

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

José Victor Pereira Costa

**Análise de Vulnerabilidades de Segurança em
Portais de Governos Eletrônicos**

Uberlândia, Brasil

2017

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

José Victor Pereira Costa

**Análise de Vulnerabilidades de Segurança em Portais de
Governos Eletrônicos**

Trabalho de conclusão de curso apresentado
à Faculdade de Computação da Universidade
Federal de Uberlândia, Minas Gerais, como
requisito exigido parcial à obtenção do grau
de Bacharel em Ciência da Computação.

Orientador: Rodrigo Sanches Miani

Universidade Federal de Uberlândia – UFU

Faculdade de Ciência da Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2017

José Victor Pereira Costa

Análise de Vulnerabilidades de Segurança em Portais de Governos Eletrônicos

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Ciência da Computação.

Trabalho aprovado. Uberlândia, Brasil, 12 de dezembro de 2017:

Rodrigo Sanches Miani
Orientador

Professor

Professor

Uberlândia, Brasil
2017

“O homem não teria alcançado o possível se, repetidas vezes, não tivesse tentado o impossível. ” - Max Weber

Resumo

O crescimento do números de computadores, juntamente com a popularização da Internet, contribuiu para a digitalização de serviços e o surgimento de portais de governos eletrônicos. Esses portais, assim como toda tecnologia, possuem vulnerabilidades de segurança que colocam em risco os serviços oferecidos pelos mesmos, bem como os dados dos usuários que os utilizam. Este trabalho tem como objetivo identificar as eventuais vulnerabilidades de segurança em portais de governos eletrônicos, buscando compreender a relação entre a estrutura econômico-tecnológica e a segurança de sistemas. As análises a serem desenvolvidas levarão em conta o uso de ferramentas, *Web Scanners*, que irão auxiliar na obtenção de resultados mais efetivos. O desenvolvimento do projeto é feito observando o cenário tecnológico nacional bem como suas estruturas de segurança. Os resultados obtidos demonstram que as vulnerabilidades mais encontradas são: Injeção, XSS e Redirecionamento e Encaminhamento Inválidos. Os estados com maior nível econômico apresentam pior desempenho na análise, destacando-se o estado do Rio de Janeiro.

Palavras-chave: Segurança da informação, Vulnerabilidades, Governo eletrônico

Lista de ilustrações

Figura 1 – Total de incidentes Reportados ao CERT.BR por Ano. Extraído do portal (CERT.BR, 2016).	10
Figura 2 – Incidentes reportados ao CERT.br do período de Janeiro a Dezembro de 2016 . Extraído do portal CERT.br (CERT.BR, 2016).	11
Figura 3 – Taxas de adoção de medidas de segurança relatadas pelos entrevistados. Extraído do relatório <i>Sob fogo cruzado Infraestrutura crítica na era da guerra cibernética</i> (BAKER; WATERMAN; IVANOV, 2013).	11
Figura 4 – Imagem ilustrando a exploração de um vulnerabilidade. Extraído de (Gabriella Fonseca Ribeiro, 2011).	16
Figura 5 – Exemplo de exploração de um vulnerabilidade XSS. Extraído do portal (OWASP, 2013)	17
Figura 6 – Componentes do <i>Scanner</i> . Adaptado de (HKSAR, 2008).	18
Figura 7 – Imagem da ferramenta Skipfish. Imagem produzida pelo autor.	32
Figura 8 – Imagem da ferramenta Uniscan. Imagem produzida pelo autor.	32
Figura 9 – Proporção de cada tipo de vulnerabilidade detectada nos resultados. Produzido pelo autor	34
Figura 10 – Gráfico representando a quantidade de <i>sites</i> com determinado tipo de vulnerabilidade. Produzido pelo autor	36
Figura 11 – Tabela de classificação do PIB. Extraído do artigo de (WIKIPÉDIA, 2017).	46

Lista de tabelas

Tabela 1 – Avaliação das ferramentas	27
Tabela 2 – Tabela de Resultados por Portal	29
Tabela 3 – Vulnerabilidades de acordo com classificação de (OWASP et al., 2013)	30
Tabela 4 – Tabela de Resultados por Portal	31
Tabela 5 – Tabela de Resultados por Portal,NMAP	33
Tabela 6 – <i>Sites</i> ordenados pelo número de vulnerabilidades do tipo A10.	35
Tabela 7 – Sites ordenados pelo número de vulnerabilidades do tipo A3.	35
Tabela 8 – <i>Sites</i> ordenados pelo número de vulnerabilidades do tipo A1.	36
Tabela 9 – Classificação dos portais levando em conta o trabalho de (OWASP et al., 2013).	37
Tabela 10 – Classificação levando em conta análise dos resultados.	39

Lista de abreviaturas e siglas

ABNT	Associação Brasileira de Normas Técnicas
CERT.BR	Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil
CERT	Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança
CMS	<i>Content Management System</i>
CSRF	<i>Cross-Site Request Forgery</i>
CVE	<i>Common Vulnerabilities and Exposures</i>
FTP	<i>File Transport Protocol</i>
HTTP	<i>Hypertext Transfer Protocol</i>
IBGE	Instituto Brasileiro de Geografia e Estatística
IEC	<i>International Electrotechnical Commission</i>
IP	<i>Internet Protocol</i>
ISO	<i>International Organization for Standardization</i>
LFI	<i>Local File Include</i>
NBR	Norma Brasileira
OWASP	<i>Open Web Application Security Project</i>
PIB	Produto Interno Bruto
RCE	<i>Remote Command Execute</i>
RFI	<i>Remote File Include</i>
SMAR	<i>Security Measure Adoption Rate</i>
SOAP	<i>Simple Object Access Protocol</i>
SQL-i	<i>Structured Query Language Injection</i>
SQL	<i>Structured Query Language</i>
SSH	<i>Secure Shell</i>

TCP	<i>Transmission Control Protocol</i>
UDP	<i>User Datagram Protocol</i>
URL	<i>Uniform Resource Locator</i>
XSS	<i>Cross-Site Scripting</i>

Sumário

1	INTRODUÇÃO	10
2	FUNDAMENTAÇÃO TEÓRICA	14
2.1	Segurança da informação	14
2.2	Vulnerabilidades	14
2.3	Scanners	17
2.4	Governo Eletrônico	18
2.5	Trabalhos Correlatos	19
3	DESENVOLVIMENTO	26
3.1	Metodologia	26
3.2	Resultados	28
3.3	Análise dos resultados	34
4	CONCLUSÃO	40
	REFERÊNCIAS	41
	APÊNDICES	43
	APÊNDICE A – PORTAS E SUAS RESPECTIVAS FUNÇÕES . .	44
	ANEXOS	45
	ANEXO A – PRODUTO INTERNO BRUTO(PIB) DO ESTADOS BRASILEIROS	46

1 Introdução

A necessidade de sistemas mais seguros tem se tornado um desafio a instituições privadas e públicas em todo mundo. Enquanto a evolução tecnológica transforma as formas de comunicação e comércio vulnerabilidades têm surgido como limitantes desse desenvolvimento, como citado no trabalho de (NAKAMURA; GEUS, 2007).

O CERT (Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil), aponta que no ano de 2016 foram contabilizados um total de 647.112 incidentes de segurança como ilustrado pela Figura 1. Se comparado aos dois anos anteriores, 2015 e 2014, o valor apresentado demonstra uma queda, entretanto pode ser considerado ainda um alto valor quando comparado ao período anterior a 2013.

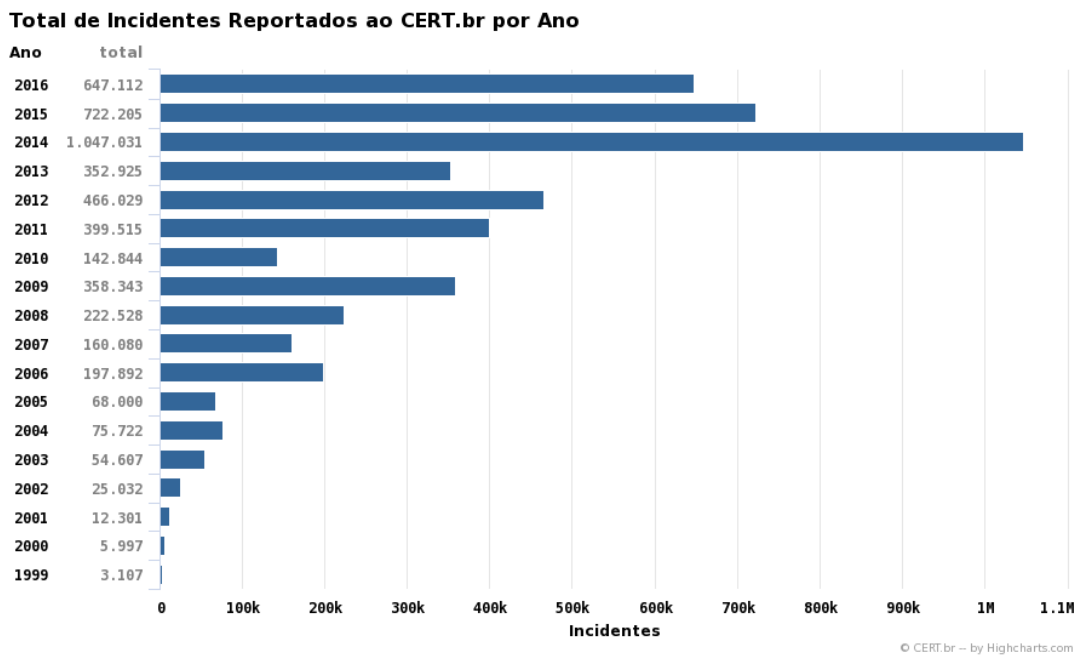


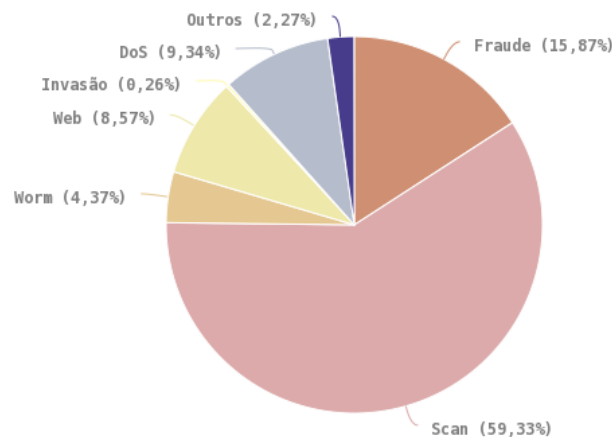
Figura 1 – Total de incidentes Reportados ao CERT.BR por Ano. Extraído do portal (CERT.BR, 2016).

A análise feita por CERT.br (2016) indica que do total de incidentes registrados 59,33% são *scans* feitos com uso de ferramentas automatizadas para detectar recursos utilizados na rede, como pode ser visto na Figura 2. Neste trabalho foram utilizadas ferramentas *scan* que serão detalhadas posteriormente.

Segundo o relatório de Baker, Waterman e Ivanov (2013), o Brasil está entre os países com a menor taxa de adoção a segurança por volta de 40% sendo medidas e soluções em segurança propostas por empresas e organizações que atuam na área, como fica ilustrado na Figura 3. As medidas consideradas no relatório de Baker, Waterman e

Ivanov (2013), são baseadas no *SMAR* (*Security Measure Adoption Rate*) as quais não são especificadas pelos autores. O relatório ainda cita um caso ocorrido em 2009, onde a mídia norte-americana afirmou que nos anos de 2005 e 2007 o Brasil sofreu cortes de energia que realizados por hackers como parte de um esquema de extorsão.

Incidentes Reportados ao CERT.br -- Janeiro a Dezembro de 2016
Tipos de ataque



© CERT.br -- by Highcharts.com

Figura 2 – Incidentes reportados ao CERT.br do período de Janeiro a Dezembro de 2016 . Extraído do portal CERT.br (CERT.BR, 2016).

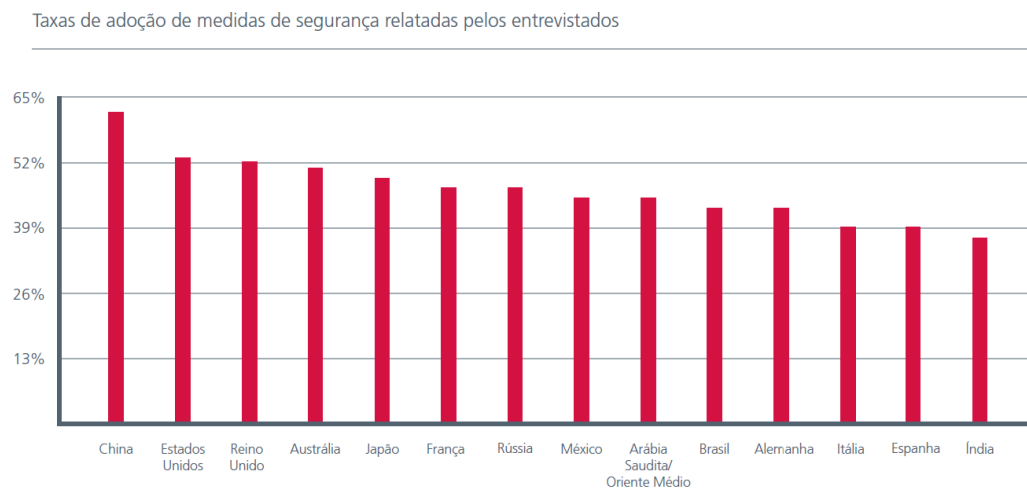


Figura 3 – Taxas de adoção de medidas de segurança relatadas pelos entrevistados. Extraído do relatório *Sob fogo cruzado Infraestrutura crítica na era da guerra cibernética* (BAKER; WATERMAN; IVANOV, 2013).

Alguns dos alvos de ataques na rede são os *sites* governamentais, também conhecidos com e-govs ou governos eletrônicos. Esses portais tem a finalidade de oferecer serviços e informações, relacionadas ao Estado, para o cidadão. Logo, são de suma importância

para o governo, pois caso alguma informação seja divulgada de forma errada os prejuízos poderiam ser incalculáveis ([SANTOS et al., 2014](#)).

Sobre a perspectiva de analisar as vulnerabilidades, o presente trabalho tem o objetivo de identificar as eventuais vulnerabilidades de segurança existentes em portais de governos eletrônicos, buscando compreender a relação entre a estrutura econômico-tecnológica e a segurança de sistemas. O trabalho se assemelha a abordagem utilizada por [Santos et al. \(2014\)](#), divergindo apenas na metodologia e na análise dos dados obtidos, onde essas etapas se baseiam nos trabalhos de ([FONSECA; VIEIRA; MADEIRA, 2007](#)).

O método utilizado na execução do projeto é centrado em quatro etapas essenciais:

1. Seleção das ferramentas: Escolha dentre os diversos softwares de detecção de vulnerabilidades, visando a escolha de ferramentas gratuitas;
2. Seleção dos portais web baseados no critério de avaliação: Critério de avaliação econômico onde são escolhidos 15 *sites* de prefeitura de cada estado da federação mais o *site* de cada estado;
3. Varredura do *sites*: varredura utilizando todas as ferramentas e adicionalmente o NMAP para a verificação de portas em aberto;
4. Análise dos resultados obtidos: análise feita buscando traçar um paralelo com a economia de cada estado e cidade.

Esse método de análise foi elaborado com base nas abordagens utilizadas por ([DOUPÉ; COVA; VIGNA, 2010](#)), ([ROCHA; KREUTZ; TURCHETTI, 2011](#)) e ([MONTE-VERDE; CAMPIOLO, 2014](#)). A seleção das ferramentas e o desenvolvimento da pesquisa tem seu apoio em uma bibliografia na qual experimentos semelhantes foram executados. Todas as ferramentas selecionadas são de código aberto, conforme será discutido na seção 3.1, ou possuem versões gratuitas. No total foram selecionadas duas ferramentas: Uniscan e Skipfish com base no trabalho de ([ROCHA; KREUTZ; TURCHETTI, 2011](#)) e ([HOLM et al., 2011](#)).

Espera-se fornecer uma projeção da segurança de *sites* eletrônicos em determinadas regiões dos países, oferecendo uma ampla análise da segurança em ambientes governamentais. O relatório elaborado ao final deste trabalho servirá como esboço da segurança dos sites abordados.

O presente estudo encontra-se organizado em capítulos, onde tem-se: o capítulo 2 evidencia a fundamentação teórica e os trabalhos correlatos, contribuindo, desta forma, para o entendimento e estruturação do trabalho; no capítulo 3 apresenta-se o desenvolvimento da pesquisa, sendo composto pela metodologia, resultados e análise dos mesmos;

por fim, no capítulo 4 desenvolve-se a conclusão, a qual reúne as considerações finais obtidas na efetivação deste trabalho.

2 Fundamentação teórica

Essa capítulo descreve a fundamentação teórica que abrange os principais conceitos necessários para compreensão deste trabalho.

2.1 Segurança da informação

Segurança da informação é definida pela [ABNT \(2005\)](#) através da norma ABNT NBR ISO/IEC 17799:2005 como sendo a preservação da confidencialidade, integridade e disponibilidade da informação.

Segundo [Stallings \(2015\)](#), as características a serem preservadas podem ser definidas como:

- *Confidencialidade*: manter restrições sobre a divulgação e acesso de informações, onde, se tem a finalidade de conservar a privacidade de informações e indivíduos. Se por exemplo um banco de dados não tiver restrição de acesso a usuários (acesso somente de pessoas cadastradas), então os dados lá contidos podem ser vazados, configurando assim uma quebra de confidencialidade.
- *Integridade*: manter informação buscando preservá-la contra modificações ou destruição imprópria incluindo a irretratabilidade e autenticidade. Suponha que um servidor seja invadido e aparentemente nenhum dos dados tenham sido alterados, entretanto existem pequenas mudanças. Caso não haja *backup* desses dados sua integridade foi comprometida.
- *Disponibilidade*: garantir o acesso rápido e confiável a informação. Um ataque pode ser realizado contra alguma aplicação web na finalidade de torná-la indisponível. Caso isso aconteça ocorrerá uma falha de disponibilidade.

2.2 Vulnerabilidades

Segundo [Aparecido e Bellezi \(2014\)](#), uma vulnerabilidade pode ser definida como um ponto falho em um sistema que permita a realização e a concretização de um ataque a um sistema computacional.

Dentre as vulnerabilidades existentes há as que afetam sistemas web. Essas são estudadas e listadas por [Owasp et al. \(2013\)](#), sendo organizadas em uma lista com as dez vulnerabilidades mais críticas (OWASP Top 10). A OWASP ou *The Open Web Application Security Project* é um fundação que tem como objetivo melhorar a segurança de *softwares*,

permitindo que indivíduos ou empresas tomem decisões informadas. A classificação da OWASP consiste nos seguintes grupos:

1. *Injection* (Injeção): ocorre quando dados e consultas não confiáveis são enviadas para o interpretador do banco de dados. Dessa forma o atacante pode iludir o interpretador fazendo com que ele execute comandos indesejados dando acesso a dados restritos.
2. Quebra de Autenticação e Gerenciamento de Sessão: quando funções da aplicação são implementadas incorretamente dando aos atacantes o poder de comprometer senhas, chaves e *tokens* de sessões ou permitindo que eles assumam a identidade de outros usuários.
3. *Cross-Site Scripting* (XSS): ocorre quando uma aplicação recebe dados sem uma validação ou filtro adequados permitindo assim que os atacantes executem *scripts* no navegador da vítima, realizando assim o sequestro de sessões a desfiguração de *sites* e redirecionamento para *sites* de caráter malicioso.
4. Referência Insegura e Direta a Objetos: ocorre quando o programador expõe uma referência à implementação interna de um objeto, onde o mesmo não possui verificação de controle de acesso. Isso permite que o atacante manipule essas referências para o acesso de dados não autorizados.
5. Configuração Incorreta de Segurança: ocorre quando a configuração padrão de servidores, *frameworks*, servidores e banco de dados é insegura. Assim, caso não sejam tomadas medidas de segurança, isso permite o acesso a atacantes.
6. Exposição de Dados Sensíveis: ocorre quando não é dada a proteção devida (criptografia, por exemplo) a dados sensíveis. Isso permite que os atacantes roubem ou realizem modificações, dando assim abertura a fraudes.
7. Falta de Função para Controle do Nível de Acesso: ocorre quando não se tem a verificação das requisições feitas pelo usuário. Logo um atacante pode modificar sua requisição permitindo que ele execute funções de nível de acesso mais alto.
8. *Cross-Site Request Forgery* (CSRF): ocorre quando a vítima que possui uma sessão ativa em um navegador é forçada a enviar requisições HTTP forjada incluindo qualquer informação vinculada a sessão(*cookie*). Isso permite ao atacante forçar o navegador da vítima a criar requisições que sejam aceitas como legítimas, como se essas requisições fossem realizadas pela própria vítima.
9. Utilização de Componentes Vulneráveis Conhecidos: ocorre quando se faz uso de componentes que possuem vulnerabilidades conhecidas. Em sua maioria esses componentes são bibliotecas, *frameworks* ou módulos que possuem privilégios elevados.

Em um ataque esses componentes podem ser explorados fazendo com que ocorram perdas de dados ou comprometimento do servidor.

10. Redirecionamentos e Encaminhamentos Inválidos: ocorre quando não há a validação das páginas para os quais o usuário é redirecionado. Isso faz com que os atacantes consigam redirecionar as vítimas para *sites* de *phishing* ou *malware*, ou até mesmo utilizem encaminhamentos para acessar a páginas as quais não se tem autorização.

As Figuras 4 e 5 ilustram respectivamente os ataques *SQL-i* e *XSS*. Na Figura 4 é mostrado a injeção de código SQL nos campos de *login* de uma aplicação, onde por exemplo, é injetado "1 = 1" que possui um valor booleano *true* que faz com que toda a tabela de usuários seja retornada.

Já na Figura 5, pode-se observar, a injeção de trechos de código, PHP, em uma caixa de texto, fazendo com que um alerta seja exibido para o usuário. Esse alerta poderia ser substituído por uma página ou *link* malicioso.

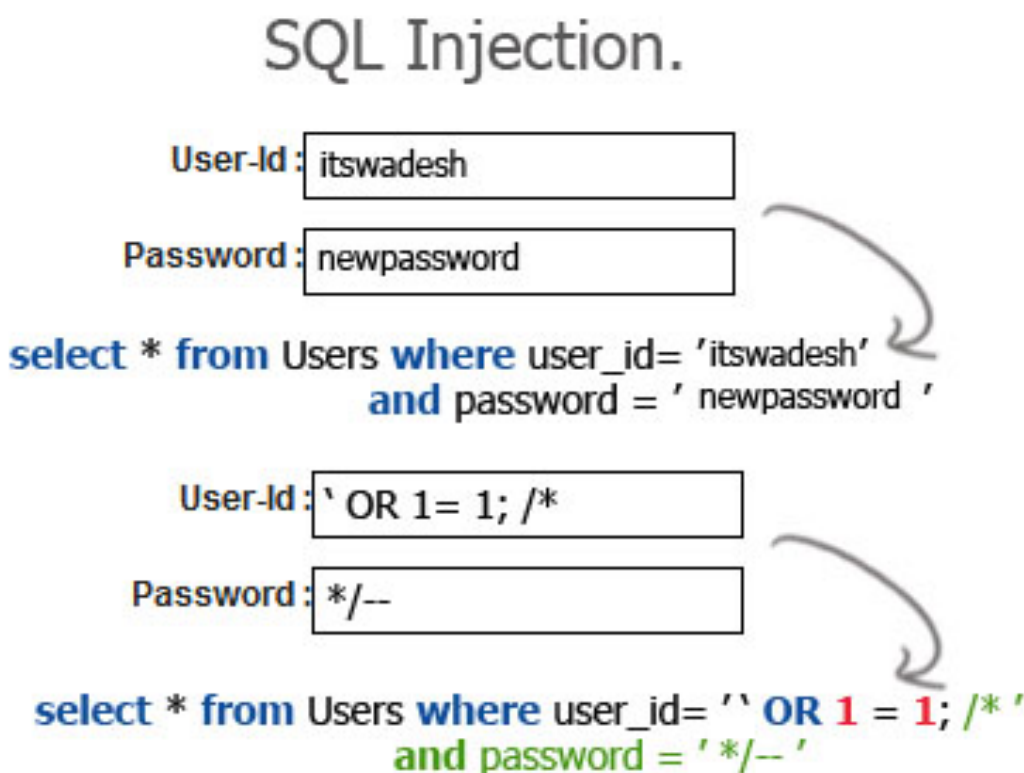


Figura 4 – Imagem ilustrando a exploração de um vulnerabilidade. Extraído de ([Gabriella Fonseca Ribeiro, 2011](#)).

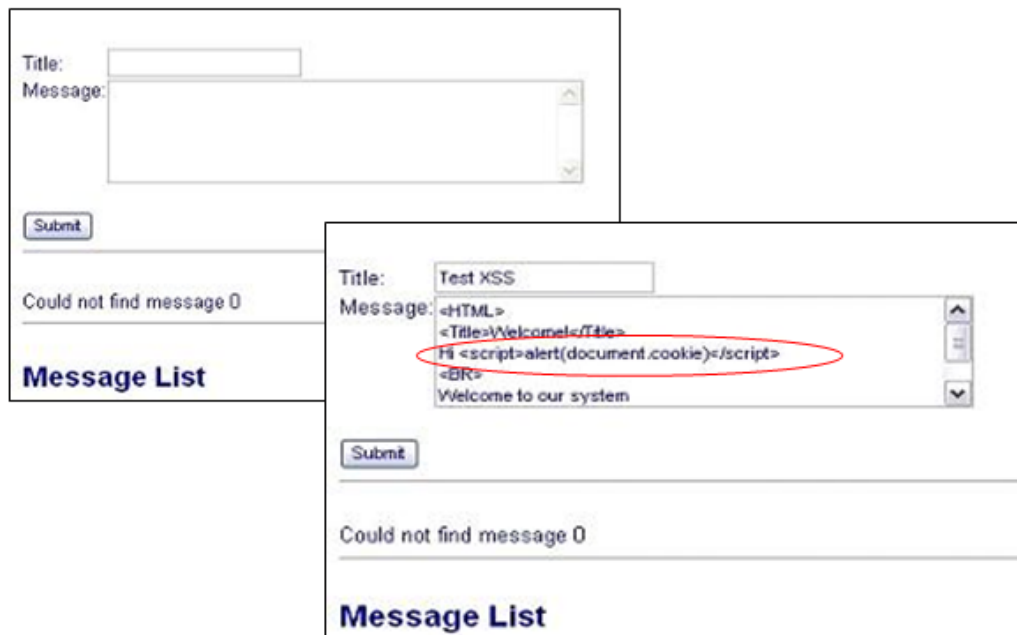


Figura 5 – Exemplo de exploração de um vulnerabilidade XSS. Extraído do portal (OWASP, 2013)

2.3 Scanners

Segundo HKSAR (2008), um *scanner* de vulnerabilidades é uma ferramenta capaz de acessar uma variedade de sistemas de informação fornecendo relatórios detalhados sobre vulnerabilidades encontradas no sistema. Sua arquitetura é formada de quatro componentes, conforme a Figura 6:

- Mecanismo de Verificação (*Scan Engine*): esse mecanismo executa a verificação e identificação de informações do sistema e vulnerabilidades de acordo com configurações e *plug-ins* instalados, podendo executar uma varredura para mais de um alvo por vez.
- Banco de Dados de Verificação (*Scan Database*): armazena as informações sobre vulnerabilidades, resultados e dados que serão utilizados no processo de verificação do *scanner*. As informações armazenadas serão dadas por cada *plug-in* adicionado. Logo deve armazenar os resultados de teste de cada *plug-in* além de uma descrição da vulnerabilidade e um identificador de CVE, do inglês *Common Vulnerabilities and Exposures*, ou Vulnerabilidades e Exposições Comuns.
- Módulo de Relatório (*Report Module*): provê relatórios para determinados tipos de usuário. Para o administrador de sistemas o *scanner* exibe um relatório técnico detalhado apontando o caminho e a uma forma de remediar ou solucionar o pro-

blema. Para o administrador de segurança são passados dados sumarizados. Para um executivo são exibidas informações, inclusive gráficos.

- Interface com Usuário (*User Interface*): permite interação com usuário, onde o mesmo pode configurar e passar parâmetros para o *software*. Essa interface pode ser gráfica ou apenas linha de comando.

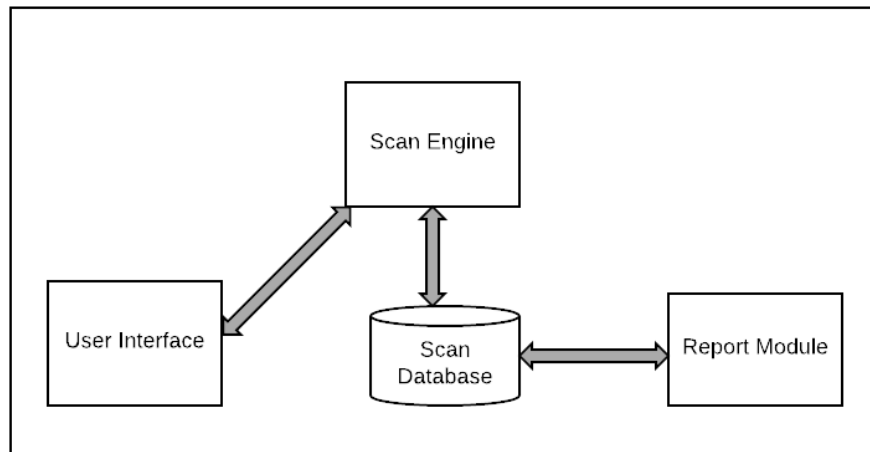


Figura 6 – Componentes do *Scanner*. Adaptado de (HKSAR, 2008).

Dentre os *scanners* mais populares estão: Vega, W3af, Golismero, Skipfish, Uniscan e Nikto. Todos esses *scanners* são gratuitos ou possuem versões gratuitas, segundo (OWASP, 2017).

2.4 Governo Eletrônico

Existem várias definições sobre o termo governo eletrônico. A definição escolhida nesse trabalho é dada por Evans e Yen (2006), sendo: "*Simply speaking, E-Government means the communication between the government and its citizens via computers and a Web-enabled presence*".

Em uma tradução não literal, os e-govs podem ser vistos como um meio de comunicação entre cidadão e Governo através de computadores conectados à internet. Essa comunicação reduz os custos e a burocracia de serviços governamentais, reduzindo os custos governamentais e o tempo gasto em atendimento ao cidadão.

Ainda segundo Evans e Yen (2006), a automação das funções governamentais ajuda a disseminar informação e aumentar os níveis de atendimento ao cidadão. Esse tipo de atitude faz com que os cidadãos entendam que sua satisfação é algo relevante. Outro ponto seria a compreensão sobre a opinião popular dadas as informações recolhidas pelo portal.

Dessa forma, os portais de governos eletrônicos, permitem que as agências governamentais reduzam custos e centralizem a tomada de decisão e eliminem redundâncias ineficientes e onerosas.

Os *sites* de governo eletrônico compreendem a municípios, estados e governo federal. Os *sites* municipais correspondem *sites* de municípios e prefeituras, por exemplo *site* da prefeitura de São Paulo. *Sites* estaduais correspondem aos *sites* do governo de cada estado, por exemplo *site* do estado do Acre. O Governo Federal também possui seu próprio *site*, que pode ainda acessar o *site* de estados e prefeituras.

2.5 Trabalhos Correlatos

Nesta sessão serão descritos os trabalhos relacionados ao tema desse trabalho: análise de vulnerabilidades em portais de governos eletrônicos. Serão discutidos a seguir os trabalhos de (SANTOS et al., 2014), (MONTEVERDE; CAMPIOLO, 2014), (DOUPÉ; COVA; VIGNA, 2010), (ROCHA; KREUTZ; TURCHETTI, 2011), (VIEIRA; ANTUNES; MADEIRA, 2009), (FONSECA; VIEIRA; MADEIRA, 2007) e (ANTUNES; VIEIRA, 2015).

Santos et al. (2014) utiliza um modelo de abordagem semelhante ao trabalho aqui proposto. Sua abordagem faz uso de um único *scanner*, o Nessus, e os *sites* analisados compõe um total de 127 considerando estados e prefeitura, onde o foco é particularmente as prefeituras do Rio de Janeiro. O método utilizado por Santos et al. (2014) em seu trabalho é denominado *g-Quality*. Este método consiste da avaliação de *sites* considerando oito pontos: usabilidade, interoperabilidade, segurança, privacidade, veracidade da informação, agilidade do serviço e transparência. Os autores deixam explícito que o foco do artigo é no quesito segurança.

Na avaliação de segurança, Santos et al. (2014) leva em conta os seguintes critérios da ferramenta: *Security Notes*(o sistema fornece informações em erros ou simplesmente as exibe por configuração padrão), *Security Warnings*(o sistema pode apresentar vulnerabilidade dependendo de alguma condição), *Security Holes*(uma falha a ser explorada foi realmente encontrada). Partindo dessas três categorias os autores classificam as vulnerabilidades encontradas considerando o grau de risco dado. Essa classificação é feita através dos rótulos: *None*(sem risco), *Low*(informação que pode ser utilizada), *Medium*(falha de segurança que pode dar acesso a um Shell), *High*(comandos podem ser executados no servidor), *Serious*(vulnerabilidade facilmente explorada), *Critical*(o sistema já está comprometido).

Os resultados obtidos por Santos et al. (2014) levam em conta a comparação entre *sites* federais, capitais e prefeituras do Rio de Janeiro. Na consideração dos resultados ele considera que a cada 3 avisos, apenas um seja verdadeiro. Logo é considerado apenas 0,33 o

número de notificações de falhas. Das falhas encontradas nenhuma pode ser considerada como sendo crítica, onde em sua maioria as falhas foram de médio e alto risco como indicado no texto.

A conclusão de Santos et al. (2014) para os resultados, principalmente para prefeituras, não foram bons. Os estados e as capitais obtiveram melhores resultados do que os municípios, como esperado, muito possivelmente pela quantidade de recursos que são investidos. Foi observado um grande número de portas abertas em determinados *sites*, o que influenciou na quantidade de falhas detectadas. Por fim, os autores concluem dando soluções para tais problemas, onde podem ser sintetizadas em: dividir os serviços oferecidos em máquinas diferentes, reduzir o número de portas abertas, melhorias no servidores ou terceirização dos serviços de tecnologia.

Monteverde e Campiolo (2014) utiliza uma metodologia diferente da exposta por (SANTOS et al., 2014). Nela os autores dividem a análise em duas etapas: identificação das vulnerabilidades e exploração. Na etapa de identificação proposta pelos autores, são utilizadas as ferramentas *W3af* e *VEGA*. Estas ferramentas têm a função de identificar as vulnerabilidades a serem exploradas na etapa seguinte. Em conjunto com essas foi utilizada mais uma ferramenta, *Burp Suite*, que permite configurar um *proxy* local e capturar requisições permitindo que as informações obtidas através de cada requisição fossem utilizadas nas ferramentas de exploração.

A escolha dos *sites* feita por Monteverde e Campiolo (2014) se baseiam em categorias, as quais foram divididas em: *sites* de comércio eletrônico, religiosos, acadêmicos, grandes portais, *sites* que utilizam CMS(*Content Management System*), governamentais, regionais e de conteúdo adulto. Ao todo foram avaliados 16 *sites* dos 100 pretendidos. Isso ocorreu, segundo os autores, devido a bloqueios de segurança nos alvos, o que reduziu drasticamente o número de *sites* avaliados.

Ao considerar as vulnerabilidades, os autores utilizaram como referência a OWASP Top 10, a mesma classificação utilizada neste trabalho. Segundo Monteverde e Campiolo (2014), 33% das vulnerabilidades detectadas no experimento foram classificadas como sendo severas levando em conta a classificação feita pela OWASP. Dessas vulnerabilidades, 90% foram consideradas vulnerabilidades de injeção, mais precisamente injeção de código SQL. Após a análise, a etapa de exploração foi executada, sendo dividida nas seguintes partes: exploração de injeção de SQL, quebra de gerenciamento de sessão, configuração incorreta de Segurança, exposição de dados sensíveis e componentes com vulnerabilidades conhecidas. Segundo os autores a maior parte das vulnerabilidades foi explorada com sucesso, evidenciando assim a situação crítica da segurança dos sistemas analisados. Uma observação importante é que dentre os trabalhos aqui citados, o de Monteverde e Campiolo (2014) é o único a realizar a etapa de exploração. A conclusão do trabalho de Monteverde e Campiolo (2014) é feita observando que medidas básicas poderiam ser tomadas no

processo de desenvolvimento de *software*. Para os autores, tais medidas seriam: o conhecimento de processos de desenvolvimento seguro pelos desenvolvedores e o incentivo das empresas. Segundo eles há a urgência de se conscientizar o mercado e as empresas dos riscos envolvidos, pois prejuízos enormes podem ocorrer. [Monteverde e Campiolo \(2014\)](#) sugere como uma possível solução o treinamento de desenvolvedores e a utilização das recomendações da OWASP no processo de desenvolvimento.

[Doupé, Cova e Vigna \(2010\)](#) oferecem uma profunda análise de ferramentas de análise de vulnerabilidades, sendo elas: Acutenix, AppScan, Burp, Grendel-Scan, Hailstorm, Milesan, N-Stalker, NTOSpider, Paros, W3AF e Webinspect. A metodologia utilizada neste trabalho se baseia no teste de ferramentas gratuitas e pagas em um ambiente controlado, *WackoPico*, que possui um número conhecido de vulnerabilidades. Este ambiente foi desenvolvido pelos autores que por sua vez estabeleceram as falhas existentes no sistema. As vulnerabilidades presentes nesse sistema são, segundo [Doupé, Cova e Vigna \(2010\)](#), comumente presentes em *sites*, sendo desde variações de XSS(*Cross-Site Scripting*) a variações de SQL-i(*SQL Injection*).

Os testes dos *web scanners* levam em conta três estados de configuração: *INITIAL*(modo inicial), o qual foi passado para o *scanner* a página inicial do ambiente; *CONFIG*(modo configurado), onde foi dado ao *scanner* um *login*(usuário e senha) válidos; *MANUAL*(modo manual), onde foi colocado um *proxy* para cada ferramenta e onde o usuário executava ações sobre cada página contendo uma vulnerabilidade. Essa variedade de cenários permitiu compreender o comportamento de cada ferramenta onde os resultados apresentados forneceram uma métrica de avaliação e comparação.

[Doupé, Cova e Vigna \(2010\)](#) apresentam os resultados considerando positivos, negativos, falsos positivos e falsos negativos. A partir dessas considerações eles elaboram um grafo de dominância estrita. O grafo de dominância estrita considera que uma dada ferramenta A domina B se e somente se toda vulnerabilidade descoberta por B foi também descoberta por A em um nível mais básico de comparação, e também A alcançou vulnerabilidades não detectadas por B. Juntamente a isso foi realizada uma análise de capacidades que consiste em medir o ataque realizado, capacidade de autenticação e *crawling*. Os autores concluem que apesar de algumas ferramentas terem se saído melhores que outras, os resultados, ainda não foram bons, sendo notada a alta taxa de falsos positivos e problemas no *parser* HTML.

[Rocha, Kreutz e Turchetti \(2011\)](#) fazem uma análise comparativa da ferramenta Uniscan com outras ferramentas. A metodologia utilizada no trabalho se baseia na comparação das funcionalidades e da arquitetura do Uniscan em relação a ferramentas utilizadas no mercado. Essa comparação é feita realizando análises em cenários controlados e reais, além de considerar aspectos de arquitetura do software como: incorporação de *plugins*, paralelização no funcionamento(multi-thread), desempenho e confidencialidade dos resul-

tados.

O primeiro aspecto levado em conta por [Rocha, Kreutz e Turchetti \(2011\)](#) são as vulnerabilidades que são apenas tratadas pela ferramenta analisada. Dentre essas vulnerabilidades existem RCE(*Remote Command Execute*), RFI(*Remote File Include*) e LFI(*Local File Include*) que são o diferencial, em termos de análise, do software discutido. Outro fator discutido é a extensibilidade do Uniscan, que por ser de código aberto facilita o desenvolvimento de *plugins*. Esse último fator é facilitado pela arquitetura modular que facilita o desenvolvimento e da robustez ao sistema.

Nos cenários de testes realizados a ferramenta foi comparada com diversas concorrentes de mercado(em sua maioria ferramentas gratuitas). Além das vulnerabilidades citadas anteriormente, foram levadas em conta XSS, SQL-i e *Blind SQL-i*(uma variação de SQL-i). Considerando ambos os cenários, controlado e real, o Uniscan apresentou um excelente desempenho na varredura dos sites, detectando maior parte das vulnerabilidades e apresentando uma baixa taxa de falsos positivos. Com isso [Rocha, Kreutz e Turchetti \(2011\)](#) demonstra a eficácia da ferramenta, onde a mesma possui bons resultados em relação ao grupo de software gratuito a qual pertence.

[Vieira, Antunes e Madeira \(2009\)](#) realizam um estudo cuidadoso sobre ferramentas automatizadas de análise de segurança. A metodologia utilizada pelos autores consiste na avaliação de um conjunto de *web scanners*, centrados em quatro pontos principais: preparação, onde serão reunidas páginas alvo(em sua maioria de publicidade); execução, a execução das ferramenta escolhidas sobre os *sites* alvo; verificação, sendo uma verificação manual, utilizando características específicas de cada vulnerabilidade, para confirmar a vulnerabilidade; análise, análise dos resultados recolhidos.

A execução do experimento foi realizada em trezentos *sites*, obtidos aleatoriamente, de um conjunto de seis mil cento e oitenta. Esse conjunto foi reunido utilizando um site que lista *web services* públicos e de uma ferramenta de busca. Na análise desses *sites* os autores optaram por ferramentas comerciais onde o foco básico se deu nas vulnerabilidades SQL-i, *XPath Injection*, *Code execution*, *Buffer Overflow*, *Username/Password Disclosure* e *Server Path Disclosure*. Essas vulnerabilidades foram verificadas através da análise das entradas dadas por cada ferramenta e também pelas exceções levantadas após cada teste.

Na análise dos resultados, [Vieira, Antunes e Madeira \(2009\)](#), classificou as falhas em três categorias: falsos positivos, duvidosas(não confirmadas) e vulnerabilidades confirmadas. Segundo os autores a maior parte das vulnerabilidades detectadas são SQL-i. Cada ferramenta identificou um tipo diferente de falha e em locais diferentes, fazendo com que os autores considerassem a união das vulnerabilidades confirmadas como sendo resultado do experimento. [Vieira, Antunes e Madeira \(2009\)](#) concluem que os resultados das ferramentas de *scanners* não são efetivos devida a dificuldade de confirmar vulnerabilidades apresentadas, resultados diferentes de cada ferramenta e também devido ao alto

número de falsos positivos. Uma pesquisa futura considerada pelos autores consiste no desenvolvimento de uma ferramenta realmente efetiva.

Fonseca, Vieira e Madeira (2007) propõem em seu trabalho um *benchmark* de *web scanners*. O método utilizado pelos autores consiste na injeção de vulnerabilidades reais em uma aplicação web. Essas falhas foram injetadas utilizando o programa *G-SWIFT*, que emula os mais frequentes tipos de falha de alto nível. Isso ofereceu uma análise próxima do cenário real, além de proporcionar o conhecimento da existência de cada falha facilitando assim a avaliação do desempenho da ferramenta.

Segundo Fonseca, Vieira e Madeira (2007) produção do teste é composta de dois estágios no primeiro verifica-se trechos código que permitem injeções que resultam em possíveis falhas; já o segundo estágio consiste nas alterações de parte do código que irá produzir as falhas, gerando assim o ambiente que será verificado por cada ferramenta. Para que houvesse essas injeções, foram feitas algumas modificações na ferramenta no *parser* da ferramenta *G-SWIFT*, permitindo assim que ela funcionasse para linguagens usuais (PHP, Java, .NET e etc). Os autores selecionaram duas aplicações para serem analisadas: *MyReferences*, ferramenta para controle e uso de referências; *BookStore*, loja de livros gerada por uma ferramenta de criação rápida de sites.

A execução do experimento segue um algoritmo elaborado pelos autores. Esse algoritmo consiste basicamente em testar inicialmente a aplicação (o que é chamado *gold run*) para verificar se já não haviam falhas. Após isso é feito um backup da aplicação sem alterações e posteriormente são injetadas as falhas. As ferramentas rodam sobre o sistema modificado. Se uma falha for encontrada ela é comparada com a falha encontrada no sistema sem modificações. Sendo uma falha já existente o processo reinicia do zero. Caso seja uma nova falha, ela será verificada (falso positivo ou falha real) e contabilizada.

Os resultados encontrados por Fonseca, Vieira e Madeira (2007) indicam uma alta taxa de falsos positivos. A porcentagem de falsos positivos varia de 20% a 70% no conjunto de ensaio. Esses valores, segundo os autores, se da devido a ineficiência de métodos de testes utilizados em ferramentas de análise automatizadas. Essa ineficiência pode ser reduzida se uma metodologia de análise e verificação, tal qual a utilizada por Fonseca, Vieira e Madeira (2007), seja utilizada. Fonseca, Vieira e Madeira (2007) propõe como trabalho futuro um *benchmarking* de um maior número de ferramentas com objetivo de auxiliar o melhoramento de analisadores automatizados.

A metodologia de *benchmarking* utilizada por Antunes e Vieira (2015) consiste em usar ferramentas automatizadas de verificação de vulnerabilidades em códigos de *web services* com e sem vulnerabilidades, permitindo assim a verificação de desempenho da ferramenta. Os autores organizaram o estudo e avaliação nos seguintes pontos: métricas, define um critério de avaliação; carga de trabalho, define o conjunto de métodos que foram utilizados pelas ferramentas para execução da busca por vulnerabilidades; procedimento,

define os procedimentos e regras que foram utilizados nos experimentos.

[Antunes e Vieira \(2015\)](#) utiliza em sua métrica uma medida de precisão dada pelo total de verdadeiros positivos dividido pela soma de falsos positivos mais os verdadeiros positivos. Os autores também consideram as vulnerabilidades descobertas em código, possuindo um valor *recall* que é dado pela divisão dos verdadeiros positivos pelo número de vulnerabilidades verdadeiras (vulnerabilidades existentes no código). A partir dessas medidas ele utiliza uma função denominada *F-Measure* que dá a medida real de desempenho de uma ferramenta.

Os casos de estudo desse trabalho se baseiam em duas abordagens: *VDBenchWS-pd*, com uma carga de trabalho predefinida onde detectores de vulnerabilidades são baseados em testes de penetração; *PTBenchWS-ud*, baseado em uma carga de trabalho definida pelo usuário com uso de ferramentas de teste de penetração. Ambas as abordagens utilizam ferramentas capazes de detectar vulnerabilidades como *SQL-i* em serviços web chamados de *SOAP* (*Simple Object Access Protocol*, ou Protocolo de Acesso a Objeto Simples em português).

A abordagem *VDBenchWS-pd* mostra um grande número de falsos positivos, tanto considerando a análise por teste de penetração, tanto pela análise estática (análise do código). Os resultados dos testes de penetração com ferramentas automatizadas são os que tiveram pior desempenho em termos gerais. Para cada uma das ferramentas foram indicados altos valores de falsos positivos chegando a 61%, no pior caso. No caso das ferramentas estáticas foi obtido um valor de 49% de falsos positivos.

A segunda abordagem, *PTBenchWS-ud*, teve desempenho próximo da primeira, de acordo com ([ANTUNES; VIEIRA, 2015](#)). Apesar de esperados melhores valores, uma vez que são passados argumentos que facilitam o processo, esta abordagem teve um alto número de falsos positivos. Considerando as métricas de avaliação, os valores de falsos positivos do *PTBenchWS-ud* chegaram a uma máxima de 62%, em uma das ferramentas, fazendo com que os valores de *F-Measure* aumentassem. Com isso, foi gerado um equilíbrio nos resultados.

[Antunes e Vieira \(2015\)](#) concluem que a metodologia de *benchmark* usada no trabalho pode ser considerada rápida e de fácil uso, uma vez que consiste apenas em executar as dadas ferramentas para determinados *sites* e verificar a existência de vulnerabilidades. Ainda segundo os mesmos, a metodologia se mostrou eficiente também a analisadores estáticos. Os autores ressaltam que os pontos principais são as definições das métricas e da carga de trabalho, que definem a maior parte dos resultados obtidos. Os autores propõem como trabalhos futuros a extensão do *benchmark* a outras vulnerabilidades, domínios e ferramentas.

Os trabalhos aqui descritos foram de fundamental importância para elaboração da

metodologia utilizada. A seleção das ferramentas utilizadas nesse trabalho se assemelha aos *benchmarking* propostos por (DOUPÉ; COVA; VIGNA, 2010; ROCHA; KREUTZ; TURCHETTI, 2011; VIEIRA; ANTUNES; MADEIRA, 2009; FONSECA; VIEIRA; MADEIRA, 2007; ANTUNES; VIEIRA, 2015). A diferença entre os trabalho citados anteriormente está no uso exclusivo de ferramentas de código aberto. Outra semelhança aos trabalhos anteriormente descritos é o tema. Assim como no trabalho de Santos et al. (2014) e Monteverde e Campiolo (2014), o estudo é centrado em portais de governos eletrônicos. Esses dois trabalhos executam uma verificação das vulnerabilidades detectadas, seja por exploração ou por análise das entradas das ferramentas, o que está fora do escopo do presente trabalho.

3 Desenvolvimento

Este capítulo tem o objetivo de discutir o desenvolvimento do trabalho levando em consideração a execução de cada uma de suas etapas. Inicialmente a metodologia será apresentada em detalhes e, posteriormente, serão apresentados os resultados assim como uma análise dos mesmos. Também é discutida a hipótese central deste trabalho, que consiste na relação econômica entre segurança e nível econômico do estado.

3.1 Metodologia

A metodologia utilizada nesse trabalho se assemelha muito aos métodos utilizados pelos autores dos trabalhos correlatos, principalmente no que tange a execução de *benchmark*. Como citado anteriormente, a execução do trabalho foi dividida em quatro passos:

1. Seleção das ferramentas;
2. Seleção dos portais;
3. Varredura dos *sites*;
4. Análise dos resultados obtidos.

O principal critério para a seleção das ferramentas é sua disponibilidade, onde todas as ferramentas utilizadas devem ser gratuitas com atualizações recentes e de fácil acesso. Com base nisso, foram inicialmente selecionadas quatro ferramentas: Uniscan, Skipfish, W3AF e Vega. Todas as ferramentas são gratuitas e oferecem um relatório minimamente detalhado do ambiente onde são executadas.

Para testar essas ferramentas foi utilizado um ambiente controlado, WackoPicko, que está presente no trabalho de [Doupé, Cova e Vigna \(2010\)](#). Esse ambiente possui um conjunto variado de vulnerabilidades, fornecendo escopo ideal para que seja compreendido o funcionamento e o alcance de cada um dos *scanners*.

As ferramentas foram analisadas inicialmente de acordo com suas funcionalidades, seguindo os seguintes critérios:

- Relatório: o relatório deve detalhar cada vulnerabilidade, além de indicar o ponto onde a mesma aconteceu. Devem existir opções de persistência para o dado relatório
- Usabilidade: o quão fácil é utilizar a ferramenta.

- Eficácia: a quantidade de vulnerabilidades detectadas pela ferramenta, considerando a classificação da OWASP, (OWASP et al., 2013).

A partir desses pontos foi elaborada a Tabela 1 que classifica as ferramentas considerando uma nota de 1 a 3 para cada critério, onde 1 significa que não atendeu minimamente os requisitos, 2 atendeu os requisitos e 3 possui requisitos acima do necessário.

Tabela 1 – Avaliação das ferramentas

Estados	Relatório	Usabilidade	Eficácia
Skipfish	3	2	3
Uniscan	2	2	3
Vega	1	3	3
W3AF	1	2	3

Em termos de eficácia todas as ferramentas cumprem, e até superam, os requisitos. Ao considerarmos usabilidade o cenário se torna um pouco diferente, com destaque do *scanner* Vega que possui interface intuitiva com o usuário. W3AF também possui versão com interface gráfica, entretanto essa versão possui vários *bugs* e é instável.

Apesar de possuir relatório elaborado, tanto Vega quanto W3AF receberam notas baixas. O primeiro por não permitir exportar a saída para outros formatos, fazendo com que o usuário fique preso ao ambiente, e o segundo por possuir um erro em sua última versão o que não permite que o relatório seja armazenado limitando-se aos resultados do terminal. Essas duas ferramentas também não possuem mais suporte no Kali Linux.

Considerando os pontos necessário foram escolhidas as ferramentas Skipfish e Uniscan. Essas ferramentas são disponibilizadas pelo ambiente de *pentest*, Kali Linux, em sua versão 2017.2. Essa versão do linux foi a versão utilizada para a execução das análises, onde foi utilizado o ambiente de máquinas virtuais da *Oracle*, o *VirtualBox* versão 5.8.1.

Uma ferramenta extra foi também selecionada para ser utilizada nos experimentos. O *scanner* de portas NMAP. Essa ferramenta é um software gratuito que permite verificar o estado das portas de comunicação TCP/UDP de um sistema final. Isso permitiu observar as portas abertas de cada *site* escolhido como alvo dos testes.

Os *sites* escolhidos como alvos foram os dos governos estaduais de cada estado, e também o *site* do Governo Federal, totalizando 28 portais. Os *sites* escolhidos permitem traçar uma relação econômica e tecnológica nos levando diretamente para a hipótese deste trabalho:

Hipótese: Existe uma relação entre segurança dos portais e o nível econômico de cada estado aos quais eles pertençam onde estados mais ricos tendem a ter *sites* de governo eletrônico mais seguros que estados mais pobres.

Ou seja, nossa suposição é a de que existe uma relação direta entre o PIB dos estados e a segurança dos portais. Um exemplo disso seria uma maior quantidade de vulnerabilidades no estado de Santa Catarina do que no estado de São Paulo, já que o PIB de São Paulo é maior do que o PIB de Santa Catarina.

A veracidade desta hipótese será verificada nas duas próximas seções: Resultados e Análise. Os resultados considerados levam em conta as vulnerabilidades presentes no trabalho de (OWASP et al., 2013), onde a métrica de classificação é diretamente proporcional ao número de vulnerabilidades, e também a gravidade das mesmas. Assim quanto mais vulnerabilidades, e quanto maior o risco que elas representam menor será a classificação da segurança de um dado portal.

3.2 Resultados

A partir dos primeiros meses de 2017 iniciou-se a seleção das ferramentas e dos portais web. Em Maio começaram os primeiros experimentos, onde foram gerados *scripts* para automatizar o processo de execução e configuração de cada ferramenta. Ao final desse mês foram iniciados os experimentos onde cada ferramenta foi executada para os 28 portais. Problemas com a instabilidade da rede utilizada, resultaram no atraso de alguns dias fazendo com que as varreduras fossem concluídas apenas no início de Junho.

Ao desconsiderarmos os eventuais problemas ocorridos durante as varreduras e a instabilidade da rede, foi encontrado um valor médio de 6 horas no Skipfish e 4 horas no Uniscan por portal. Foi obtido para a primeira ferramenta um total de 168 horas ou 7 dias ininterruptos de execução. Para segunda ferramenta foi obtido um total de 112 horas ou aproximadamente 5 dias ininterruptos de execução. No total o experimento levou em média 280 horas ou aproximadamente 12 dias de trabalho ininterrupto. Esse valor aumenta se considerarmos a execução da ferramenta NMAP, com um média de tempo de 15 minutos para cada portal e 7 horas totais de execução, totalizando assim 12 dias e 7 horas de execução.

A Tabela 2 ilustra os resultados apresentados por cada uma das ferramentas, onde as linhas tracejadas representam os testes que não foram concluídos ou não foram executados. Essa execução ou não conclusão pode estar ligada a mecanismos de segurança presentes nos portais, como políticas específicas de *Firewall* ou adoção de Sistemas de Detecção/Prevenção de Intrusão.

O *scanner* Skipfish apresentou um maior número de vulnerabilidades em ambiente real, assim como nós testes realizados nos experimentos de *benchmark*. O Uniscan apresentou apenas uma vulnerabilidade detectada sendo a mesma no portal do estado da Bahia.

Tabela 2 – Tabela de Resultados por Portal

Estados	Skipfish	Uniscan
Governo Federal	2048	-
Acre(AC)	1	0
Alagoas(AL)	29	-
Amapá(AP)	-	0
Amazonas(AM)	19	0
Bahia(BA)	1650	1
Ceará(CE)	267	0
Distrito Federal(DF)	-	-
Espírito Santo(ES)	-	0
Goiás(GO)	126	0
Maranhão(MA)	29	0
Mato Grosso(MT)	-	-
Mato Grosso do Sul(MS)	-	0
Minas Gerais(MG)	186	-
Pará(PA)	7	0
Paraíba(PB)	-	0
Paraná(PR)	26	0
Pernambuco(PE)	-	0
Piauí(PI)	134	0
Rio de Janeiro(RJ)	166	-
Rio Grande do Norte(RN)	39	0
Rio Grande do Sul(RS)	-	0
Rondônia(RO)	-	0
Roraima(RR)	-	0
Santa Catarina(SC)	-	0
São Paulo(SP)	23	-
Sergipe(SE)	46	-
Tocantins(TO)	398	0

A Tabela 3 ilustra a distribuição entre as vulnerabilidades descobertas e a classificação da Owasp et al. (2013), levando em conta uma classificação das 10 vulnerabilidades mais graves e recorrentes e sistemas web. Pode ser observado que a vulnerabilidade com o maior número de ocorrências é a A10, seguido de A2 e A1. Além disso é observado que a única vulnerabilidade detectada pelo Uniscan é do tipo A1.

A utilização da sigla A seguida do número se refere a cada uma das vulnerabilidades citadas na seção 2 sendo descritas abaixo:

- A1 - Injeção
- A2 - Quebra de Autenticação e Gerenciamento de Sessão
- A3 - *Cross-Site Scripting(XSS)*
- A4 - Referência Insegura e Direta a Objetos

- A5 - Configuração Incorreta de Segurança
- A6 - Exposição de Dados Sensíveis
- A7 - Falta de Função para Controle de Nível de Acesso
- A8 - *Cross Site Request Forgery(CSRF)*
- A9 - Utilização de Componentes Vulneráveis Conhecidos
- A10 - Redirecionamento e Encaminhamentos Inválidos

Tabela 3 – Vulnerabilidades de acordo com classificação de (OWASP et al., 2013)

Vulnerabilidades	Skipfish	Uniscan
A1	22	1
A2	0	0
A3	131	0
A4	0	0
A5	0	0
A6	0	0
A7	0	0
A8	0	0
A9	0	0
A10	5039	0

Na Tabela 4 podem ser observadas as distribuições das vulnerabilidades mostradas na Tabela 3 levando em conta o portal na qual a mesma foi detectada, ainda considerando a notação anterior. Da mesma forma que a Tabela 2, as linhas tracejadas representam que a execução do *scanner* no dado portal não pode ser efetuada.

A Tabela 5 mostra os resultados obtidos através da ferramenta NMAP. Na coluna "Porta Padrão" é considerada a presença da porta 80, 8080 e/ou 443. As 80 e 8080 portas padrão usadas para identificar um serviço web, e 443 é esperada em um servidor web pois identifica o uso de TLS (HTTPS). "X" indica a presença da porta 80, 8080 e/ou 443. Na coluna "Outras Portas" é levado em consideração o conjunto de portas que não são usualmente utilizadas em servidores web. A coluna número de portas considera a "Porta Padrão" juntamente com o número de "Outras Portas". Os portais que possuem uma linha tracejada não puderam ser verificados pois assim como citado anteriormente, alguns portais possuem mecanismos de proteção que impossibilitam a ação da ferramenta. O NMAP, diferentemente das outras ferramentas, indica a presença dessas proteções em seus resultados.

O método de classificação de cada ferramenta influencia na quantidade de vulnerabilidades mostradas em cada tabela. Em particular, a ferramenta Uniscan apresenta

Tabela 4 – Tabela de Resultados por Portal

Portais	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
Governo Federal	0	0	0	0	0	0	0	0	0	2048
Acre(AC)	1	0	0	0	0	0	0	0	0	0
Alagoas(AL)	0	0	0	0	0	0	0	0	0	29
Amapá(AP)	-	-	-	-	-	-	-	-	-	-
Amazonas(AM)	0	0	0	0	0	0	0	0	0	19
Bahia(BA)	1	0	0	0	0	0	0	0	0	1649
Ceará(CE)	0	0	0	0	0	0	0	0	0	267
Distrito Federal(DF)	-	-	-	-	-	-	-	-	-	-
Espírito Santo(ES)	-	-	-	-	-	-	-	-	-	-
Goiás(GO)	0	0	0	0	0	0	0	0	0	126
Maranhão(MA)	0	0	0	0	0	0	0	0	0	29
Mato Grosso(MT)	-	-	-	-	-	-	-	-	-	-
Mato Grosso do Sul(MS)	-	-	-	-	-	-	-	-	-	-
Minas Gerais(MG)	8	0	0	0	0	0	0	0	0	178
Pará(PA)	0	0	0	0	0	0	0	0	0	7
Paraíba(PB)	-	-	-	-	-	-	-	-	-	-
Paraná(PR)	0	0	1	0	0	0	0	0	0	25
Pernambuco(PE)	-	-	-	-	-	-	-	-	-	-
Piauí(PI)	1	0	0	0	0	0	0	0	0	133
Rio de Janeiro(RJ)	10	0	130	0	0	0	0	0	0	26
Rio Grande do Norte(RN)	0	0	0	0	0	0	0	0	0	48
Rio Grande do Sul(RS)	-	-	-	-	-	-	-	-	-	-
Rondônia(RO)	-	-	-	-	-	-	-	-	-	-
Roraima(RR)	-	-	-	-	-	-	-	-	-	-
Santa Catarina(SC)	-	-	-	-	-	-	-	-	-	-
São Paulo(SP)	2	0	0	0	0	0	0	0	0	21
Sergipe(SE)	0	0	0	0	0	0	0	0	0	36
Tocantins(TO)	0	0	0	0	0	0	0	0	0	398

resultados de forma bem simples, sendo exibidos apenas a vulnerabilidade seguida do *link* e dos locais onde a mesma foi encontrada.

O Skipfish, além de encontrar o maior número de vulnerabilidades, apresenta os dados de forma mais detalhada. A ferramenta faz uma classificação de cores sobre a dada vulnerabilidade, representada por uma esfera seguida do nome.

As Figuras 7 e 8 ilustram a interface de cada ferramenta.

É considerada a seguinte classificação para a ferramenta Skipfish:

- Vermelha: vulnerabilidade de alto risco;
- Laranja: vulnerabilidade de risco médio;
- Azul: vulnerabilidade de baixo risco.



Scanner version: 2.10b Scan date: Thu Jun 15 16:18:10 2017
 Random seed: 0x9e40ff89 Total time: 4 hr 29 min 30 sec 971 ms
[Problems with this scan? Click here for advice.](#)

Crawl results - click to expand:

<http://www.saopaulo.sp.gov.br/> 2 8 62 415 627 347
 Code: 200, length: 51087, declared: text/html, detected: application/xhtml+xml, charset: UTF-8 [[show trace +](#)]

Document type overview - click to expand:

application/rss+xml (3)
application/xhtml+xml (19)
text/html (4)

Issue type overview - click to expand:

Shell injection vector (2)
Incorrect or missing charset (higher risk) (4)
External content embedded on a page (higher risk) (4)
HTML form with no apparent XSRF protection (45)
External content embedded on a page (lower risk) (17)
Node should be a directory, detection error? (9)
Response varies randomly, skipping checks (19)
Parent behavior checks failed (no brute force) (36)
Limits exceeded, fetch suppressed (168)
Resource fetch failed (183)

Figura 7 – Imagem da ferramenta Skipfish. Imagem produzida pelo autor.



SCAN TIME Scan Started: 15/5/2017 1:5:41
TARGET Domain http://www.ap.gov.br/ Server Banner: nginx Target IP: 177.84.201.15
CRAWLING Directory check: File check: CODE: 200 URL: http://www.ap.gov.br/libraries/joomla/utilities/compat/php50x.phpcart2_0/acart2_0.mdb CODE: 200 URL: http://www.ap.gov.br/access.log CODE: 200 URL: http://www.ap.gov.br/active.log CODE: 200 URL: http://www.ap.gov.br/admin/cpllogfile.log CODE: 200 URL: http://www.ap.gov.br/admin/database/wwForum.mdb CODE: 200 URL: http://www.ap.gov.br/Admin_files/order.log CODE: 200 URL: http://www.ap.gov.br/adovbs.inc CODE: 200 URL: http://www.ap.gov.br/ban.log CODE: 200 URL: http://www.ap.gov.br/blahb.ida CODE: 200 URL: http://www.ap.gov.br/blahb.idq Check robots.txt: Check sitemap.xml: Crawling finished, found: 1 URL's Source Code Disclosure: File Upload Forms:

Figura 8 – Imagem da ferramenta Uniscan. Imagem produzida pelo autor.

O número de vulnerabilidades de um dado tipo é exibido na frente do nome do mesmo. Alguns dos altos valores nas tabelas podem ser associados a repetições, uma mesma vulnerabilidade contabilizada duas vezes, uma vez que um mesmo tipo de vulnerabilidade pode ser classificado como de alto e médio risco, por exemplo. Isso ocorre pois

não foi medido o risco de cada vulnerabilidade e sim contabilizada sua existência.

O Skipfish detectou mais vulnerabilidades do que aquelas presentes na tabela da (OWASP et al., 2013) mas as mesmas não foram contabilizadas pois o foco do trabalho é usar a classificação proposta pela OWASP.

Tabela 5 – Tabela de Resultados por Portal,NMAP

Estados	Porta padrão	Outras Portas	Número de Portas
Governo Federal	X	-	2
Acre(AC)	X	-	1
Alagoas(AL)	X	-	1
Amapá(AP)	-	-	-
Amazonas(AM)	X	-	2
Bahia(BA)	X	-	1
Ceará(CE)	X	-	2
Distrito Federal(DF)	X	22, 111, 9080, 9090	6
Espírito Santo(ES)	X	-	2
Goiás(GO)	X	21, 2121	4
Maranhão(MA)	X	-	2
Mato Grosso(MT)	X	-	1
Mato Grosso do Sul(MS)	X	-	1
Minas Gerais(MG)	X	-	2
Pará(PA)	X	21, 3389	3
Paraíba(PB)	X	-	1
Paraná(PR)	X	-	2
Pernambuco(PE)	X	-	2
Piauí(PI)	X	-	1
Rio de Janeiro(RJ)	X	-	1
Rio Grande do Norte(RN)	X	8008, 8010	4
Rio Grande do Sul(RS)	X	200, 5060	4
Rondônia(RO)	-	-	-
Roraima(RR)	X	21, 22	4
Santa Catarina(SC)	-	-	-
São Paulo(SP)	X	68	1
Sergipe(SE)	X	55	1
Tocantins(TO)	X	-	2

3.3 Análise dos resultados

Nesta seção discutiremos os resultados obtidos levando em conta aspectos de segurança e também a hipótese central do trabalho.

A execução dos experimentos ocasionou em um grande número de dados recolhidos que estão resumidos nas tabelas da seção anterior. Foram detectadas um total de 5193 vulnerabilidades, pertencendo aos tipos: Injeção (A1), XSS (A3) e Redirecionamento e Encaminhamentos Inválidos (A10). Em sua maioria as vulnerabilidades são de Redirecionamento e Encaminhamentos Inválidos com 97%, seguido de XSS com 2,5% e Injeção com aproximadamente 0.5%. A Figura 9 ilustra essa distribuição.

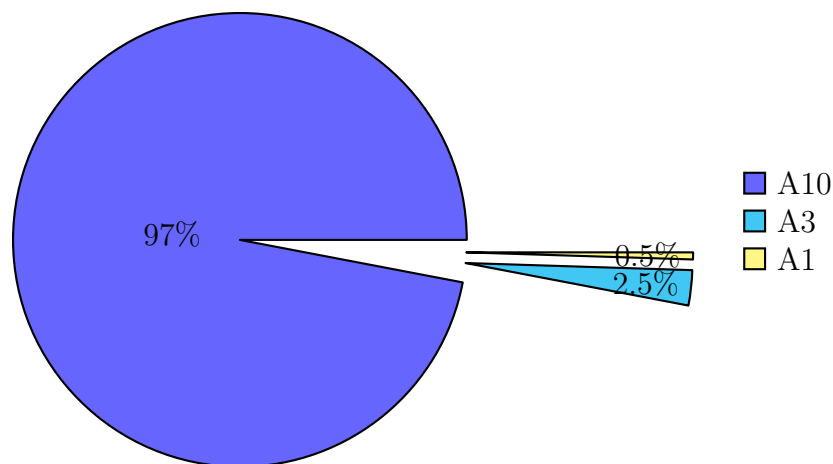


Figura 9 – Proporção de cada tipo de vulnerabilidade detectada nos resultados. Produzido pelo autor

A vulnerabilidade A10 está presente em quase todos os portais que foram analisados com sucesso pela ferramenta *Skipfish*, com exceção do Acre. O portal que apresentou o maior número de vulnerabilidades desse tipo foi o portal do governo federal, correspondendo aproximadamente 39.43% das vulnerabilidades detectadas e 40.6%, especificamente, do tipo A10. A Tabela 6 mostra os *sites* ordenados pelo número de vulnerabilidades do tipo A10.

Um exemplo de vulnerabilidade detectada foi um redirecionamento para a página “Cultura”, no *site* do Governo Federal. Na barra de redirecionamento há uma URL, que redireciona com um final “barra.js”. Em alguns casos é comum que alguns argumentos sejam passados (por exemplo “/barra.js?url = exemplo.com”). Caso um argumento, URL, fosse passado um atacante poderia criar uma URL maliciosa que redireciona os usuários para um *site*, que executa *phishing* e instala *malware*.

Apesar dos altos valores da vulnerabilidade A10 presentes no *site* do Governo Federal, o mesmo apresenta um resultado positivo em relação a sua segurança. Isso ocorre, pois, parte dos valores detectados na análise podem ser falsos positivos. Além disso, a

Tabela 6 – *Sites* ordenados pelo número de vulnerabilidades do tipo A10.

<i>Sites</i>	Número de vulnerabilidades do tipo A10
Governo Federal	2048
Bahia(BA)	1649
Tocantins(TO)	398
Ceará(CE)	267
Minas Gerais(MG)	178
Piauí(PI)	133
Goiás(GO)	126
Rio Grande do Norte(RN)	48
Sergipe(SE)	36
Alagoas(AL)	29
Maranhão(MA)	29
Rio de Janeiro(RJ)	26
Paraná(PR)	25
São Paulo(SP)	21
Amazonas(AM)	19
Pará(PA)	7

vulnerabilidade A10 é uma das vulnerabilidades de menor severidade, segundo (OWASP et al., 2013), devido ao seu próprio impacto e de poder ser facilmente detectada. Outro grande responsável pelo número de vulnerabilidades do tipo A10 foi o portal do estado da Bahia com aproximadamente 32.7% das vulnerabilidades desse tipo, e 31.7% do total de vulnerabilidades descobertas. Nesse mesmo portal foi detectado uma vulnerabilidade do tipo A1, sendo um dos tipos de maior severidade.

O segundo tipo de vulnerabilidade mais numeroso nos testes foi a A3, *XSS*(*Cross-Site Scripting*), com 131 vulnerabilidades detectadas o que corresponde a aproximadamente 2.5% do total de vulnerabilidades analisadas. A Tabela 7 ordena os *sites* que possuem vulnerabilidades do tipo A3 por sua quantidade.

Tabela 7 – *Sites* ordenados pelo número de vulnerabilidades do tipo A3.

<i>Sites</i>	Número de vulnerabilidades do tipo A3
Rio de Janeiro(RJ)	130
Paraná(PR)	1

O maior responsável pelas vulnerabilidades de tipo A3 foi o Rio de Janeiro contribuindo com aproximadamente 99.2% do total de vulnerabilidades desse tipo. Aqui também deve ser considerada a possível existência de falsos positivos, mas ainda deve ser levando em conta que este tipo de vulnerabilidade é classificado como um dos tipos de maior severidade. A análise também indicou que o *site* do estado do Paraná contém uma vulnerabilidade do tipo A3.

Um exemplo de vulnerabilidade A3 detectada foi em um *link* da pagina de comunicação com o usuário do portal do Rio de Janeiro. Este *link* recebe uma um parâ-

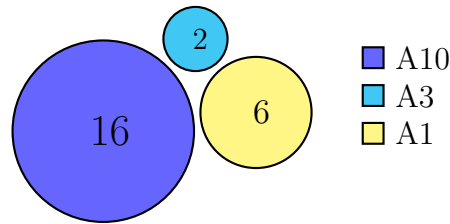


Figura 10 – Gráfico representando a quantidade de *sites* com determinado tipo de vulnerabilidade. Produzido pelo autor

metro(*string*), como no exemplo "+request.getParameter("exemplo") + ">";. Caso um atacante altere esse parâmetro para um *script* malicioso, <script>document.location='http://exemploataque.com/cgi-bin/cookie.cgi?foo='+document.cookie</script>", torna-se possível sequestrar as sessões de usuários da página.

O terceiro e último tipo de vulnerabilidade detectado foi o A1, Injeção. Ele não está presente em números tão expressivos quanto A3 e A10, entretanto ele supera A3 em números de *sites* que apresentam a vulnerabilidade. A Figura 10 ilustra o cenário.

A Tabela 8 ordena os *sites* que possuem vulnerabilidades do tipo A1 por sua quantidade.

Tabela 8 – *Sites* ordenados pelo número de vulnerabilidades do tipo A1.

<i>Sites</i>	Número de vulnerabilidades do tipo A1
Rio de Janeiro(RJ)	10
Minas Gerais(MG)	8
São Paulo(SP)	2
Acre(AC)	1
Bahia(BA)	1
Piauí(PI)	1

O estado do Rio de Janeiro mais uma vez lidera em número de vulnerabilidades detectadas, agora com vulnerabilidade do tipo A1. Esse tipo de resultado chama atenção, uma vez que a vulnerabilidade A1 é a primeira colocada na classificação feita por (OWASP et al., 2013). Isso faz com que esse portal seja um dos menos bem avaliados em termos de severidade de vulnerabilidades encontradas, pois o mesmo possui dois dos tipos de vulnerabilidades mais preocupantes.

Como exemplo de vulnerabilidade A1, podemos tomar O URL para página de consulta da casa civil do Rio de Janeiro (".../casacivil/exibeconteudo?"). Nela podemos notar uma consulta semelhante a um comando SQL (uso de um possível *framework*). Se os campos dessa URL são parâmetros, então podemos colocar comandos como ' ou '1'=1. Esse comando retorna todos os registros da tabela. Isso pode romper com a confidencialidade dos dados armazenados pelo *site*.

Minas Gerais é o segundo *site* a possuir maior número de vulnerabilidades do tipo

A1, seguido por São Paulo. Acre, Bahia e Piauí tiveram o mesmo número de vulnerabilidades do tipo A1 detectadas, fazendo com que a ordenação nesse caso seja alfabética levando em conta a primeira letra dos seus respectivos nomes.

A Tabela 9 classifica a situação de cada estado levando em conta a sua posição no trabalho de (OWASP et al., 2013). Os primeiros colocados apresentam os piores resultados em termos de severidade das vulnerabilidades encontradas.

Tabela 9 – Classificação dos portais levando em conta o trabalho de (OWASP et al., 2013).

<i>Sites</i>	A1	A3	A10
Rio de Janeiro(RJ)	10	130	26
Minas Gerais(MG)	8	0	178
São Paulo(SP)	2	0	21
Bahia(BA)	1	0	1649
Piauí(PI)	1	0	133
Paraná(PR)	0	1	25
Acre(AC)	1	0	0
Governo Federal	0	0	2048
Tocantins(TO)	0	0	398
Ceará(CE)	0	0	267
Goiás(GO)	0	0	126
Rio Grande do Norte(RN)	0	0	48
Sergipe(SE)	0	0	36
Alagoas(AL)	0	0	29
Maranhão(MA)	0	0	29
Amazonas(AM)	0	0	19
Pará(PA)	0	0	7

Além de vulnerabilidades também foram obtidos resultados pela ferramenta de análise NMAP, que identifica as portas abertas de um determinado endereço. O Apêndice A contem a descrição da função de cada porta encontrada na Tabela 5 do capítulo anterior.

Segundo Beaver (2007), algumas das portas detectadas são alvos recorrentes de ataques e podem até mesmo representar vulnerabilidades. Dentre essas portas estão:

- 21 - Localizada em Goiás, Pará, Roraima
- 22 - Localizada em Roraima e Distrito Federal
- 3389 - Localizada no Pará

Essas portas, por sua vez, identificam: 21, FTP, serviço de transmissão de arquivos e 22, serviço de acesso remoto seguro (SSH) para computadores com sistemas operacionais baseado em Unix. A porta 3389 identifica o protocolo RDP que é um serviço de acesso remoto do Windows. A porta 21 está presente na maior parte dos portais, seguida das

portas 22, 3389. A transmissão de arquivos pela porta 21 não permite o acesso a arquivos confidenciais que estejam no servidor mas, eventualmente, poderia permitir o envio de arquivos e programas maliciosos. As portas 22 e 3389 permitem que um usuário se conecte remotamente ao computador. Ambas as portas representam um alto risco para o servidor Web, uma vez que caso o atacante consiga estabelecer comunicação, poderá ter acesso a serviços do servidor, podendo comprometer tanto integridade, quanto disponibilidade e confidencialidade.

Após apresentar uma análise dos resultados obtidos no processo de varredura e investigação dos portais, é possível abordar a hipótese central deste trabalho:

Hipótese: Existe uma relação entre segurança dos portais e o nível econômico de cada estado aos quais eles pertençam, onde, estados mais ricos tendem a ter *sites* de governo eletrônico mais seguros que estados mais pobres.

Levando em conta os dados de [Wikipédia \(2017\)](#), retirados no IBGE, podemos afirmar que a hipótese falha. Isso ocorre pois os três estados mais ricos do país (São Paulo, Minas Gerais e Rio de Janeiro) se encontram no topo da tabela de resultados com o maior número de vulnerabilidades de alto risco. Além disso grandes portais, como os dos governos do Governo Federal, Distrito Federal e Paraná, possuem portas em aberto que possibilitam o aumento das chances de sucesso de um ataque.

A Tabela 10 sintetiza, em termos gerais, os resultados dessa pesquisa, considerando como classificação a severidade das vulnerabilidades detectadas. Esse resultado torna evidente a contradição e a da hipótese evidenciando assim que não foi possível traçar uma relação aparente entre a economia dos estados e a segurança de seus portais.

Tabela 10 – Classificação levando em conta análise dos resultados.

<i>Sites</i>	A1	A3	A10	Presença de portas não padrão
Rio de Janeiro(RJ)	10	130	26	-
Minas Gerais(MG)	8	0	178	-
São Paulo(SP)	2	0	21	X
Bahia(BA)	1	0	1649	-
Piauí(PI)	1	0	133	-
Paraná(PR)	0	1	25	-
Acre(AC)	1	0	0	-
Governo Federal	0	0	2048	-
Tocantins(TO)	0	0	398	-
Ceará(CE)	0	0	267	-
Goiás(GO)	0	0	126	X
Rio Grande do Norte(RN)	0	0	48	X
Sergipe(SE)	0	0	36	X
Alagoas(AL)	0	0	29	-
Maranhão(MA)	0	0	29	-
Amazonas(AM)	0	0	19	-
Pará(PA)	0	0	7	X

4 Conclusão

Este trabalho analisou as vulnerabilidades de portais de Governos Eletrônicos buscando associar a riqueza dos estados dos portais com seu nível de segurança sendo baseado nos trabalhos de (SANTOS et al., 2014), (MONTEVERDE; CAMPIOLO, 2014), (DOUPÉ; COVA; VIGNA, 2010), (ROCHA; KREUTZ; TURCHETTI, 2011), (VIEIRA; ANTUNES; MADEIRA, 2009), (FONSECA; VIEIRA; MADEIRA, 2007) e (ANTUNES; VIEIRA, 2015).

O estudo conseguiu atingir resultados concretos, que serviram como material para que a hipótese proposta fosse negada, ou seja, não foram encontrados indícios de que portais de governo eletrônico de estados mais ricos sejam mais seguros (do ponto de vista de vulnerabilidades de segurança) do que portais de estados mais pobres.

A análise das vulnerabilidades, levando em conta sua severidade a partir da classificação (OWASP et al., 2013), permitiu a construção de uma classificação para os *sites* estudados. Através disso foi observado que portais de estados mais ricos, com um maior PIB, possuem um maior número de vulnerabilidades, onde o portal com a pior avaliação foi do governo do estado do Rio de Janeiro.

Em um trabalho futuro, espera-se aumentar o número de *sites* a serem analisados incluindo também *sites* de prefeituras de todos os estados. Uma nova perspectiva de análise pode ser adotada levando em consideração os falsos positivos bem como o uso de novas ferramentas. Seria importante elaborar uma outra perspectiva que possibilitasse a elaboração de uma hipótese baseada no nível tecnológico de cada estado e do número de serviços oferecidos pelo portal, visto que quanto maior o número de funcionalidades em um sistema, maior é o número de pontos de ataque.

Referências

- ABNT. ABNT NBR ISO/IEC 17799: Tecnologia da informação — Técnicas de segurança — Código de prática para a gestão da segurança da informação. *Tecnologia da informação - Técnicas de segurança - código de prática para a gestão da segurança da informação*, p. 120, 2005. Citado na página 14.
- ANTUNES, N.; VIEIRA, M. Assessing and Comparing Vulnerability Detection Tools for Web Services: Benchmarking Approach and Examples. *IEEE Transactions on Services Computing*, v. 8, n. 2, p. 269–283, 2015. ISSN 19391374. Citado 5 vezes nas páginas 19, 23, 24, 25 e 40.
- APARECIDO, C. A. G. M.; BELLEZI, M. A. Análise de Vulnerabilidades com OpenVAS e Nessus. p. 34–44, 2014. Citado na página 14.
- BAKER, S.; WATERMAN, S.; IVANOV, G. *Sob fogo cruzado: Infraestrutura crítica na era da guerra cibernética*. [S.l.]: McAfee, 2013. Citado 3 vezes nas páginas 5, 10 e 11.
- BEAVER, K. *Hacking for Dummies*. [s.n.], 2007. v. 465. 1–74 p. ISSN 14764687. ISBN 076455784X. Disponível em: <<http://www.ncbi.nlm.nih.gov/pubmed/20505669>>. Citado na página 37.
- CERT.BR. *Estatísticas do CERT.br – Incidentes*. 2016. Disponível em: <<https://www.cert.br/stats/incidentes/>>. Citado 3 vezes nas páginas 5, 10 e 11.
- DOUPÉ, A.; COVA, M.; VIGNA, G. Why Johnny can't pentest: An analysis of black-box web vulnerability scanners. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 6201 LNCS, p. 111–131, 2010. ISSN 03029743. Citado 6 vezes nas páginas 12, 19, 21, 25, 26 e 40.
- EVANS, D.; YEN, D. C. E-Government: Evolving relationship of citizens and government, domestic, and international development. *Government Information Quarterly*, v. 23, n. 2, p. 207–235, 2006. ISSN 0740624X. Citado na página 18.
- FONSECA, J.; VIEIRA, M.; MADEIRA, H. Testing and comparing web vulnerability scanning tools for SQL injection and XSS attacks. *Proceedings - 13th Pacific Rim International Symposium on Dependable Computing, PRDC 2007*, p. 365–372, 2007. Citado 5 vezes nas páginas 12, 19, 23, 25 e 40.
- Gabriella Fonseca Ribeiro. *.Net - Como evitar ataques por SQL Injection - Eu Faço Programas*. 2011. 1 p. Disponível em: <<http://eufacoprogramas.com/net-como-evitar-sql-injection/>>. Citado 2 vezes nas páginas 5 e 16.
- HKSAR, T. G. o. t. H. K. S. A. R. Government of the. An Overview Of Vulnerability Scanners. *The Government of the Hong Kong Special Administrative Region*, n. February, p. 16, 2008. Disponível em: <<http://www.infosec.gov.hk/english/technical/files/vulnerability.pdf>>. Citado 3 vezes nas páginas 5, 17 e 18.

HOLM, H. et al. A quantitative evaluation of vulnerability scanning. *Information Management & Computer Security*, v. 19, n. 4, p. 231–247, 2011. ISSN 0968-5227. Citado na página 12.

MONTEVERDE, W. A.; CAMPIOLO, R. Estudo e Análise de Vulnerabilidades Web. p. 415–423, 2014. Citado 6 vezes nas páginas 12, 19, 20, 21, 25 e 40.

NAKAMURA, E. T.; GEUS, P. L. de. *Segurança de redes em ambientes cooperativos*. [S.l.]: Novatec Editora, 2007. Citado na página 10.

Network Sorcery, I. *Well known SCTP, TCP and UDP ports, 0 through 999*. 2012. Disponível em: <<http://www.networksorcery.com/enp/protocol/ip/ports00000.htm>>. Citado na página 44.

OWASP. *Testing for Cross site scripting*. 2013. 1 p. Disponível em: <https://www.owasp.org/index.php/Testing_for_Cross_site_scripting>. Citado 2 vezes nas páginas 5 e 17.

OWASP. *Vulnerability Scanning Tools - OWASP*. 2017. Disponível em: <https://www.owasp.org/index.php/Category:Vulnerability_Scanning_Tools>. Citado na página 18.

OWASP, P. et al. OWASP Top 10 - 2013: Os dez riscos de segurança mais críticos em aplicações web. *OWASP Top 10*, p. 23, 2013. Citado 11 vezes nas páginas 6, 14, 27, 28, 29, 30, 33, 35, 36, 37 e 40.

ROCHA, D.; KREUTZ, D.; TURCHETTI, R. Uma ferramenta livre e extensível para detecção de vulnerabilidades em sistemas Web. *10th International Symposium on Autonomous Decentralized Systems (ISADS)*, n. July, p. 747–752, 2011. ISSN 2163-1484. Citado 6 vezes nas páginas 12, 19, 21, 22, 25 e 40.

SANTOS, M. C. P. et al. Segurança em Sítios de Governo Eletrônico Brasileiros : um estudo de caso. *ResearchGate*, p. 1–4, 2014. Citado 5 vezes nas páginas 12, 19, 20, 25 e 40.

STALLINGS, W. *Criptografia e segurança de redes: princípios e práticas*. 6. ed. São Paulo: Pearson Education do Brasil, 2015. 578 p. ISBN 9788543014500. Citado na página 14.

VIEIRA, M.; ANTUNES, N.; MADEIRA, H. Using web security scanners to detect vulnerabilities in web services. *Proceedings of the International Conference on Dependable Systems and Networks*, p. 566–571, 2009. Citado 4 vezes nas páginas 19, 22, 25 e 40.

WIKIPEDIA. *List of TCP and UDP port numbers — Wikipedia, The Free Encyclopedia*. 2017. [Online; accessed 11-November-2017]. Disponível em: <https://en.wikipedia.org/w/index.php?title=List_of_TCP_and_UDP_port_numbers&oldid=809657024>. Citado na página 44.

WIKIPÉDIA. *Lista de unidades federativas do Brasil por PIB — Wikipédia, a enciclopédia livre*. 2017. [Online; accessed 13-outubro-2017]. Disponível em: <https://pt.wikipedia.org/w/index.php?title=Lista_de_unidades_federativas_do_Brasil_por_PIB&oldid=50132158>. Citado 3 vezes nas páginas 5, 38 e 46.

Apêndices

APÊNDICE A – Portas e suas respectivas funções

As portas encontradas possuem as seguintes funções, segundo ([WIKIPEDIA, 2017](#)),([Network Sorcery, 2012](#)):

- 21 - *File Transport Protocol(FTP)*
- 22 - *Secure Shell(SSH)*
- 55 - *ISI Graphics Language*
- 68 - *Bootstrap Protocol (BOOTP), Host Configuration Protocol (DHCP)*
- 80 - *Hypertext Transfer Protocol (HTTP)*
- 111 - *Open Network Computing Remote Procedure Call (ONC RPC)*
- 200 - *IBM System Resource Controller*
- 443 - *Hyper Text Transfer Protocol SecureHTTPS*
- 2121 - *servexec/TCP, xinuexpansion1/UDP*
- 3389 - *Microsoft Terminal Server (RDP)/Windows Based Terminal (WBT)*
- 5060 - *SIP, Session Initiation Protocol*
- 8008 - *Porta alternativa ao HTTP. Vista como 80 e 8080*
- 8010 - Não contam especificação.
- 8080 - *Porta alternativa ao HTTP. Vista como 80 e 8008*
- 9080 - *Groove Collaboration software GLRPC/WebSphere Application Server HTTP Transport/Remote Potato by FatAttitude, Windows Media Center addon/ServerWMC, Windows Media Center addon*
- 9090 - *WebSM/Openfire Administration Console/SqueezeCenter control (CLI)/ Cherokee Admin Panel*

Anexos

ANEXO A – Produto Interno Bruto(PIB) do estados brasileiros







Posição		Unidade federativa	PIB (R\$ 1.000)	Participação relativa em 2014 em % do total	Participação relativa em 2015 em % do total
Em 2015	Comparada a 2014				
1	— (0)	 São Paulo	▲ 1.939.890.000	— 32,2	▲ 32,4
2	— (0)	 Rio de Janeiro	▲ 659.137.000	▼ 11,6	▼ 11,0
3	— (0)	 Minas Gerais	▲ 519.326.000	▼ 8,9	▼ 8,7
4	— (0)	 Rio Grande do Sul	▲ 381.985.000	— 6,2	▲ 6,4
5	— (0)	 Paraná	▲ 376.960.000	▼ 6,0	▲ 6,3
6	— (0)	 Santa Catarina	▲ 249.073.000	▲ 4,2	— 4,2
7	— (0)	 Bahia	▲ 245.025.000	▲ 3,9	▲ 4,1
8	— (0)	 Distrito Federal	▲ 215.613.000	▲ 3,4	▲ 3,6
9	— (0)	 Goiás	▲ 173.632.000	▲ 2,9	— 2,9
10	— (0)	 Pernambuco	▲ 156.955.000	▲ 2,7	▼ 2,6
11	▲ (2)	 Pará	▲ 130.883.000	▼ 2,2	— 2,2
12	▲ (1)	 Ceará	▲ 130.621.000	▲ 2,2	— 2,2
13	▼ (2)	 Espírito Santo	▲ 120.363.000	— 2,2	▼ 2,0
14	— (0)	 Mato Grosso	▲ 107.418.000	▲ 1,8	— 1,8
15	— (0)	 Amazonas	▲ 86.560.000	▼ 1,5	▼ 1,4
16	— (0)	 Mato Grosso do Sul	▲ 83.082.000	▲ 1,4	— 1,4
17	— (0)	 Maranhão	▲ 78.475.000	— 1,3	— 1,3
18	— (0)	 Rio Grande do Norte	▲ 57.250.000	▼ 0,9	▲ 1,0
19	— (0)	 Paraíba	▲ 56.140.000	— 0,9	— 0,9
20	— (0)	 Alagoas	▲ 46.364.000	— 0,7	▲ 0,8
21	— (0)	 Piauí	▲ 39.148.000	▲ 0,7	— 0,7
22	— (0)	 Sergipe	▲ 38.554.000	▼ 0,6	— 0,6
23	— (0)	 Rondônia	▲ 36.563.000	— 0,6	— 0,6
24	— (0)	 Tocantins	▲ 28.930.000	▲ 0,5	— 0,5
25	— (0)	 Acre	▲ 13.622.000	— 0,2	— 0,2
26	— (0)	 Amapá	▲ 13.861.000	— 0,2	— 0,2
27	— (0)	 Roraima	▲ 10.354.000	— 0,2	— 0,2
Total			5.995.787.000	100	100

Figura 11 – Tabela de classificação do PIB. Extraído do artigo de ([WIKIPÉDIA, 2017](#)).