895ace5f4:fred
All hashes have been recovered
```

Fonte: Print screen gerado pelo autor (2016).

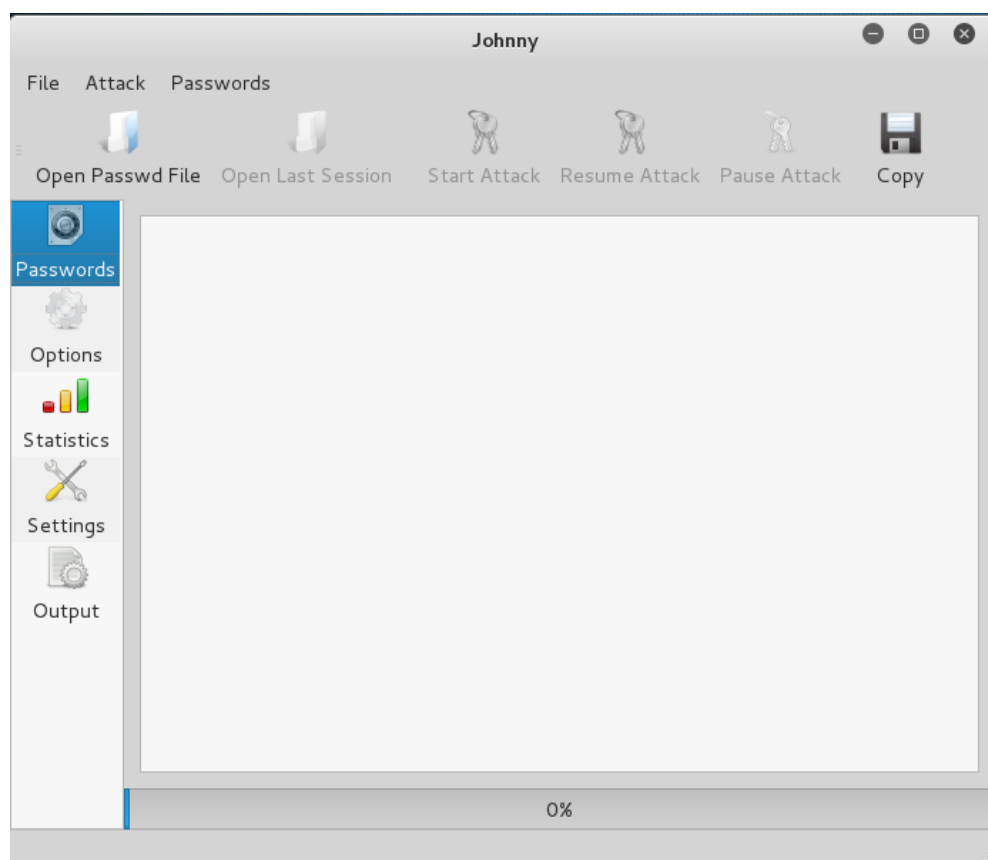


#### 4.2.8. John the Ripper

John the Ripper é um software de código aberto e gratuito, muito utilizado para quebra de senhas. Inicialmente desenvolvido para os sistemas Unix, agora disponibilizado também em Windows e OSX.

Essa ferramenta é capaz de fazer força bruta em senhas cifradas em DES, MD4 e MD5.

Figura 12. Interface gráfica John the Ripper

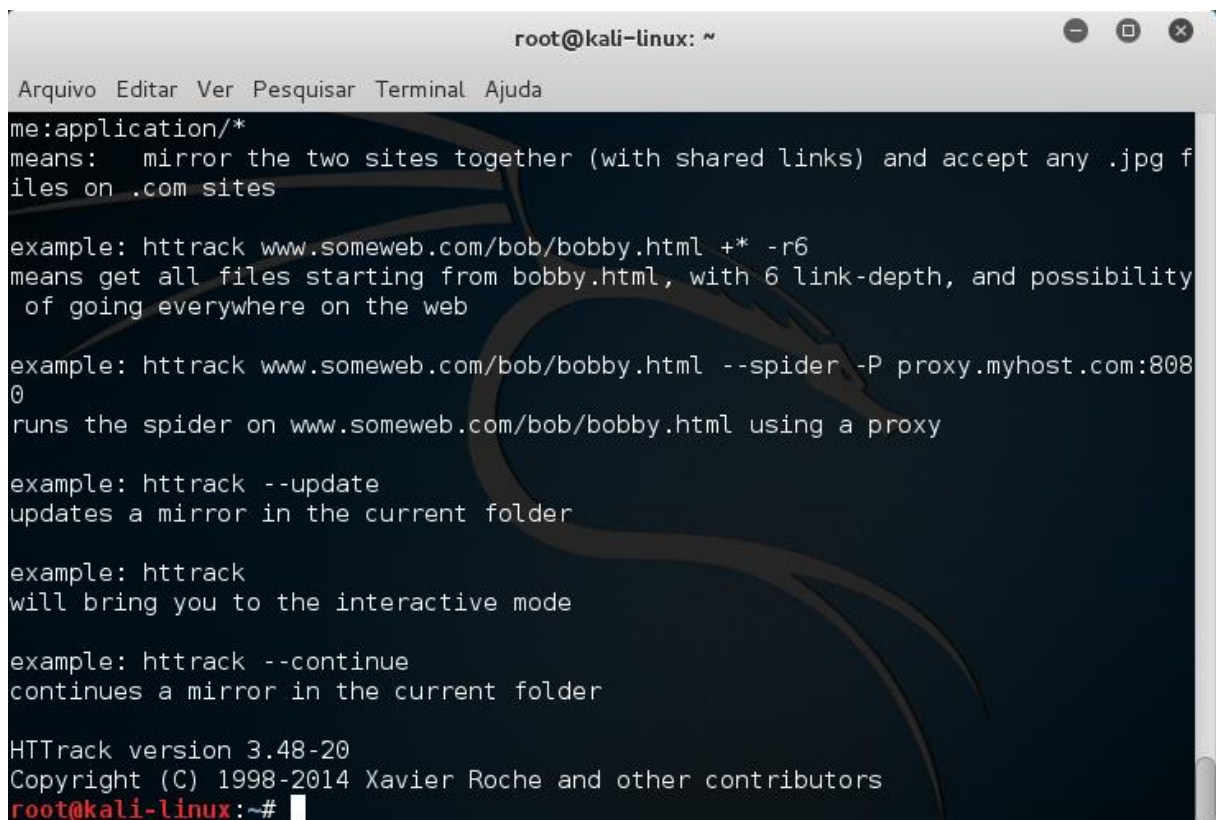


Fonte: Print screen gerado pelo autor (2016).

#### 4.2.9. HTTRACK

HTTRACK é uma ferramenta livre e de código aberto que permite o download de aplicações web para o computador local, com essa ferramenta você consegue visualizar de forma completa o código fonte da aplicação.

Figura 13. Interface HTTRACK



```
root@kali-linux: ~
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
me:application/*
means:  mirror the two sites together (with shared links) and accept any .jpg f
iles on .com sites

example: httrack www.someweb.com/bob/bobby.html +* -r6
means get all files starting from bobby.html, with 6 link-depth, and possibility
of going everywhere on the web

example: httrack www.someweb.com/bob/bobby.html --spider -P proxy.myhost.com:808
0
runs the spider on www.someweb.com/bob/bobby.html using a proxy

example: httrack --update
updates a mirror in the current folder

example: httrack
will bring you to the interactive mode

example: httrack --continue
continues a mirror in the current folder

HTTrack version 3.48-20
Copyright (C) 1998-2014 Xavier Roche and other contributors
root@kali-linux:~#
```

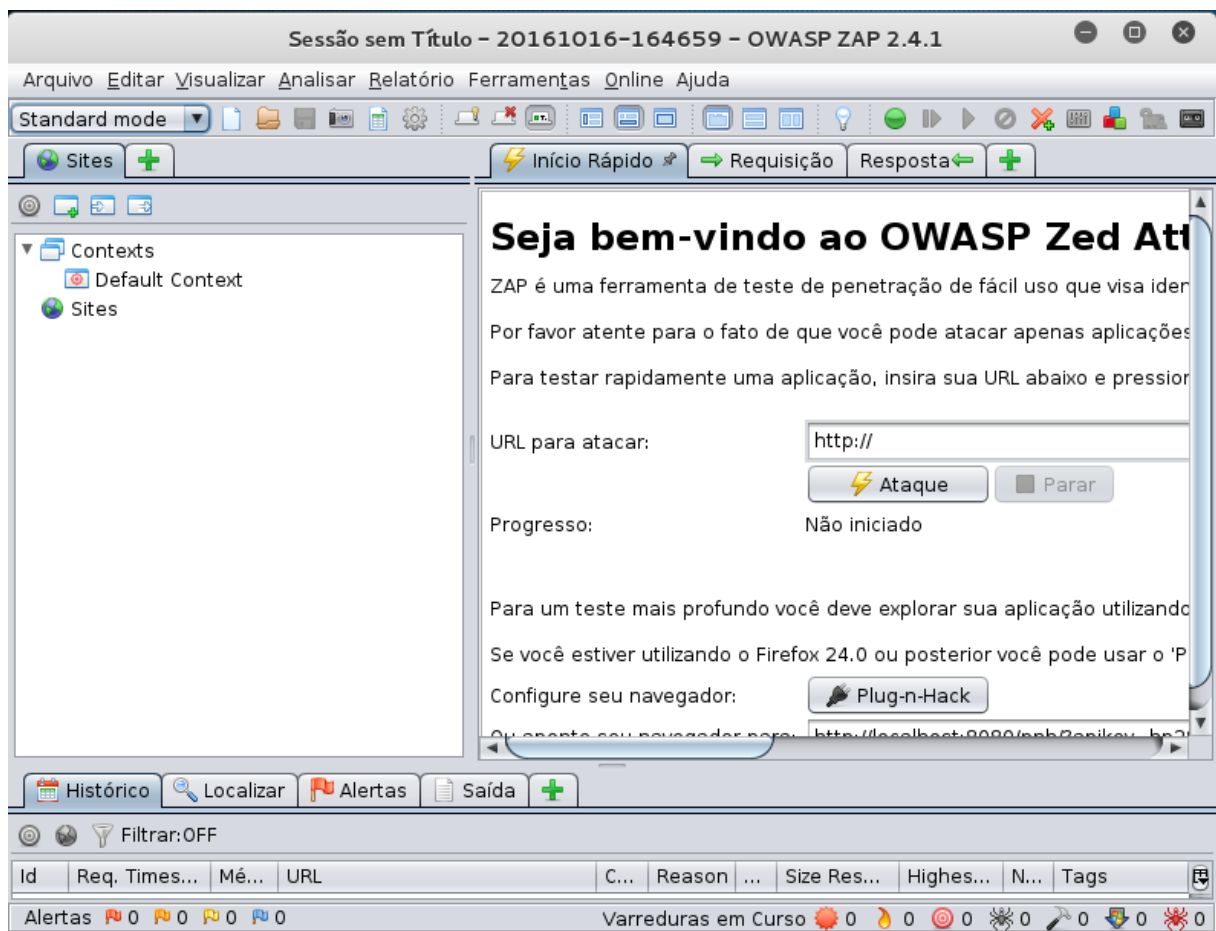
Fonte: Print screen gerado pelo autor (2016).

#### 4.2.10. OWASP Zap

O OWASP Zap é uma ferramenta open source criada pela OWASP, que age como um servidor proxy com a finalidade de detectar vulnerabilidades em aplicações web.

Este é um dos projetos mais ativos da OWASP, já tendo sido traduzido para mais de 25 linguagens diferentes.

Figura 14. Interface gráfica OWASP Zap

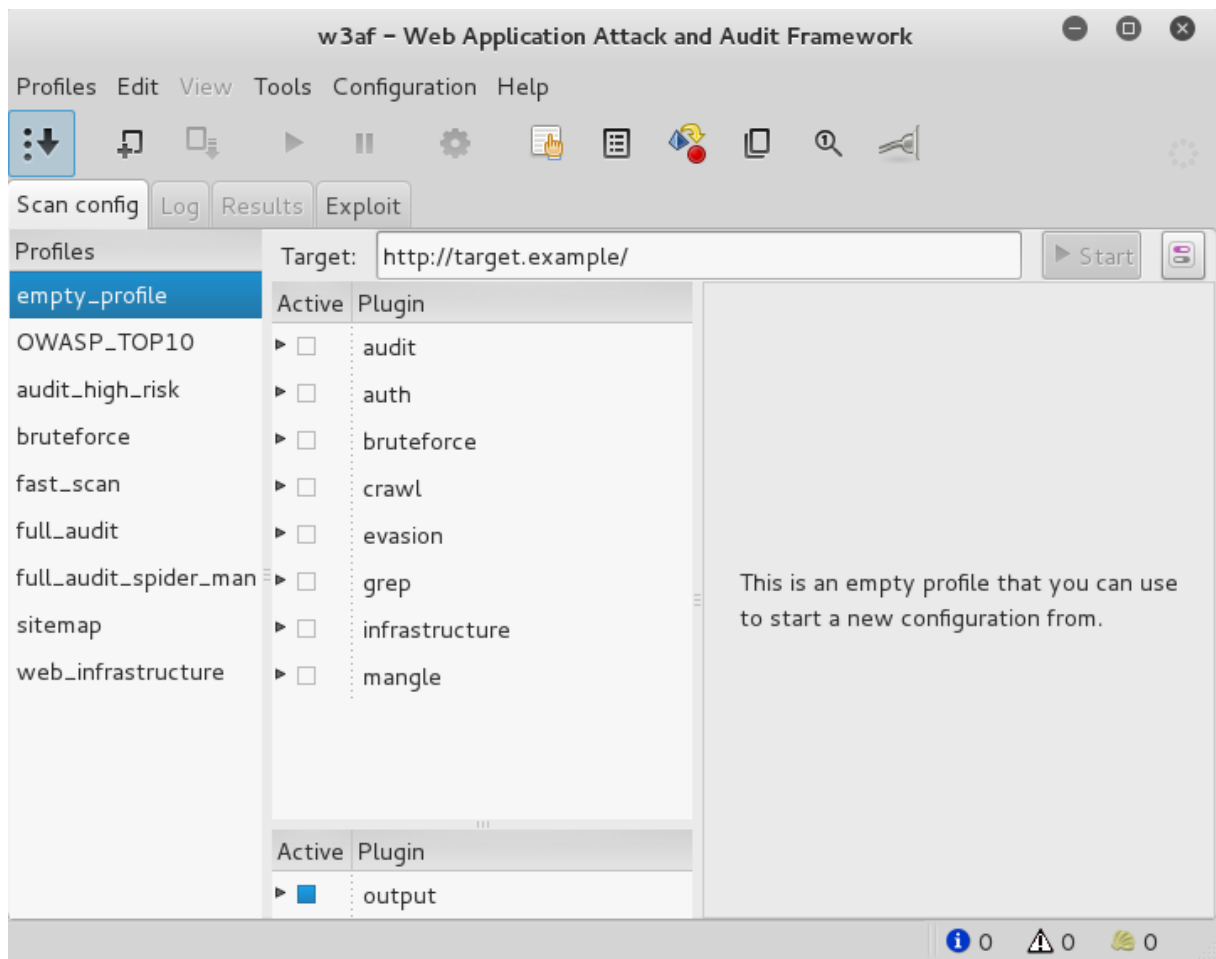


Fonte: Print screen gerado pelo autor (2016).

#### 4.2.11. W3af

W3af é uma ferramenta gratuita e de código aberto, sua principal funcionalidade é permitir os testes de vulnerabilidades contra servidores web, assim o software tem como objetivo fazer a varredura e identificação de falhas em ambiente web.

Figura 15. Interface gráfica W3af

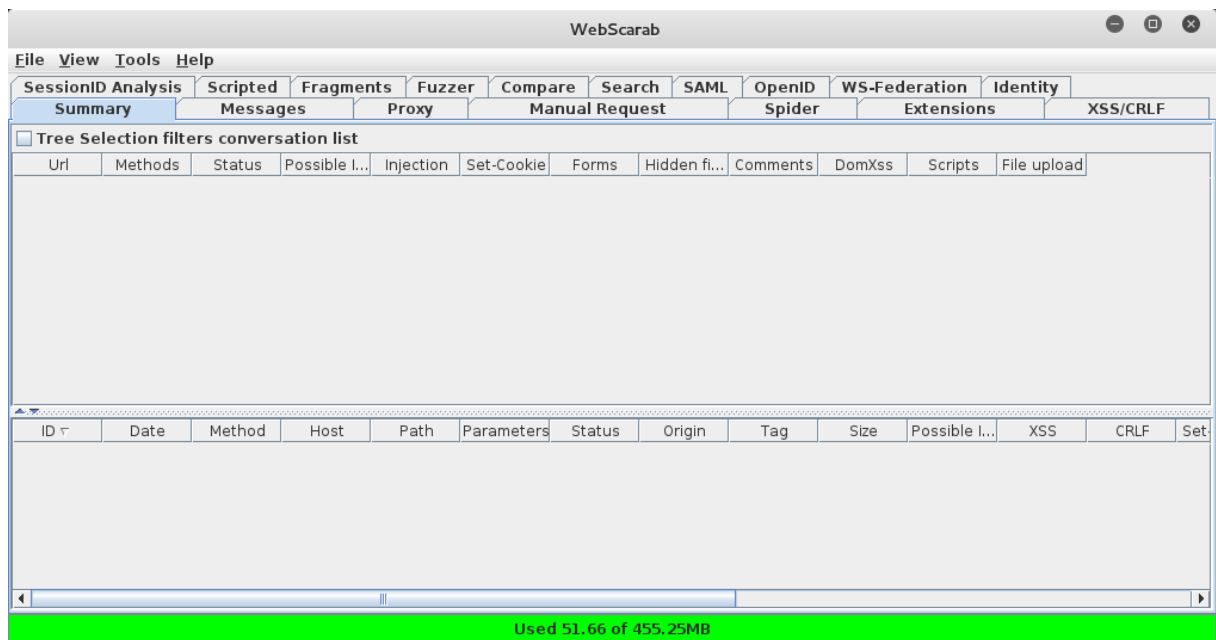


Fonte: Print screen gerado pelo autor (2016).

#### 4.2.12. WebScarab

WebScarab é uma ferramenta gratuita e de código aberto, muito utilizada para testes em aplicações web, serve como um Proxy que intercepta requisições realizadas por algum usuário, registrando todo o tráfego realizado para uma posterior avaliação.

Figura 16. Interface gráfica WebScarab



Fonte: Print screen gerado pelo autor (2016).

### 4.3. Vulnerabilidades existentes no framework DVWA

Nesta seção demonstro todas as vulnerabilidades existentes no framework DVWA, essas informações foram utilizadas como base para a realização dos testes.

Tabela1. Vulnerabilidades do framework DVWA

<b>Tipo de vulnerabilidade</b>	<b>Total de falhas</b>
XSS (Cross Site Scripting)	15
SQL Injection	10
CSRF (Cross Site Request Forgery)	4
Falhas de interceptação de requisições	8
Ataque a hash de senhas	9

Fonte: Tabela criada pelo autor (2016).

#### 4.4. Análise dos dados através das ferramentas selecionadas

Nesta seção da monografia, são apresentados todos os resultados das ferramentas selecionadas, através da execução de testes de segurança no framework DVWA.

Tabela 2. Desempenho das ferramentas

<b>Ferramenta</b>	<b>Quantidade de falhas existentes</b>	<b>Quantidade de falhas detectadas</b>	<b>Tempo total levado</b>	<b>Falsos positivos</b>	<b>Falsos negativos</b>
JSQL	10	5	15 minutos	0	0
Vega	46	40	30 minutos	5	0
BEEF	15	15	20 minutos	0	0
Skipfish	46	35	30 minutos	10	0
SQLMap	10	10	10 minutos	0	0
Burp Suite	8	8	5 minutos	0	0
HashCat	9	9	10 minutos	0	0
John ripper	9	8	15 minutos	0	0
HTTRACK	4	3	4 minutos	0	0
OWASP Zap	46	40	28 minutos	2	0
W3af	46	46	25 minutos	0	0
WebScarab	46	35	38 minutos	15	4

Fonte: Tabela criada pelo autor (2016).

Analisando o desempenho das ferramentas demonstradas logo acima, podemos dar um grande destaque para as ferramentas SQLMap no quesito injeção de SQL, W3af no quesito scanner de vulnerabilidades e BEEF no quesito XSS (Cross Site Scripting).

#### 4.5. Ranking das ferramentas mais eficientes

Nesta seção apresento o ranking das ferramentas mais eficientes após testes executados no framework DVWA, separando por especialidade da ferramenta.

Lembrando que os falsos positivos e negativos foram considerados para a montagem do ranking.

Tabela 3. Ranking – Especialidade: Scanner de vulnerabilidades

<b>Colocação</b>	<b>Nome da ferramenta</b>
Primeira colocação	W3af
Segunda colocação	OWASP Zap
Terceira colocação	Vega
Quarta colocação	Skipfish
Quinta colocação	WebScarab

Fonte: Tabela criada pelo autor (2016).



Tabela 4. Ranking – Especialidade: Injeção de SQL

<b>Colocação</b>	<b>Nome da ferramenta</b>
Primeira colocação	SQLMap
Segunda colocação	JSQL

Fonte: Tabela criada pelo autor (2016).

Tabela 5. Ranking – Especialidade: Reconhecimento e ataque a senhas

<b>Colocação</b>	<b>Nome da ferramenta</b>
Primeira colocação	HashCat
Segunda colocação	John the ripper

Fonte: Tabela criada pelo autor (2016).

As ferramentas BEEF, Burp Suite e HTTRACK não precisam de ranking devido a terem competido sozinhas nas suas especialidades.

## 5. CONSIDERAÇÕES FINAIS

Iniciei esse trabalho acadêmico com algumas questões levantadas após um problema de mercado muito comum entre as pequenas e médias empresas, que era se as ferramentas gratuitas e de código aberto eram realmente eficientes na detecção de vulnerabilidades web.

Após os testes realizados, consegui demonstrar e comprovar que sim, que as ferramentas além de serem eficientes, trazem um grande ganho agregado para as pequenas e médias empresas, diminuindo os custos com ferramentas proprietárias de detecção de vulnerabilidades web.

Posso destacar como grande contribuição dessa pesquisa, a identificação das ferramentas de código aberto que são mais eficientes na detecção de vulnerabilidades web, demonstrando os pontos fortes e fracos de cada ferramenta, lembrando que se tratam de ferramentas open source, que podem ter seu código fonte melhorado, assim, contribui também para a comunidade de software livre, voltada para as ferramentas de segurança da informação.

Outro ponto muito importante abordado por essa pesquisa acadêmica, foi considerar os falsos positivos e negativos para a montagem do ranking das ferramentas mais eficientes, até o momento, nenhum pesquisador realizou esse experimento considerando esses pontos.

No final consegui ter uma amostra clara das ferramentas mais eficientes para serem utilizadas em problemas reais no mercado de trabalho, onde não deixam nada a desejar, até porque em muitos pontos, as ferramentas demonstraram ser bastante eficientes na detecção de vulnerabilidades web, considerando o tempo total levado para a realização dos testes, muitas das ferramentas detectaram de 90% a 100% das vulnerabilidades contempladas no ambiente de teste criado.

Para finalizar não poderia deixar de comentar sobre as ferramentas SQLMap, BEEF e W3af, que se destacaram entre as demais, conseguindo identificar 100% das vulnerabilidades encontradas no ambiente de teste controlado com o framework DVWA.

## REFERÊNCIAS

ENGBRETSON, 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.

OWASP. Top 10 Web Vulnerabilities ([www.owasp.org/index.php/Top\\_10\\_2013](http://www.owasp.org/index.php/Top_10_2013)). Acesso em 18 de Fevereiro de 2016.

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

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

SECTOOLS. **Top 125 security tools**. ([www.sectools.org](http://www.sectools.org)). Acesso em 18 de Fevereiro de 2016.

APARECIDO, Rogério. **Sistema de Rastreamento de Vulnerabilidades em Aplicações Web: Ferramenta OpenTracker**. São Paulo: 2010.

ASSUNÇÃO, Marcos. **Análise de eficiência na detecção de vulnerabilidades em ambientes Web com o uso de ferramentas de código aberto**. Minas Gerais: 2015.

GONÇALVES, Armando. **Cross-Site Scripting: Uma Análise Prática**. Pernambuco: 2009.

BROUWER, Patrick. **Ethical Hacking Foundation – Exin Workshop**. São Paulo: 2016.

CWE/SANS. **Top 25 Most Dangerous Software Errors**. (<http://cwe.mitre.org/top25>). Acesso em 18 de Fevereiro de 2016.