UNIVERSIDADE ESTADUAL DO NORTE DO PARANÁ

***CAMPUS* LUIZ MENEGHEL**

DOUGLAS DA SILVA MACHADO

FERRAMENTA PARA ANÁLISE E DIAGNÓSTICO DE VULNERABILIDADES EM SITES WEB

Bandeirantes
2016

Douglas da Silva Machado

FERRAMENTA PARA ANÁLISE E DIAGNÓSTICO DE VULNERABILIDADES EM SITES WEB

Trabalho de Conclusão de Curso submetido como requisito parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Prof. Me. Luiz Fernando Legore do Nascimento.

Bandeirantes
2016

Douglas da Silva Machado

FERRAMENTA PARA ANÁLISE E DIAGNÓSTICO DE VULNERABILIDADE EM SITES WEB.

Trabalho de Conclusão de Curso submetido como requisito parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

COMISSÃO EXAMINADORA

_____
Prof. Me. Luiz Fernando L. do Nascimento
UENP – *Campus* Luiz Meneghel

_____
Prof. Me. Mauricio Massaru Arimoto
UENP – *Campus* Luiz Meneghel

_____
Prof. Me. Ricardo Gonçalves Coelho
UENP – *Campus* Luiz Meneghel

Bandeirantes, 21 de julho de 2016

Dedico este trabalho a minha mãe, uma das pessoas que me ajudaram a chegar até aqui, e também a minha namorada, que em momentos difíceis me apoiou e me deu forças para continuar.

AGREDECIMENTOS

Gostaria de deixar aqui meus agradecimentos a todos que fizeram parte direta e indiretamente de tudo que eu passei nesses anos, anos esses onde acrescentei muito conhecimento e fiz grandes amigos. Agradeço a todos os professores pela conhecimento passado e toda calma que tiveram na minha formação.

A minha mãe Ivonete Aparecida, que foi essencial para o desenvolvimento do que eu sou hoje, e que me dá forças para continuar em busca do melhor.

A minha namorada Fabricia Regina por estar comigo nos piores momentos e também nos melhores momentos, auxiliando e me ajudando a superar os momentos ruins e me trazendo muitos momentos bons.

Aos amigos que criei nessa caminhada e que levarei pro resto da vida.

E para todos aqueles que direta ou indiretamente contribuíram para a minha formação, meu muito obrigado!

“A mudança não virá se esperarmos por outra pessoa ou outros tempos. Nós somos aqueles por quem estávamos esperando. Nós somos a mudança que procuramos” (Barack Obama).

RESUMO

A internet tem se tornado principal fonte de negócios e trabalho de pessoas e empresas de diversas tarefas, dado ao seu alcance em milhões de pessoas em tempo curto. Isto se deve ao uso de aplicações Web para tarefas como compras, negociações, acesso ao banco, inscrições em concursos e eventos, compartilhamento de documentos e recursos entre outras, aplicações assim necessitam de segurança pois trabalham com dados de clientes, de usuários e de organizações sendo a informação a parte mais importante de uma empresa. Desta forma este trabalho buscou abordar a segurança em aplicações Web, com foco em vulnerabilidades como: ***L****ocal* ***F****ile* ***I****nclusion,* ***R****emote* ***F****ile* ***I****nclusion, Sql Injection, Directory Traversal e Xss.* Reunindo na ferramenta denominada Arcadio com bases nos testes realizados em um ambiente simulado os melhores scanners para essa ferramenta, os scanner escolhidos são: Fimap, Uniscan, Xsser e Nikto, os critérios de seleção encontra-se na sessão 5.2 Ambiente Simulado. A ferramenta tem como objetivo facilitar a análise de aplicações para desenvolvedores iniciantes ou qualquer pessoa que tenha interesse em segurança da informação.

Palavras-chave: Segurança web 1; Análise de Vulnerabilidade 2; Segurança da Informação 3.

ABSTRACT

The internet has become the main source of business and working people and companies several tasks given within reach millions of people in a short time . This is due to the use of Web applications for tasks such as shopping negotiations, database access, registration in competitions and events, documents and resource sharing among others, so applications need security because they work with customer data, users and organizations with the information the most important part of a company. This work sought to address security in Web applications, focusing on vulnerabilities as: ***L****ocal* ***F****ile* ***I****nclusion,* ***R****emote* ***F****ile* ***I****nclusion, Sql Injection, Directory Traversal e Xss.* Gathering in tool called Arcadio with bases in tests in a simulated environment the best scanners for this tool, the selected scanner are: Fimap, Uniscan, Xsser and Nikto, the selection criteria is in session 5.2 Simulated Environment The tool aims to facilitate the analysis of applications for novice developers or anyone who has an interest in information security.

Key words: web security 1; Vulnerability Analysis 2; Information Security 3.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

B2B	Business to Business
B2C	Bussiness to Client
C2C	Client to Client
DDOS	Distributed Denial of Service
HTML	Linguagem de Marcação e HiperTexto
LFI	Local File Inclusion
QUERY	Processo de Extração de informação em um Bando de Dados
RFI	Remote File Inclusion
SQL	Structured Query Language
URL	Uniform Resource Locator
XML	Linguagem Extensível de Marcação Genérica
XSS	Cross Site Scripting

Sumário

1. INTRODUÇÃO

Devido à expansão da Internet, há uma crescente demanda pela migração de aplicações em ambiente Desktop para ambiente Web. Os benefícios desta migração são claros, tais como, processos em tempo real, compras realizadas em poucos minutos e facilidade de acesso a aplicações Web, sites empresariais, serviços de relacionamento entre outras aplicações disponíveis para qualquer pessoa no mundo. Com isso, a Internet passou a ser um dos principais meios de comunicação e de negócios no mundo, tornando-se uma das opções mais flexíveis e rentáveis atualmente. Segundo pesquisa conduzida pelo Comitê Gestor de Internet no Brasil [cgi.br 2015], o número de pessoas conectadas a internet no ano de 2015 chegou a 148,2 milhões. Com o crescimento de acesso à rede Internet, aumenta-se também a quantidade de ataques a empresas que utilizam sites, tanto para a venda de seus produtos quanto para o provimento de serviços nessa plataforma.

Antigamente, as empresas de tecnologia e desenvolvedores de aplicações voltavam suas atenções apenas para a infraestrutura em que suas aplicações eram hospedadas, com o intuito de proteger os dados armazenados. No entanto, com o avanço da Web 2.0 e em conjunto com as crescentes aplicações Web, os focos dos atacantes voltaram-se em maior parte para falhas de segurança em aplicações. Essas passaram a ser o grande alvo dos ataques mais modernos. Conclusivamente, a segurança passou a ser um requisito relevante não somente às redes de computadores, mas também ao nível de aplicações [CROSS, 2006].

São 5 as vulnerabilidades mais encontradas atualmente nos sites Web, segundo a IMPERVA, ***L**ocal **F**ile **I**nclusion, **R**emote **F**ile **I**nclusion, Sql Injection, **C**ross **S**ite **S**cripting e **D**irectory **T**raversal*, sendo o *Cross Site Scripting* o mais recorrente, dado o avanço de aplicações no lado do cliente. A Figura 1 ilustra que, 58,5% dos sites pesquisados contém vulnerabilidade deste tipo.

Falhas como essas são frequentemente encontradas nos sites Web e não devem ser negligenciadas pelos desenvolvedores, visto que essas vulnerabilidades podem oferecer ao invasor informações confidenciais dos usuários do sistema. Entender, explorar e prevení-las é de fundamental importância para se evitar possíveis contratempos no uso do sistema. Neste contexto, também é necessária a utilização de ferramentas que efetuem varreduras de possíveis pontos falhos no sistema antes que o mesmo seja entregue ao cliente para que não se entregue um

sistema vulnerável o que poderia trazer prejuízos incalculáveis a empresa ou organização.

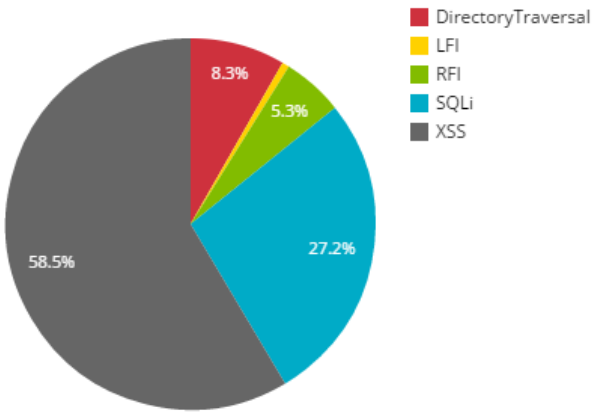

Figura 1: Ataques mais ocorrentes

Fonte: [IMPERVA 2015]

Considerando portanto que muitos desenvolvedores de aplicações Web não dão a devida importância no quesito segurança, esse trabalho visa contribuir para a redução das vulnerabilidades mais comumente encontradas. Além de permitir que o desenvolvedores possam se utilizar dessa ferramenta a qual é capaz de efetuar diversos *scanners* de vulnerabilidades com a finalidade de tornar a aplicação mais segura. Contribuindo assim, para que pequenos desenvolvedores e graduandos de cursos de computação façam uso da mesma como forma de estimular as boas práticas.

O uso desta ferramenta está voltado principalmente aos desenvolvedores iniciantes e equipes de empresas responsáveis por esse setor, porém não exime que qualquer usuário possa dela fazer seu uso.

O projeto e desenvolvimento da ferramenta, a partir de análise de outras soluções mais simples disponíveis sendo elas Nikto, Fimap, Uniscan e Xsser,

servem também como: a) base para melhorar e aperfeiçoar o conhecimento de vulnerabilidades Web. b) meio de divulgação e conscientização relacionado a segurança de aplicações e sistemas. c) forma de contribuir para o desenvolvimento de aplicações voltado para as questões de segurança de sistemas Web.

A ferramenta desenvolvida foi denominada de Arcádio, sendo essa, constituída de uma arquitetura simples para análise das vulnerabilidades.

1.1 Formulação e Escopo do Problema

A falta de um ambiente de teste para exploração de vulnerabilidade e detecção de falhas em sistemas pode ser um dos fatores que o levam a conter vulnerabilidades. Hoje os testes e análises das aplicações são feitos manualmente variando padrões de empresa para empresa. Existem práticas pré-estabelecidas, como por exemplo, o Owasp(*Open Web Application Security*), baseada na classificação de riscos e as ISO 27001 e ISO 17799 [Ranieri 2007], que abordam padrões de testes. Se especializar nas ISO's ou Padrões para o pequeno desenvolvedor, pode lhe ser de alto custo e muitas vezes os projetos realizados por esses desenvolvedores não são a altura de um padrão do tipo ISO, tornando assim uma especialização cara e desnecessária. O padrão OWASP não necessita de certificações, porém é necessário que se tenha tempo para a preparação das técnicas para explorar os sistemas.

1.2 Justificativa

Aplicações Web, por sua natureza, costumam ser publicamente acessíveis. O mesmo acontece com os servidores Web, que dão visibilidade a essas aplicações, em virtude da necessária disponibilidade das mesmas a usuários legítimos. Há uma maior chance de ataques a sistemas desse nível por possuírem uma abrangência maior de acesso em relação a aplicações direcionadas ao Desktop [Graves, 2010].

Testes de aplicações para as organizações é uma etapa cara, pois demanda tempo e ferramentas para que a análise do sistema seja completa [Ricardo A, 2012].

Testes visam encontrar vulnerabilidades ou falhas ao decorrer das transações de dados nos processos do sistema. Saber quais ferramentas utilizar e porque, é importante para que não se tenha desperdício de tempo na escolha dos scanners, pois os mesmo podem vir a ser incapazes de encontrar os problemas na aplicação

.

> "*Uma ferramenta por si só não pode resolver o que fundamentalmente é um problema de processo no desenvolvimento*" [Neil MacDonald]

Ainda segundo Neil [2014], o uso de apenas uma ferramenta para a análise de problemas e possíveis vulnerabilidades não será capaz de encontrar todos os problemas que o sistema possa vir ter, uma vez que ferramentas têm suas limitações e novas vulnerabilidades são encontradas todos os dias. Desta forma é preciso entender quais scanners são interessantes no processo de análise de uma aplicação, se elas realmente suprem as necessidades de um usuário e se atenderão as necessidades.

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo geral desse trabalho é o desenvolvimento de uma ferramenta a qual seja capaz de analisar sites Web e neles explorar algumas das vulnerabilidades mais conhecidas no ambiente Web. Além disso, tal ferramenta tem por objetivo utilizar-se dos melhores scanners, trabalhando em conjunto, permitindo assim auxiliar desenvolvedores Web a diagnosticar vulnerabilidades em seus sites melhorando o desenvolvimento de aplicativos com maior segurança.

1.3.2 Objetivos Específicos

- Identificação das principais vulnerabilidades no site da IMPERVA que possam impactar no funcionamento de um ambiente Web;

- Levantamento das principais scanners dentro do ambiente Livre(Kali Linux) utilizadas para varredura de vulnerabilidades em ambientes Web;
- Exploração das vulnerabilidades por meio dos scanners escolhidos;
- Criação da ferramenta de análise das vulnerabilidades;
- Teste das ferramentas selecionadas com a ferramenta em que o usuário indica o site que deseja ser analisado.
- Apresentação de um relatório final contendo, quando possível as vulnerabilidades encontradas no site, indicando ainda o que o desenvolvedor deverá fazer com a finalidade de corrigir o código com a falha.

1.4 Organização do Trabalho

Este trabalho segue organizado da seguinte forma: No capítulo 2 é descrito a metodologia utilizada no desenvolvimento deste trabalho; No capítulo 3 é descrito sobre a fundamentação teórica em que são abordados os principais conceitos para um bom entendimento deste estudo; No capítulo 4 descrevem-se todo os estudos feitos com os scanners e ferramentas utilizadas para a criação do Arcadio, assim como a descrição de cada uma; No capítulo 5 descrevem-se o ambiente de desenvolvimento da ferramenta, quais foram os critérios utilizados para a escolha dos scanner e como foi criado o ambiente de teste; E por fim, no capítulo 6 tem-se a conclusão do trabalho com, o que pode ser alcançado com o trabalho e quais são as recomendações para trabalhos futuros.

2 Trabalhos Relacionados

[Pelizzi and Sekar 2012] Enfatizam o uso de ferramentas de detecção do XSS no lado do cliente, como extensões utilizadas nos navegadores Web. Destacam ainda os pontos positivos e negativos dessa medida contra XSS, avaliando assim tipos de filtros e a usabilidade das extensões como ferramentas de proteção.

[Rocha 2012] Usa uma ferramenta livre extensível para fazer a varredura em aplicações web, com o intuito de cobrir vulnerabilidades que outras ferramentas do mesmo seguimento que ele criou não cobriam, como inclusão de arquivos remotos.

[Rosa 2012] Usa algumas metodologias para a detecção de vulnerabilidades entre essas metodologias estão Owasp que detalha ao usuário um passo a passo de análise, a utlização de uma metodologia se mostrou eficaz no trabalho o que poderá vir a ser usado como base neste onde seria possível saber se era necessário ou não uma metodologia especifica na ferramenta.

[Hinrichs et al. 2013] apresenta uma linguagem declarativa para desenvolvimento de aplicações Web elaborada para eliminar automaticamente várias vulnerabilidades comuns. Entre as vulnerabilidades evitadas com o uso da linguagem estão a Injeção de SQL o XSS e a Falta de Controle em nível de Acesso.

Hinrichs ainda destaca os pontos negativos da linguagem em não prevenir contra outras vulnerabilidades comuns, como a Falsificação de Solicitação entre Sites e a Quebra de Gerenciamento de Sessão. Para a contribuição da comunidade cientifica este trabalho tem como fruto um framework diferente de todos os trabalhos estudados que apenas fazem o levantamento das ferramentas utilizadas, neste framework é possível trabalhar com o URL do site alvo e as vulnerabilidades que pretende explorar, a ferramenta devolvera em um relatório se o sistema tem a vulnerabilidade ou não.

3. Fundamentação Teórica

Neste capítulo serão abordados temas indispensáveis para o entendimento desse trabalho. Neles, são apresentados os tipos de vulnerabilidades e a segurança da informação em ambientes Web. Além disso, é descrito como cada *vulnerabilidade* trabalha.

3.1 Segurança da Informação

A informação, sem dúvida, é a parte mais importante de uma empresa. Se a empresa perde os dados ou se seus dados são roubados isso pode causar vários problemas ou então até a falência da empresa, como aconteceu com muitas empresas no atentado terrorista de 2001 em Nova York. A Segurança da Informação é a área responsável por proteger os dados da empresa contra ameaças ou até mesmo vazamento interno.

Atualmente, a informação é arma estratégica em qualquer empresa e também um recurso de vital importância nas organizações. A segurança da informação é um recurso que tem por finalidade proteger e também é uma forma de gestão. “A segurança da informação de uma empresa garante, em muitos casos, a continuidade de negócio, incrementa a estabilidade e permite que as pessoas e os bens estejam seguros de ameaças e perigos.” [BLUEPHOENIX, 2008].

Trabalhar em cima de ambientes perfeitos, programas atualizados ainda não é tudo pois mesmo que a meta seja 100% da segurança outras partes do processo influências na sua qualidade final. Todo o desenvolvimento de um sistema exige processos, pessoas fazem parte do processo e o maior erro em aplicações vem de falhas humanas [RM SILVA 2008].

3.2 Ambientes Web

Com o uso cada vez mais freqüente 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 ***B**usiness **t**o **B**usiness (B2B)* e ***B**usiness **t**o*

C**lient (B2C)* ou ***C**lient **t**o **C**lient (C2C),* utilizam sistemas de gerenciamento de bancos de dados que são parcialmente ou totalmente integrados à ***WWW.

Estar preparado para todo tipo de vulnerabilidade é extremamente impossível pois são cerca de 56 vulnerabilidades descobertas todos os dias, segundo apontamento feito pela empresa *Symnate [Symnate 2016*]. Tais vulnerabilidades podem segundo a empresa citada, abrir algumas brechas no site, e com isso o invasor tem oportunidades para invadir o sistema.

Algumas aplicações fazem consultas a banco de dados baseadas nas informações recebidas dos clientes. Assim, se os dados recebidos não forem validados corretamente, o invasor poderá pegar dados de usuários do sistema através de simples comandos de SQL.

3.3 Vulnerabilidades Web

Embora haja muitos tipos de vulnerabilidades Web, nesse trabalho serão apresentadas apenas algumas delas sendo RFI, LFI, SQL Injection, Directory Traversal e Xss. A escolha dessas deu-se com base na pesquisa feita por uma empresa conceituada e voltada para a segurança de informações e organizações de uma maneira geral.

3.3.1 RFI – Remote File Inclusion

Remote File Inclusion permitem ao atacante incluir códigos e dados maliciosos, os quais resultam quase sempre em ataques devastadores. Segundo [Owasp 2007], os ataques de execução de arquivos maliciosos podem afetar aplicações que utilizam PHP, XML e ainda todos os frameworks que aceitem nomes de arquivo ou arquivos dos usuários.

Este tipo de vulnerabilidade acontece quando não há uma validação dos dados (exemplo: verificar se o arquivo que está sendo incluído realmente está no diretório especificado pela aplicação Web), esses dados são passados como parâmetro e utilizados em algumas das funções dos scripts PHP, como include(),

include_once(); Geralmente essas funções têm como objetivo a inclusão de dados ou arquivos locais na aplicação.

Para exemplificar, é ilustrado uma passagem de dados via parâmetro. Nela, há uma inserção da página "**buscaVulnerabilidade.php**" é feita junto a uma determinada URL.

http://www.arcadio.com/index.php?inc=buscaVulnerabilidade.php

Considerando que o parâmetro utilizado, inc do arquivo index.php recebe a string buscaVulnerabilidade.php, o código fonte do arquivo index.php contém, portanto, uma inclusão usando o parâmetro inc.

Dessa forma, um atacante pode passar qualquer tipo de arquivo ou falha para a inserção no sistema via URL.

O código da vulnerabilidade apresentada na Figura 2 ilustra uma vulnerabilidade do tipo RFI.

Para explorar esse tipo de vulnerabilidade é necessário a manipulação dos dados que estão sendo passados por parâmetros à um arquivo php. Se no lugar de buscaVulnerabilidade.php for apontado uma URL, que contenha um código malicioso, isso poderia causar sérios danos aos usuários dessa web site.

```
1  <?php
2          $incfile = $_REQUEST["file"];
3          include($incfile.".php");
4  ?>
```

Figura 2 Vulnerabilidade RFI

Observando a *URL* abaixo apresentada, tem se o arquivo index.php efetuando uma inclusão do arquivo cmd.txt o qual está hospedado em *www.arcadio.com*. Nesse caso, é o próprio PHP quem cria a conexão com o site invasor para a inclusão do arquivo cmd.txt

http://www.arcadio.com/index.php?inc=http://invasor.com/cmd.txt?cmd=id

3.3.2 LFI – Local File Inclusion.

A vulnerabilidade LFI é uma vulnerabilidade capaz de permitir inclusões de arquivos locais. Este tipo de ataque pode expor arquivos e dados dos sistemas hospedeiros da aplicação Web [Alfredo, 2013],.

Esse tipo de vulnerabilidade assim como a RFI, também utiliza as funções como include(), include_once(), require() entre outras funções que chamam páginas e arquivos na aplicações, porém a inclusão é de arquivos locais e não de arquivos remotos. Isso naturalmente reduz a quantidade de possibilidade do invasor [Rocha Douglas 2012].

O código da ilustração 3, o código PHP recebe um parâmetro de nome inc via método GET, o qual tem seu valor atribuído a variável $inc. A linha 3 do código faz a inclusão do arquivo cuja o nome está atribuído à varável $inc.

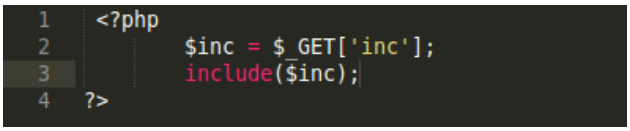

Figura 3. Vulnerabilidade LFI

No caso do LFI o invasor pode utilizar o código vulnerável para obter acesso a arquivos e dados do sistema local. Como exemplo, é possível passar como parâmetro a variável inc o caminho para um arquivo da aplicação. Como o /etc/passwd/ é um arquivo padrão nos sistemas operacionais Linux, o parâmetro (index.php?inc=/etc/passwd) permitirá checar quais os usuários cadastrados em um determinado servidor. Neste caso, com as permissões corretas e em um ambiente que não está isolado, o PHP irá incluir e irá também exibir o arquivo /etc/passwd [Abyssec, 2011].

Como é descrito na Figura 3, um invasor poderá explorar vulnerabilidades através de um simples navegador, digitando uma URL como:

http://Arcádio.uenp.edu.br/index.php?page=/etc/passwd.

Com isso, o conteúdo do arquivo /etc/passwd será incluído no arquivo index.php e será exibido no navegador. Uma vez obtido os nomes do usuários desse sistema, o invasor poderá em seguida obter a senha através da técnica de Força

Bruta. Força bruta é uma técnica onde o usuário através de uma lista de combinações faz repetidas chamadas a uma Query direta no banco com o objetivo de acertar muitas vezes o login ou a senha de um usuário.

3.3.3 XSS – Cross Site Scripting.

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 worms, etc. [Owasp 2007].

Este tipo de vulnerabilidade visa sites Web que fixam dinamicamente conteúdo sem a permissão dos administradores do sistema.

A Figura 4 descreve um simples método para guardar o *cookie* do usuário através de aplicações em PHP. Nessa, pode se observar que a linha 16 descreve um código básico que recupera via o método GET as informações que estão sendo passada via URL. A linha 8, 9 e 10 do *script* apenas seta na sessão os dados de *cookie* do usuário logado no sistema.

```
1  <?php
2  /*
3   * Um simples exemplo de cookies que geralmente
4   * são utilizados em sistemas web.
5   * PHPSESSID -> SID padrão da configuração do PHP
6   * login e senha -> cookies ficticios
7   */
8          setcookie('PHPSSESID', 'meu_sid');
9          setcookie('login', 'meu_login');
10         setcookie('senha', 'minha_senha');
11
12 /**
13  * Parte vulnerável do código: Imprime os dados passados
14  * pelo usuário sem nenhum tratamento
15  */
16         echo $_GET['var'];
17
18 ?>
```

Figura 4. Vulnerabilidade XSS

Esse *script* é utilizado no desenvolvimento para que o usuário não precise ficar fazendo a identificação no sistema para cada passo realizado. O problema está na linha 16, em que não se tem uma validação dos dados recebidos pelo método e

já está sendo impresso através do comando echo. O script abaixo descreve um modo de explorar a vulnerabilidade da linha 16.

```
<a href="http://confiavel.org.br/busca.cgi?CC=<SCRIPT SRC='http://Arcádio.uenp.edu.br/badguy.js
'></SCRIPT>"></a>
```

Nesse exemplo, o site não contem um filtro apropriado para proteger-se de inserção de *scripts*. Nesse caso, o site invasor irá passar através de parâmetros o *script* hospedado em Arcádio.uenp.edu.br, executando remotamente um arquivo.

3.3.4 SQL – Structured Query Language

Linguagem de Consulta Estruturada ou simplesmente SQL também podem sofrer determinados tipos de ataques. 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 com parte do comando ou consulta. A informação maliciosa fornecida pelo atacante engana o interpretador que irá executar comandos mal-intencionados ou manipular informações. [Owasp 2007].

Este método utiliza-se da vulnerabilidade do código implementado na aplicação para alterar a *string* de conexão ao banco de dados, comprometendo toda a troca de informações entre “Aplicação – Base de Dados” [Magaldi 2010].

A Figura 5 ilustra uma possível falha de SQL na qual os dados não validados pelo desenvolvedor estão recebendo qualquer tipo de consulta no banco de dados.

```
1  <?php
2              $usuario = $_POST['usuario'];
3              $senha = $_POST['senha'];
4     $query_string ="select * from usuarios where codigo ='($usuario)' AND senha ='($senha)'";
5  ?>
```

Figura 5. Vulnerabilidade Sql-Injection

Com a falha sendo detectada pelo invasor o mesmo pode usar tanto a URL da aplicação quanto os campos que fazem a requisição de *select* no banco de dados. Permitindo assim que o banco de dados retorne dados consultados de maneira simples. O código abaixo ilustra esse tipo requisição:

```
SELECT id, senha FROM usuarios WHERE id = '' or 2 = 2 and senha = '1234';
```

Com essa Query de consulta e com a falha exposta o invasor obteria informações do usuário como o seu id e a senha.

3.3.5 Directory Traversal

O Directory Traversal, também conhecido como passagem de diretório, tem por objetivo acessar arquivos e diretórios que estão armazenados fora da pasta raiz da Web. A vulnerabilidade acontece quando a permissão do diretório onde os arquivos estão armazenados oferece liberdade para que qualquer pessoa possa ler os arquivos, escrever ou até mesmo alterá-las. A imagem 6 abaixo ilustra o diretório vulnerável onde o atacante consegue visualizar as informações do diretório wwwroot.

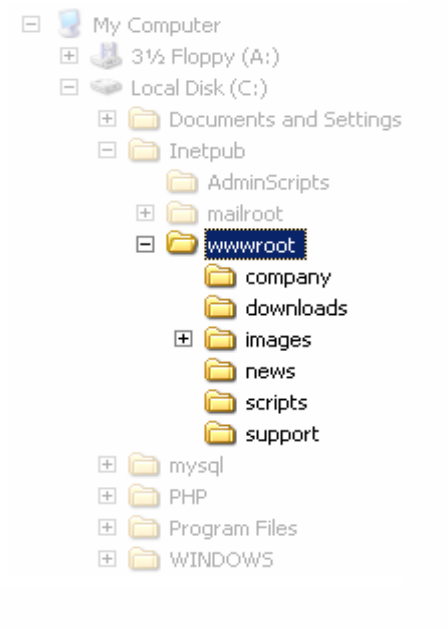

Figura 6. Raiz exposta para ataque a Directory Traversal

Fonte: [Acunetix]

Em geral, tais arquivos dão apenas permissão de acesso e não de escrita pelo próprio sistema operacional. Embora haja uma restrição nativa, alguns arquivos podem ser lidos.

Este ataque é também conhecido como *"dot-dot-slash",* "passagem de diretório", "escalada de diretório" e "retrocesso". Como pode ser observado na Figura 6 o diretório fica vulnerável ao atacante.

4. Materiais e Métodos

Este trabalho descreve uma pesquisa qualitativa, utilizando os métodos bibliográficos, documental e exploratório.

Em relação ao andamento da pesquisa, ela possui característica de caráter experimental, pois utiliza scanners sendo eles: Nikto, Uniscan, Fimap e Xsser para análises de vulnerabilidades consideradas mais comuns encontradas em sites Web.

A escolha desses scanners utilizados nesse trabalho teve como base tempo de análise, as atualizações e saídas dos relatório de cada scanner, isto é, os critérios adotados foram para *scanners* que não gerassem muito dados considerados pouco relevantes como por exemplo: Análise de outras vulnerabilidades não cobertas por este trabalho. Levou se também em consideração para a escolha, estudos relacionados ao tema.

A escolha de apenas alguns dos diversos tipos de vulnerabilidades foi feita de forma a permitir uma melhor especificação dos recursos de cada scanner e com base na pesquisa realizada pela IMPERVA. Obtendo assim, uma análise comparativa mais profunda entre a quantidade de falhas detectadas, no cenário simulado.

O trabalho foi gerado em ambiente livre e todas as ferramentas utilizadas foram escolhidas com base na disponibilidade da ofertada pelo pacote Kali Linux.

4.1 Ferramentas para Análise de Vulnerabilidades

Hoje existem no mercado dezenas de scanners para análise das vulnerabilidades em ambiente Web. Muitas delas são comerciais, com valores altos o que desfavorece pequenas empresas e desenvolvedores [Bruno Braga, 2013]. Neste trabalho um levantamento dos scanners foi elaborado para testes, todas os scanners são livres e estão disponíveis para ser usados em qualquer distribuição com base em Debian, para a instalação dos scanner é necessário a utilização do pacote de dados Katoolin que traz para o usuário as versões mais atualizadas dos scanners. Os scanners escolhidos para compor o quadro do Arcadio são descritos a seguir.

4.1.1 Nikto

O Nikto é um scanner livre, escrito em Perl usado para detectar vulnerabilidades em servidores *Web* [CIRT, 2015]. É simples de ser usado e atualizado, gera relatórios em txt, html e csv. Localiza padrões de vários arquivos inseguros, configurações e programas.

Esse scanner busca arquivos falhos nos diretórios da aplicação, validação de dados que estão sendo chamados e formulários sem verificação além de certificações. Por abordar vulnerabilidades de Directory Traversal o Nikto é parte das ferramentas utilizadas no Arcádio para a análise das vulnerabilidades.

4.1.2 Uniscan

Uniscan é um scanner livre, extensível e modular com funcionalidades práticas como detecção de inclusões locais e remotos de arquivos, Sql Injection. O Uniscan foi desenvolvido para analisar de forma estática aplicações Web. Gera relatórios no formato html, pdf e xml [Rocha Douglas 2012].

As vulnerabilidades cobertas pelo scanner incluem:

- RFI
- LFI
- SQL Injection
- Blind SQL
- Cross Site Scripting
- Stress Tests

O Blind Sql Injection é um método que assemelha-se ao SQL Injection, a diferença é que nesse caso a página possui uma certa segurança e o invasor não visualiza os dados da base de dados através da página web.

O teste de estresse ou Stress Tests é quando o scanner faz muita requisição na aplicação buscando a falha da aplicação até que a mesma exponha as vulnerabilidades do sistema.

4.1.3 Fimap

Fimap é uma ferramenta desenvolvida em python que pode auxiliar, auditar, explorar e encontrar erros de inclusão de arquivos locais e remotos. Usado apenas para bugs de RFI e LFI, é uma ferramenta sólida e aberta com grande poder de expansão. Atualmente o projeto está em andamento e disponível para a comunidade livre para sucessivas contribuições no github [Iman Karim, 2015].

O Scanner foi desenvolvido por Iman Karim, e tinha como principal problema a detecção de vulnerabilidade em sites desenvolvidos com a linguagem PHP, até que Xavier Garcia desenvolve um módulo a estes tipos de sites, o que tornou a ferramenta mais robusta para análises.

4.1.4 Xsser

É um framework que tem como objetivo procurar, explorar e reportar aplicações Web que estejam vulneráveis a ataques XSS. As explorações são feitas tanto locais quanto remotas [Cyberspace, 2009].

O “The mosquito” como é conhecido pela comunidade voltada à exploração de vulnerabilidade, facilita a detecção de diversas vulnerabilidades através de variadas formas de ataque como, por exemplo, ataque *Xss proxy* e *Xss shell [Porcullis Labs, 2008]*. Porém o foco desta ferramenta no presente trabalho é voltado a sites Web. O Scanner é interessante por ser incremental. Utilizado de maneira simples e sem muitas preocupações técnicas pode-se iniciar uma análise simples apenas através do comando abaixo:

```
xsser -u "http://host.com"
```

O comando acima inicia o Scanner de maneira básica, faz a injeção de scripts pré programados na aplicação Xsser. Com isso já se pode ter uma pequena análise em sites Web. Mas se for preciso detalhar o ataque, o Scanner conta com uma documentação bem robusta, atualizada e com ótimas informações de como melhorar os ataques a fim de se detectar falhas nas aplicações.

4.2 Métodos de Teste Software

Teste de software é o processo de execução de um produto para determinar se ele atingiu suas especificações e funcionou corretamente no ambiente para o qual foi projetado. O seu objetivo é revelar falhas em um produto, para que as causas dessas falhas sejam identificadas e possam ser corrigidas pela equipe de desenvolvimento antes da entrega final. Por conta dessa característica das atividades de teste, diz-se que sua natureza é “destrutiva”, e não “construtiva”, pois visa ao aumento da confiança de um produto através da exposição de seus problemas, porém antes de sua entrega ao usuário final.

4.2.1 Black-Box

Black-Box nada mais é que um método de teste em que não se pode ver o conteúdo de um sistema e sim a saída do mesmo. Nesse tipo de teste o desenvolvedor ou testador do sistema, está preocupado com as informações que aparecem na tela da aplicação, pois esse é o objetivo dessa etapa, fazer o sistema falhar a partir da interface.

Esse tipo de análise se comporta como um invasor do sistema tentando encontrar vulnerabilidades enquanto o sistema executa. Para encontrar o mesmo problema do tipo SQL Injection sem ter que varrer todo o código, uma análise dinâmica tem uma abordagem menos teórica: o analista de segurança executa a aplicação e procura por campos em formulários ou outro tipo de entrada de dados, manipula os dados que são passados para a aplicação e de acordo com o resultado verifica na prática a existência ou não de uma vulnerabilidade. A análise dinâmica tem a vantagem de explorar outras vulnerabilidades na aplicação que não podem ser detectadas durando o desenvolvimento da aplicação [Felipe Freire, 2009].

4.2.2 White-Box

Também conhecidos como testes estruturais, testes caixa de vidro ou testes caixa clara, são testes que conhecem a estrutura de implementação do software e são geralmente aplicados a unidades pequenas de programas (como por exemplo

funções de validação de formulário ou então todos os métodos que chamam uma página interna ou externa). Geralmente é desempenhado pelo próprio programador durante a programação. Esse tipo de teste tem alguns benefícios como: o programador pode testar pequenas partes do programa (isso faz com que a depuração seja mais fácil); o programador conhece o comportamento esperado do sistema (dessa forma pode identificar melhor as falhas); enfim, conhecendo melhor o código, ele pode identificar falhas que seriam mais difíceis aos olhos de outros. [Dataprev 2014].

Diferente do Black Box, o White Box pode ser visto na analise mais pessoal das aplicações testadas, onde o usuário tem permissão para modificar o código fonte da aplicação, a partir do relatório das vulnerabilidades o usuário poderá saber qual parte do sistema possa ter a falha e através disso contê-la.

5. Desenvolvimento

O objetivo desse trabalho constituiu da criação de uma ferramenta o qual fosse capaz de auxiliar o usuário ou programador Web iniciante, na análise de segurança de seus sistemas e aplicações de forma simples e gratuita. A essa ferramenta deu-se o nome de Arcadio, o qual se encontra atualmente hospedado no github: https://github.com/crawer/arcadio.git e que qualquer pessoa pode entrar no diretório e fazer o Download a fim de se testar a ferramenta ou melhorá-la da maneira que necessitar.

O Arcadio não se prende apenas no módulo de consulta de vulnerabilidade, é oferecido ao usuário um módulo específico para controle das aplicações já testadas por usuário ou programador Web [Histórico]. Assim, através de uma conta previamente criada, o programador Web poderá saber estatisticamente qual a porcentagem de erros obtidos em uma ou em várias de suas aplicações.

São ainda apresentados ao usuário, códigos de exemplo, possibilitando obter maior compreensão sobre a segurança, permitindo reduzir o número de erros no desenvolvimento de suas aplicações. Essa ferramenta apresenta, portanto um diferencial em relação a muitos scanners disponíveis. Pois além de ser gratuitas na licença MIT, permite que seja apresentado ao usuário por meio de relatório como tais falhas acontecem nas aplicações e sua forma de tratamento com exemplos.

O usuário do Arcadio não precisa ter conhecimento avançado de análise de vulnerabilidades. A partir do momento que o usuário recebe o relatório da aplicação testada o mesmo pode ir até a aba “Como tratar”, e entender melhor cada vulnerabilidade e como tratá-la.

O acesso ao sistema é realizado a partir da página inicial que faz a validação do login e senha de cada usuário. Todos os usuários, exceto o administrador, têm a mesma regra de acesso. Isto é, cada usuário pode acessar apenas o que for permitido à própria conta.

A tela inicial do sistema é aberta com o histórico de pesquisas feitas pelo usuário além de alguns dados por regiões, permitindo um acompanhamento sobre quais as vulnerabilidades encontradas no país. A Figura 7 ilustra o front-end d ferramenta.

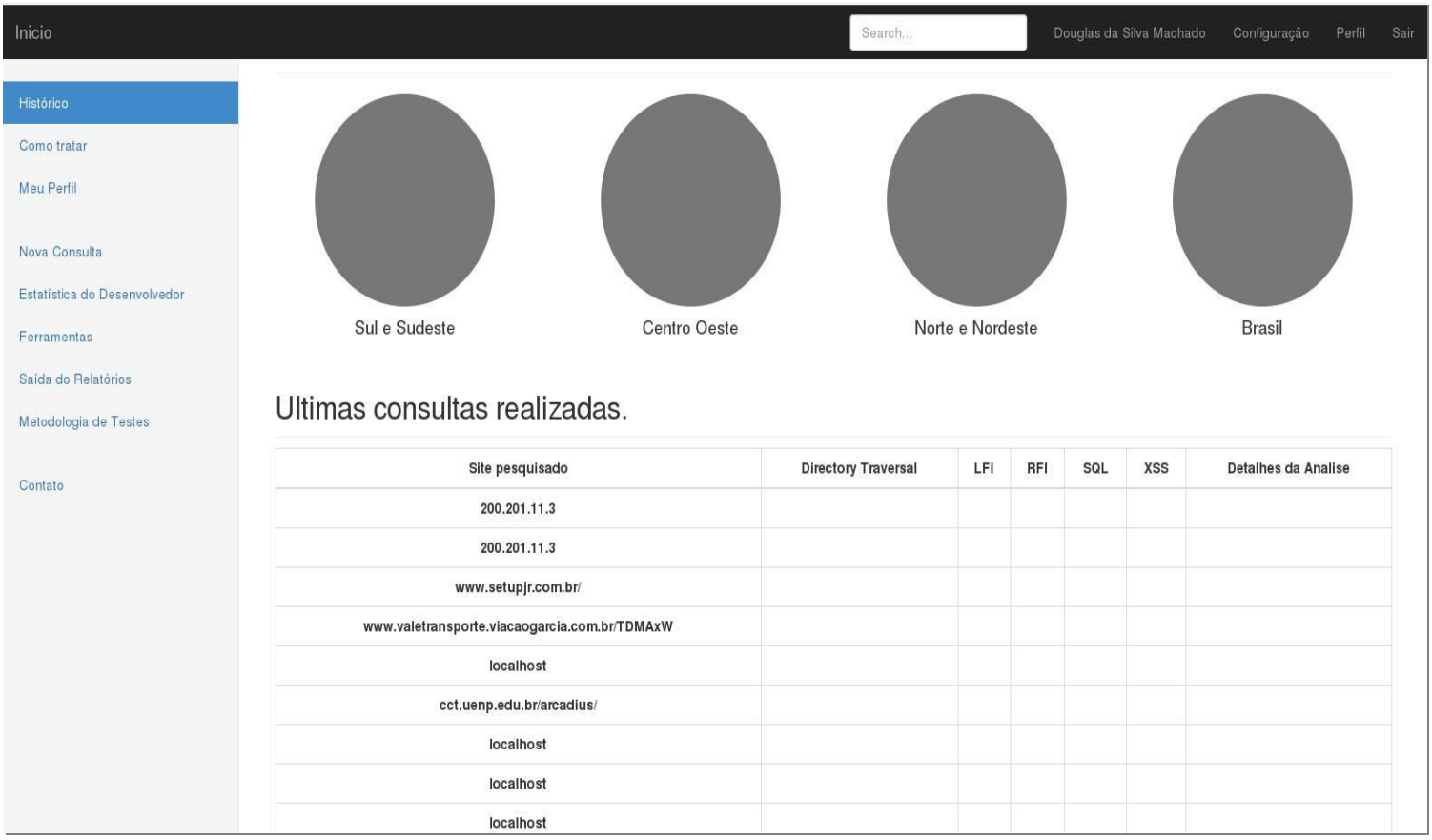

Figura 7. Arcadio

O diagrama de Casos de Uso UML, visto na Figura 8 abaixo, exibe todas as funcionalidades que o usuário terá quando optar por utilizar o sistema nas tarefas de desenvolvimento.

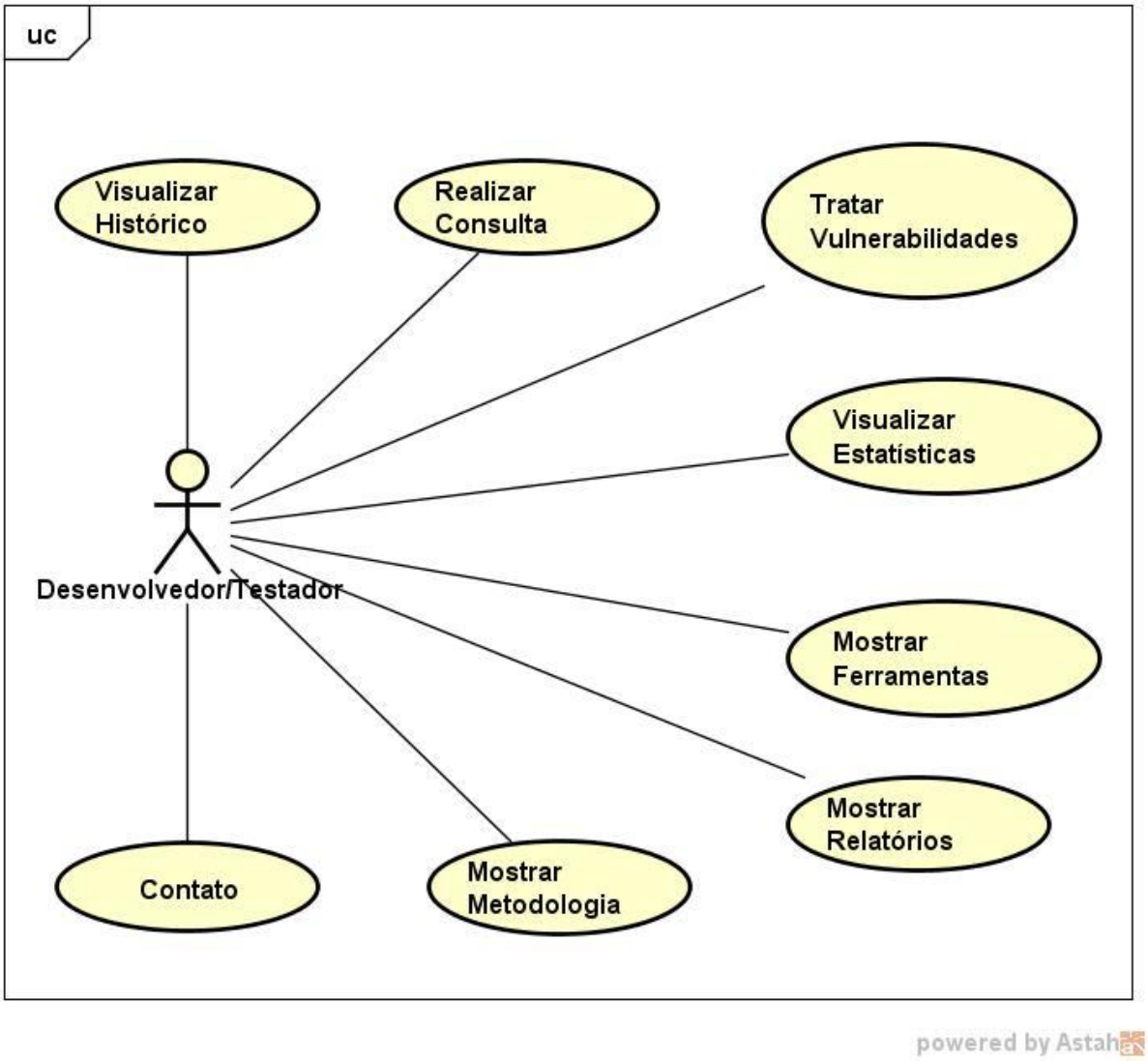

Figura 8. Funcionalidades do Framework

- **Visualizar Histórico:** Mostra ao usuário o histórico de sites pesquisados, quais vulnerabilidades foram encontradas naquela pesquisa além de um gráfico geral das vulnerabilidades pesquisadas até o momento.

- **Tratar Vulnerabilidade:** Essa página mostra ao usuário as informações de códigos falhos e como pode ser feito o tratamento das 5 vulnerabilidades, com isso o desenvolvedor poderá se basear no que o Arcádio exemplifica e o código desenvolvido na aplicação. Se a pesquisa retornou alguma vulnerabilidade é só analisar o meio de tratamento.

- **Realizar Consulta:** Aqui é a página principal do Arcadio, o usuário tem como opção a inserção do link do site em que a aplicação está hospedada e após a análise do “Arcadio” é gerado uma saída como resultado ao usuário, com essa saída se o mesmo não tiver o mínimo de entendimento de ferramenta ele tem a opção de entender as saídas na aba “Saída dos Relatórios”, em que é explicado cada tela, e o que aconteceu na análise.

- **Visualizar Estatística**: Com a estatística o usuário poderá saber qual a porcentagem de sites que ele vem analisando que contem um tipo de vulnerabilidade. Com isso ele tem um relatório onde poderá entender quais partes do desenvolvimento ele precisa melhor, visto que cada vulnerabilidade aborda um tipo diferente de desenvolvimento.

- **Mostrar Ferramentas**: Que tipo de ferramentas são utilizado nas análises e porque a escolha delas aqui na ferramenta. Saber quais scanners estão sendo utilizadas no Arcadio é interessante pois muitas vezes o usuário já tem experiência de testar o sistema em alguns desses scanners, e poderá ter a opção de testar ou não o sistema.

- **Mostrar Relatórios**: Aqui o principal objetivo é “educar” o usuário com as saídas geradas na “Realizar consulta”. Muitos usuários não têm o mínimo de noção do que as saídas das análises querem dizer e como usar isso a seu favor para testar se realmente o site testado está vulnerável. Nesse módulo da ferramenta é oferecido ao usuário exemplo de saídas dos scanners e o que ele deve levar como para a análise mais especifica na correção do sistema analisado.

- **Mostrar Metodologia**: Todas as ferramentas têm suas próprias metodologias de testes e de força bruta para testar a aplicação, porém o Arcádio mostra ao usuário apenas o padrão utilizado pelo sistema em si, entrada de link, análise de aplicação, ferramentas testadas, vulnerabilidades encontradas e saída de relatórios.

- **Contato**: Utilizada pelo usuário caso tenha alguma dúvida do sistema ou do relatório de saída, além de sugestões de melhorias.

Essas são as funcionalidades que cercam o Arcadio,são objetivas ao usuario iniciante ou ao pequeno desenvolvedor, o mesmo tem a opção do teste, realiza se caso necessário, se não entender nada das ferramentas pode ir até a janela de relatórios e entender o que os scanners dizem. Todo e qualquer usuário que utilizar o sistema saberá como trabalhar com qualquer informação e fazer valer a análise de suas aplicações.

5.1 Ambiente de Desenvolvimento

Para o desenvolvimento do Arcádio foram necessários alguns requisitos, tais como: Uma linguagem para o desenvolvimento do sistema, eventualmente um Framework que foi utilizado com integração com a linguagem utilizada que é PHP, um editor de texto (Sublime text 3), utilizado para aumentar a produtividade do desenvolvimento, Scanners disponíveis no Kali Linux, disponibilizados através do Katoolin.

No desenvolvimento do Arcádio foram utilizados os seguintes recursos:

- Linguagens:
  - PHP5 + HTML5
- Framework:
  - Bootstrap 2.
- Servidor de aplicação:
  - Apache2
- Banco de Dados:
  - MySql 5.7.13
- Editor de texto:
  - Sublime Text 3.
- Sistema Operacional:
  - Xubuntu 14.04 64bts.

Todas as ferramentas utilizadas no desenvolvimento do Arcadio necessitaram de módulos e bibliotecas adicionais, sendo essas apresentadas a seguir:

- libmoose-perl

- perl
- Plack::Hanler::Staler
- Test::TCP
- ReadKey

Houve no decorrer do desenvolvimento da ferramenta uma dificuldade para que as ferramentas Uniscan, Nikto e Fimap pudesse trabalhar juntas em especial a ferramenta Uniscan que necessita dos dois últimos módulos para a execução correta do scanner.

Foi utilizado o MySql como banco de dados devido a sua performance e consistência, além da sua popularidade e facilidade de uso, visto que o conhecimento do autor em cima da ferramenta não dificultaria no desenvolvimento.

A escolha do Perl se deu em função de que 90% dos scanners utilizados nos estudos e utilizara os módulos Perl.

A utilização do CSS é feita a partir do Framework *BOOTSTRAP o* qual permitiu reduzir o tempo de criação de folhas CSS durante o desenvolvimento do sistema.

A ferramenta é composta em forma de camadas. A camada mais superficial, cliente ou front-end permite a entrada de dados e a interação com a camada do servidor a aplicação, a qual é mais baixa. Essa segunda camada, estabelece a ligação com as ferramentas instaladas no SO, além do Apache e do banco de dados MySql.

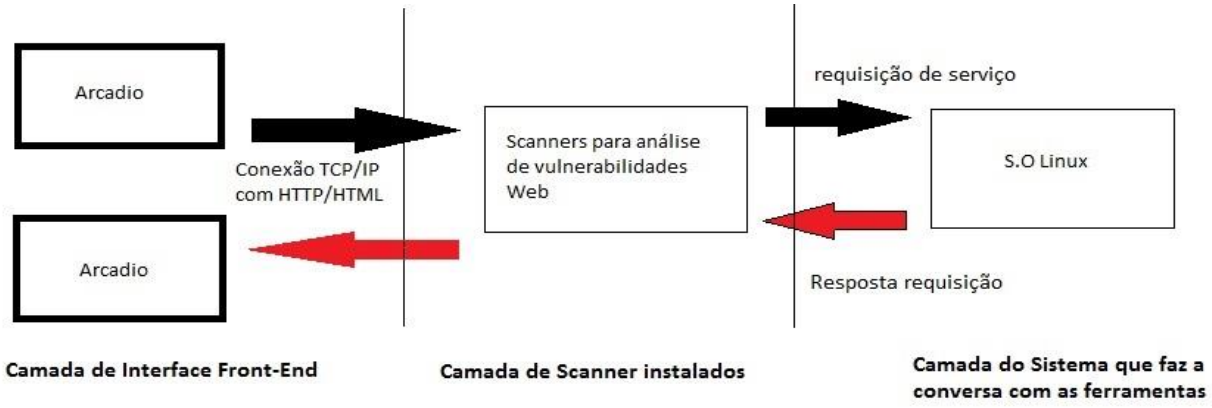

Figura 9 Camadas que a ferramenta utiliza

A Figura 10 ilustra uma das muitas abordagens utilizadas para fazer uma análise em sites Web.

Essas fases de teste acontece geralmente em qualquer tipo de analise, seja ela padronizada ou não, onde é necessário sempre a escolha do alvo, seguindo assim para a escolha das vulnerabilidades que se deseja testas, o uso da ferramenta é primordial passando a fase de correção do erro encontrado na aplicação. O Arcadio funciona basicamente desta maneira.

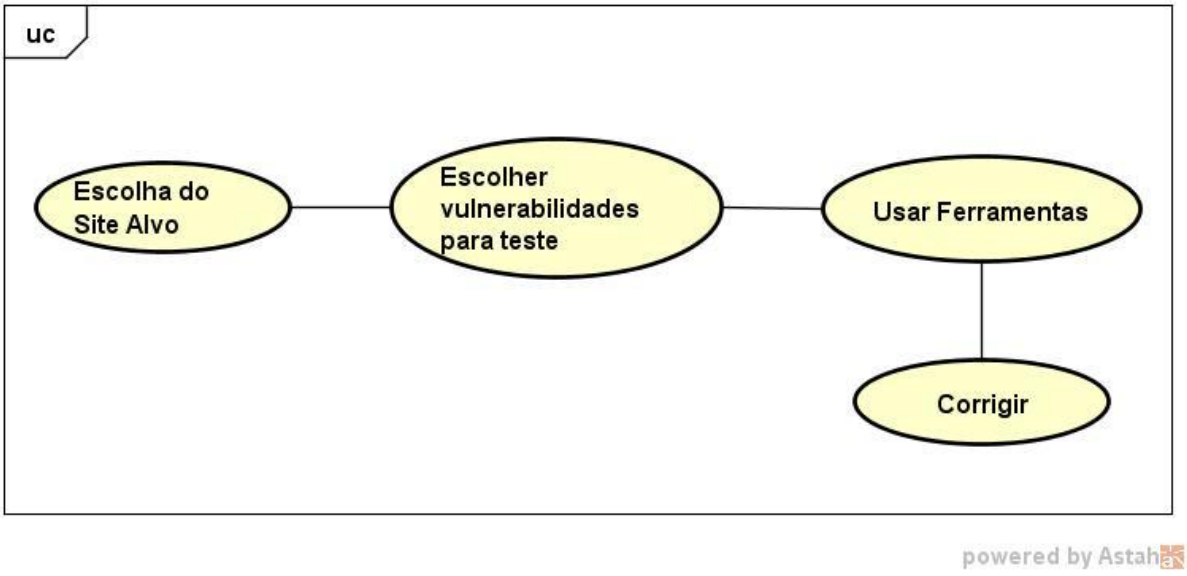

Figura 10. Fases Teste de Intrusão.

5.2 Ambiente de Teste Simulado

Com a finalidade de encontrar os melhores scanners para compor o quadro do Arcadio, foram criados códigos maliciosos desenvolvidos para testar e agrupar as melhoras ferramentas dentro do ambiente Linux e gratuitas, em um ambiente simulado.

Os testes foram feitos para que pudesse ser levantado uma quantidade de ferramentas capazes de cobrir as vulnerabilidades estudadas neste trabalho. Sendo assim foi criado uma aplicação contendo as 5 vulnerabilidades: RFI, LFI, Sql, Directory Traversal e Xss.

Os códigos maliciosos utilizados na criação do ambiente de testes são vistos na Figura 11. Nesta é apresentado o código do arquivo index.html, que é a página

inicial do ambiente de testes. Nessa página, há uma mensagem de apresentação e um formulário em que o usuário deverá entrar com o login e a senha para ter acesso ao sistema

```
1  <!DOCTYPE html>
2  <html>
3        <head>
4          <meta charset="UTF-08">
5        </head>
6        <body>
7              <form action="arquivo.php" method="post">
8                login: <inout type="text" name="login" id="login">
9                senha: <input type="password" name="senha" id="pass"><br>
10               <input type="submit" value="submit">
11               <p><a href="arquivo.php?arqTest=Test.html">Acessar Página</a></p>
12             </form>
13       </body>
14 </html>
```

Figura 11. Código Vulnerável

.

Observa-se na linha 12 que o código inclui o arquivo test.html após o usuário submeter o preenchimento do formulário. Caso a inclusão torne-se possível, uma mensagem informando a vulnerabilidade é apresentada. Conforme é visto na Figura 10.

```
<!DOCTYPE html>
<html>
      <head>
        <meta charset="UTF-08">
      </head>
      <body>
      <h1>Este site está vulneravel a inclusão de arquivo</h1>
      </body>
</html>
```

Figura 12. Conteúdo test.html

O index.html apresentado anteriormente na Figura 11, passa por parâmetro um arquivo, este arquivo é recebido na página PHP, recebe. php como ilustrado na Figura 13.

Com o código da Figura 13, o arquivo passado na execução da aplicação será incluído, pois não se tem uma validação se aquele arquivo realmente faz parte do sistema e com isso a vulnerabilidade será explorada.

```php
<?php
        $arquivo = $_post['arqTest'];
        include($arquivo);
?>
```

Figura 13. Conteúdo recebe.php

Com o código da Figura 14, pode-se observar que não há uma validação sobre os dados que estão sendo passados direto para a Query de consulta ao banco, com isso é possível testar as ferramentas para tentar obter o melhor resultado sob falhas de SQL Injection.

```php
<?php
        require_once 'conectaBanco';
        $usuario = $_POST['login'];
        $senha =$_POST['senha'];
        $query_string ="SELECT * FROM usuario_arcadio where usr ='".$usuario."' and senha='".$senha."';";
?>
```

Figura 14. Vulnerabilidade Sql

Tabela 1 – Resultado comparativo dos scanners testados no ambiente simulado

	LFI	**RFI**	**Directory Traversal**	**SQL**	**XSS**
Uniscan	x	x		x	
SqlNinja					
Vega				x	
Nikto			x		
PowerFuzze				x	x
Xsser					x
SqlMap					
fimap			x	x	
JSQL				x	

Com os resultados obtidos dos testes simulados, foi possível apresentar a Tabela 1.

Os critérios utilizados para a escolha dessa ferramenta, foram:

- **Flexibilidade**;

  Todas as ferramentas podem ter módulos incrementados, podendo assim melhorar a parte especifica de testes nas aplicações como por exemplo: Aumentar módulos de testes em estresse.

- ***Tempo* de resposta**;

  O tempo de resposta das ferramentas é levado em conta visto que ferramentas como a Powerfuzze demoram por fazer a analise de todas as vulnerabilidades que ela cobre, como o foco do trabalho é apenas 5 vulnerabilidades, a quantidade de informações geradas seriam um desperdício de tempo para a ferramenta.

- **Relatórios dos scanners: *Terminal*, Arquivos, XML, Interface, PDF**;

  O tipo de relatório que cada ferramenta gera também é foi um critério muito importante pois relatórios gerados através de interface não seriam utilizados na ferramenta desenvolvida visto que não tem como fazer a comunicação da ferramenta Arcadio com o Scanner de análise. Então foi dado prioridade para ferramentas que tivessem seus relatórios no terminal.

- **Ferramentas *atualizadas*.**

Com base nesses critérios de seleção dos scanners é possível se ter um quadro para analise dentro da ferramenta Arcadio, esse quadro é baseado apenas em ferramentas livres e atualizadas e que podem ser incrementadas a partir da necessidade pois assim é possível manter a ferramenta atualizada e posteriormente incrementar novas funcionalidades a mesma.

5.3 Arquitetura de Testes

O diagrama (workflow), Figura 15 ilustra o funcionamento da arquitetura de testes do Arcádio. Após iniciado a análise, passa ao falso/positivos para cada vulnerabilidade, e em seguida, é gerado um relatório sobre a análise realizada.

A ferramenta testa vulnerabilidade por vulnerabilidade, verificando o seu falso positivo e passa para a próxima, no fim gera o relatório ao usuário que saberá o que deve ser feito a partir da análise das ferramentas.

As vulnerabilidades descritas a baixo são analisadas pelas ferramentas escolhidas onde o scanner Uniscan testa vulnerabilidades do tipo RFI e LFI além de SQL Injection, o scanner Nikto e Fimap testa vulnerabilidades do tipo Directory Traversal e por fim a vulnerabilidade XSS é testada pelo scanner Xsser.

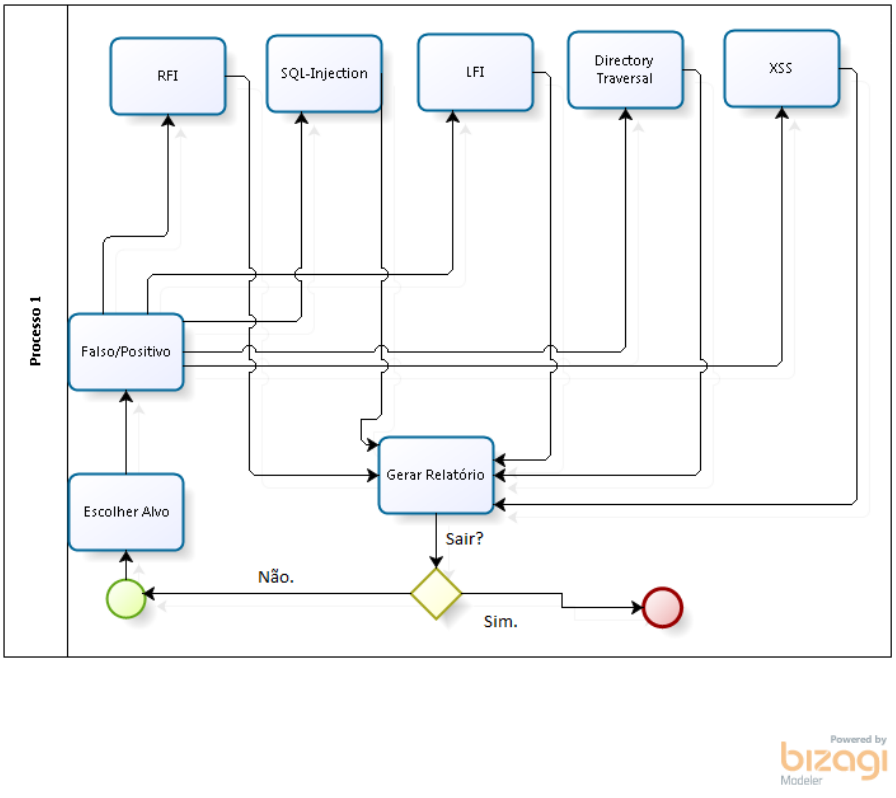

Figura 15. Arquitetura de teste de intrusão

O modulo de pós-teste do Arcadio, consiste em um relatório contendo informações de análise feitas pelo usuário.

Alguns testes foram realizados na ferramenta ja finalizada com os scanner escolhidos sendo eles Nikto, Fimap, Uniscam e Xsser, com o intuito de testar seu funcionamento com todas estes scanners em conjuntos. O resultado desses testes pode ser visto na Tabela 2 as quais foram encontradas em ambientes reais.

Tabela 2 – Testes realizados em ambientes reais da Web.

Site	RFI	LFI	SQL	DirectoryTraversal	Xss
Rádio Web Local				X	X
Mídia Digital-CJ				X	
Mídia Digital-TN					X
Arcádio		X			

A Tabela 2 ilustra os testes realizados com a ferramenta finalizado. As três mídias são de natureza jornalística, escolhidas pois contém imagens, audios, vídeos e muitos desses sites são desenvolvidos a algum tempo, sendo assim vulnerabilidades são recorrentes.

Como pode ser visto na tabela, os quatro sites foram detectados algum tipo de vulnerabilidade sendo dois sites com Xss e dois com Directory Traversal, sendo esse ultimo ter sido possível a visualização de um arquivo com informações dadas pelo administrador sobre o sistema contendo as permissões que arquivos continham e quais foram as modificações feitas na aplicação.

6 Conclusões

É possível concluir que o tema segurança da informação é algo abrangente sendo algo que envolve diversas etapas além de ferramentas como as estudadas neste trabalho. Necessitando assim de uma ferramenta mais acessível e usual de ser utilizado por um desenvolvedor iniciante, podendo entender como acontece a análise de um sistema e como as vulnerabilidades ocorrem na aplicação.

Neste trabalho foi possível o desenvolvimento da ferramenta Arcadio, em que foi utilizado uma abordagem que permitisse que qualquer usuário pudesse utilizar-se dele para fazer análises de vulnerabilidades em ambiente Web consideradas mais comuns.

Com o estudo realizado sobre scanners de vulnerabilidades, foi possível detectar que o trabalho é algo extremamente importante na detecção de vulnerabilidades, o que torna uma ferramenta a ser utilizado nos processos de desenvolvimento de aplicações Web. Uma das funcionalidades interessantes dos scanners utilizados na detecção das falhas é o poder de conseguir detectar páginas que são escondidas pelo navegador, esse é um dos módulos utilizados por diversos scanners estudados e que abre brecha para testes sob algumas vulnerabilidades.

A ferramenta desenvolvida neste trabalho é um ponto forte de todo o estudo realizado envolvendo o tema segurança da informação, a arquitetura é simples, usual, moderna e trabalha com scanners ativos e recentes na detecção de falhas. Poderá ajudar nas boas práticas de desenvolvimento de aplicações Web através das análises que poderão ser feitas no sistema. A ferramenta foi pensado para receber novas funcionalidades, com isso é possível acrescentar novas regras de análises, novos parâmetros e módulos específicos para qualquer vulnerabilidade Web.

Além disso, esta ferramenta está disponível no github: https://github.com/crawer/arcadio.git na forma de licença MIT para a Instituição com o intuito de auxiliar os alunos iniciante no desenvolvimento de suas aplicações.

A ferramenta necessita de uma validação eficaz em cima dos ambientes de testes realizados, tanto reais quanto simulado é necessário entender se a ferramenta realmente consegue detectar falhas com ambientes reais e fazer a ferramenta trabalhar em sistemas com grande quantidade de funcionalidades. Os testes realizados neste trabalho foram feitos apenas para saber se a ferramenta

conseguiria trabalhar e detectar alguns tipos de vulnerabilidades simuladas e com isso passar a analisar vulnerabilidades em alguns ambientes reais.

6.1 Trabalhos Futuros

O Arcadio deve continuar em constante evolução visto que novas vulnerabilidades sempre estarão a aparecer. Com isso, novos scanners necessitam ser criados e incorporados a esse framework.

A maior limitação do sistema é não ter a capacidade de analisar sites que usam uma quantidade grande de dados, como por exemplo um e-commerce, sendo assim seria interessante novas funcionalidades que abordassem esse tipo de nicho. Além de não garantir que as vulnerabilidades apontadas realmente são reais, o que a ferramenta faz é trabalhar em cima de scanners desenvolvidas por outras pessoas e empresas e através dos resultados obtidos pelos scanners é levantado os dados da análise.

Outra limitação do trabalho é a validação da ferramenta, pois a mesma não foi testada em diferentes tipos de sistemas e aplicações, e não chegou a ser disponibilizado para testes para outras pessoas onde seria possível fazer a avaliação de usabilidade. Sendo assim seria extremamente importante que tivesse uma validação da ferramenta em trabalhos futuros em ambientes reais.

Como sugestão, de trabalhos futuros pode-se apontar os seguintes itens:

a. Criação de módulos voltados a aplicações de e-commerce;
b. Aumentar do número de vulnerabilidades cobertas pela ferramenta;
c. Diagnóstico de falso/positivo;
d. Construir uma arquitetura em tempo real de escuta usando nodeJS.
e. Validação do sistema em ambientes diversos como: PHP com JAVASCRIPT, WORDPRESS, JOOMLA, JSF, RUBY ON RAILS entre outros aplicativos Web disponíveis.

7. Referências

ABYSSSEC. **PHP Fuzzing In Action**. Disponível em: <http://www.exploitdb.com/download_pdf/12943>. Acesso em: 25/11/2015

B, Braga. **Porque testar software é caro?** 2013

BLUE PHOENIX. **"Boas práticas de segurança**". Disponível em: <www.bluephoenix.pt.> Acessado em: 28/01/2016

CERT.br, NBSO. **Práticas de Segurança para Administradores de Redes Internet.** Disponível em: <http://www.cert.br/docs/seg-adm-redes/>. Acessado em: 05/01/2016. Versão 1.2. NIC BR Security Office, 2003.

CERT.BR and CGI.BR. **Cartilha de Segurança para Internet**. Glossário 2012.

CIRT. Nikto2 | CIRT.net. Disponível em: <http://cirt.net/nikto2/>. Acesso em: 14/01/2016.

COSTA, Aécio. **Tecnologias para apresentação de Dados**. Disponível em:<http://www.aeciocosta.com.br/wp-content/uploads/FG/Projeto%20de%20Sistemas%20na%20Internet%202014-1/5-PSI-%20Tecnologias%20para%20apr esse ntacao%2 0-%20JavaScript.pdf> Acesso em: 09/01/2016.

Cross, Michael. "***Developer's guide to web application security***". Syngress, 2011.

DATAPREV. **Visão conceitual de testes V0,3**. 2014 Disponível em: <http://desenvolvimento.dataprev.gov.br/ativos_processo/padrao_desenvolvimento_software/r_pddataprev_201404/arquivos/testes/ori_Visao_Conceitual_Testes.pdf> Acesso**:** 2/01/2016

FERNANDA, Adrielle. **Segurança da Informação.** Disponível em <http://www.ice.ed

u.br/TNX/encontrocomputacao/artigosinternos/aluno_adrielle_fernanda_seguranca_da_informacao.pdf> Acesso em 12/01/2016.

Felipe Freire. **Segurança: Por que fazer dois tipos de testes para encontrar os mesmos erros?** Disponível em: < https://www.ibm.com/developerworks/community/blogs/rationalbrasil/entry/seguran_c3_a7a_por_que_fazer_dois_tipos_de_testes_para_encontrar_os_mesmos6?lang=en > Acesso: 12/07/2016

GRAVES, K. CEH : **Certified Ethical Hacker Study Guide**. [S.l.]: Sybex, 2010. 439 p

Hinrichs, T. L., Rossetti, D., Petronella, G., Venkatakrishnan, V. N., Sistla, A. P., and Zuck, L. D. **Weblog: a declarative language for secure web development.** Paginas 59–70. 2013

I, Karim. **Fimap**. Disponível em: <https://github.com/kurobeats/fimap> Acesso em 5 de janeiro de 2016.

IMPERVA. **Imperva Web Application Attack** Report. Janeiro 2010. Disponível em:

L. Heitor, Q. Patricia, B. André, “**SQL Injection, entenda o que é, aprenda a evitá-lo**”. 2010.

OWASP Secure Coding Practices – “**Melhores Práticas de Codificação Segura OWASP Guia de Referência Rápida**”. 2010

OWASP XSSER. **What is Xsser?** Disponível em: <https://www.owasp.org/index.php/OWASP_XSSER> Acesso: 22/05/2016.

Pelizzi, Riccardo, and R. Sekar. "**Protection, usability and improvements in reflected XSS filters**."*ASIACCS*. 2012.

Portcullis Labs. **Xss Shel** Disponível em < https://labs.portcullis.co.uk/tools/xss-shell/> Acesso: 12/07/2016

Ramos, A Ricardo. **“Introdução, Verificação, Validação e Teste de Software**” 2012.

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 03/12/2015.

Rosa; **Comparação e especialização de metodologias de segurança em aplicações web para o contexto de sistemas de icp.** Disponível em:<https://projetos. nf.ufsc.br/arquivos_projetos/projeto_1256/tcc_lucas_rosa_final.pdf> Acesso em 19/02/2016.

Souza, Lucas Lima. "**Desenvolvimento seguro de aplicações web seguindo a metodologia owasp**." (2012).

Souza, Ranieri Marinho de. **"Implantação de ferramentas e técnicas de segurança da informação em conformidade com as normas ISO 27001 e ISO 17799**." 2008.

Thompson, Michael. "Bluephoenix: **Application modernization technology** audit." 2004.

Wagner Aparecido Monteverde, Rodrigo Campiolo. **Estudo e Análise de Vulnerabilidades Web.** Disponível em <http://www.lbd.dcc.ufmg.br/colecoes/sbseg/2014/0065.pdf> Acesso em 18/02/2016.

WEBSECURIFY. Websecurify | **Web Application Security Scanner and Manual Penetration Testing Tool.** Disponível em: < http://www.websecurify.com>. Acesso em: 10/01/2016.