INSTITUTO FEDERAL DE ALAGOAS - CAMPUS ARAPIRACA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

EDUARDO VÍTOR VIEIRA TORRES

Análise de Vulnerabilidades dos Portais Web das Câmaras Municipais Alagoanas

Alagoas
Dezembro de 2022

EDUARDO VÍTOR VIEIRA TORRES

Análise de Vulnerabilidades dos Portais Web das Câmaras Municipais Alagoanas

Trabalho de conclusão de curso apresentado ao Curso de Bacharelado em Sistemas de Informação do Instituto Federal de Alagoas Campus Arapiraca, como requisito parcial para a obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Daniel Lacet de Faria Fireman
Instituto Federal de Alagoas Campus Arapiraca

Alagoas
Dezembro de 2022

Dados Internacionais de Catalogação na Publicação
Instituto Federal de Alagoas
***Campus* Arapiraca**

T693a

Torres, Eduardo Vítor Vieira.
Análise de vulnerabilidades dos portais web das câmaras municipais alagoanas / Eduardo Vítor Vieira Torres. – 2022.
1 PDF: il., color. (1 aquivo : 746 kB).

Arquivo digital no formato PDF do trabalho acadêmico com 40 folhas.

Orientação: Prof. Dr. Daniel Lacet de Faria Fireman.
Trabalho de Conclusão de Curso (Graduação, Bacharelado em Sistemas de Informação) – Instituto Federal de Alagoas, *Campus* Arapiraca, Arapiraca, 2022.

1. Segurança da informação. 2. Vulnerabilidade. 3. Portais eletrônicos governamentais. I. Título.

CDD: 005.8

Luciete B. da Silva
Bibliotecária – CRB-4/1739

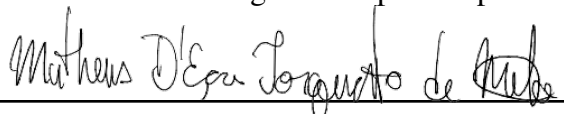
EDUARDO VÍTOR VIEIRA TORRES

Análise de Vulnerabilidades dos Portais Web das Câmaras Municipais Alagoanas

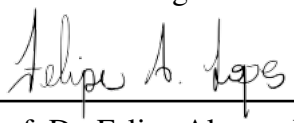
Trabalho de conclusão de curso apresentado ao Curso de Bacharelado em Sistemas de Informação do Instituto Federal de Alagoas Campus Arapiraca, como requisito parcial para a obtenção do título de Bacharel em Sistemas de Informação.

Trabalho aprovado. Alagoas, 27 de dezembro de 2022:

Prof. Dr. Daniel Lacet de Faria Fireman
Instituto Federal de Alagoas Campus Arapiraca



Prof. Me. Matheus D'Eca Torquato De Melo
Instituto Federal de Alagoas Campus Arapiraca



Prof. Dr. Felipe Alencar Lopes
Instituto Federal de Alagoas Campus Arapiraca

Alagoas
Dezembro de 2022

Agradecimentos

Agradeço a Deus, autor e sustentador da vida, por ser minha força, refúgio e fortaleza e ter colocado no meu caminho pessoas maravilhosas que puderam me guiar até hoje.

Agradeço aos meus pais e amigos, por estarem ao meu lado durante toda essa caminhada. Agradeço especialmente ao professor Daniel Fireman, meu orientador e amigo, pela paciência, pelo seu tempo e pelo apoio dado a mim e a esta ideia de TCC. Agradeço também ao meu amigo, Gabriel Tavares por ter disponibilizado códigos de apoio e um modelo de fichamento usado para a escrita da seção de Trabalhos Correlatos neste trabalho.

Por fim, agradeço a todos os professores do IFAL Campus Arapiraca pelos ensinamentos, pela amizade e pela dedicação a educação neste país. Área que vem sendo tão desvalorizada nos últimos anos.

“ Pra quem tem pensamento forte, o impossível é só questão de opinião ” - Charlie Brown Jr

Resumo

Em um mundo cada vez mais informatizado e com acesso à Internet, é imprescindível a existência de portais de governos eletrônicos, os quais disponibilizam serviços e informações, que facilitam o acesso à consulta dos cidadãos no que se refere as suas demandas. Nesse sentido, existem portais de câmaras municipais que fornecem informações sobre vereadores em exercício, notícias sobre a câmara, além de conterem o portal da transparência, que possui informações sobre ações governamentais, execução orçamentária e financeira (receitas e despesas), dentre outras. Como qualquer sistema que está conectado à Internet, tais portais podem possuir vulnerabilidades de segurança que colocam em risco os serviços oferecidos pelos mesmos, bem como os dados dos usuários que os utilizam. O presente trabalho pretende verificar eventuais vulnerabilidades de segurança existentes em portais das câmaras municipais do estado de Alagoas de modo que seja possível analisar as vulnerabilidades conforme a classificação OWASP Top 10 de 2021, bem como avaliar vulnerabilidades encontradas segundo o modelo Interlegis e segundo o PIB das cidades. Para coleta de vulnerabilidades, foi utilizado o *scanner* Wapiti. Os resultados obtidos exibiram um total de 667 vulnerabilidades. Ademais, 10% dos portais apresentaram vulnerabilidades críticas. Como exemplos de tipos de vulnerabilidades encontradas: injeção - que permite consultas mal-intencionadas a bancos de dados e configuração incorreta de segurança - que pode acarretar roubo de sessões de usuário.

Palavras-chave: Segurança da informação, Vulnerabilidades, Portal de câmara eletrônico

Abstract

In an increasingly computerized world with access to the Internet, the existence of electronic government portals is essential, which provide services and information, which facilitate access to consultation by citizens with regard to their demands. In that regard, there are city council portals that provide information on acting councilors, news about the councils, in addition to containing the transparency portal, which has information on government actions, budgetary and financial execution (revenues and expenses), among others. Just like any system that is connected to the Internet, such portals may have security vulnerabilities that put the services offered by them at risk, as well as the data of the users who use them. The present work has the general objective of verifying any existing security vulnerabilities in portals of the municipal councils of the state of Alagoas so that it is possible to analyze the vulnerabilities according to the OWASP Top 10 classification of 2021, as well as to evaluate vulnerabilities found according to the Interlegis model and according to the GDP of the cities. To collect vulnerabilities, the Wapiti *scanner* was used. The results obtained showed that the total number of vulnerabilities found was 667. Furthermore, 10% of portals had critical vulnerabilities. As examples of the types of vulnerabilities found: Injection - which allows malicious queries to databases and Incorrect Security Configuration - which can lead to theft of user sessions.

Keywords: Cybersecurity. Vulnerabilities. City councils' portals

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ABNT	Associação Brasileira de Normas Técnicas
AWS	*Amazon Web Services*
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	*Content Management System*
CSV	*Comma-separated values*
CTIR Gov	Centro de Prevenção, Tratamento e Resposta a Incidentes Cibernéticos de Governo
EC2	*Amazon Elastic Compute Cloud*
HTML	*HyperText Markup Language*
IAC	*Infrastructure as Code*
IBGE	Instituto Brasileiro de Geografia e Estatística
ISO	*International Organization for Standardization*
IP	*Internet Protocol*
JSON	*JavaScript Object Notation*
LAI	Lei de Acesso à Informação
LFI	*Local File Inclusion*
LOA	Lei Orçamentária Anual
NBR	Norma Brasileira
OWASP	*Open Web Application Security Project*
PIB	Produto Interno Bruto
RFI	*Remote File Inclusion*
SQL	*Structured Query Language*
SQLi	*Structured Query Language Injection*
SSL	*Secure Sockets Layer*
SSRF	*Server-side request forgery*
URL	*Uniform Resource Locator*
XML	*Extensible Markup Language*
XSS	*Reflected Cross-site Scripting*

Sumário

1 Introdução

Instituições públicas e privadas vem adotando as mais variadas soluções tecnológicas baseadas na web, como, por exemplo, sites e aplicativos, para aprimorar seus processos e facilitar a execução de suas atividades-fim. Concomitantemente a isso, pessoas mal-intencionadas surgem cometendo crimes cibernéticos para comprometer os sistemas de diversas organizações e impedir o progresso das mesmas, como relata o trabalho de (WENDT; JORGE, 2013).

O CERT (Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil) aponta que no ano de 2020 e 2019 houve uma média de 770.203 incidentes de segurança. Uma análise mais profunda desses dados revela que no ano de 2020 aconteceram 1822 ataques por dia, enquanto em 2019 ocorreram aproximadamente 2398 ataques por dia. A Figura 1 ilustra o número de incidentes de segurança reportados entre 1999 e 2020 segundo o CERT.

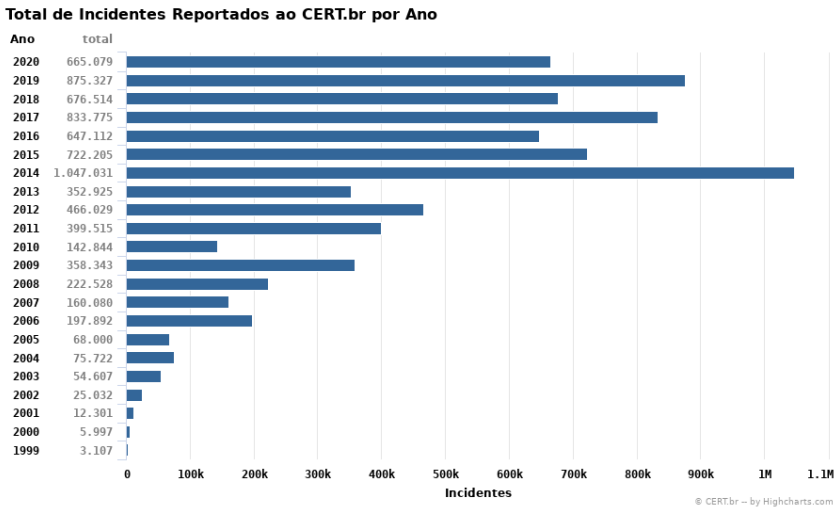


Figura 1 – Incidentes de segurança reportados ao CERT por ano. Extraído do portal (CERT.BR, 2020)

Em relação a incidentes em sites de governos eletrônicos, o CTIR Gov (Centro de Prevenção, Tratamento e Resposta à Incidentes Cibernéticos de Governo) informa que, entre os anos de 2018 e 2022, houve uma média anual de 6765 incidentes, o que corresponde a uma média de 18 ataques a portais de governo por dia. A Figura 2 mostra incidentes, notificações e vulnerabilidades coletados pelo CTIR entre os anos de 2018 a 2022.

Com relação a prejuízos causados por esses incidentes, segundo (SENADO, 2022) entre 2017 e 2018, os prejuízos advindos dos ataques cibernéticos no Brasil ultrapassaram US$ 20 bilhões, o que ultrapassa R$ 80 bilhões. Tais dados demonstram o quão rotineiros são os ataques hackers no Brasil contra instituições públicas e empresas privadas, que afetam diretamente os dados confidenciais dos indivíduos, como também as operações dessas instituições.

Uma vulnerabilidade segundo (AGRA; BARBOZA, 2019) pode ser definida como uma brecha que atacantes podem utilizar para burlar o funcionamento correto de um sistema, ou ainda,

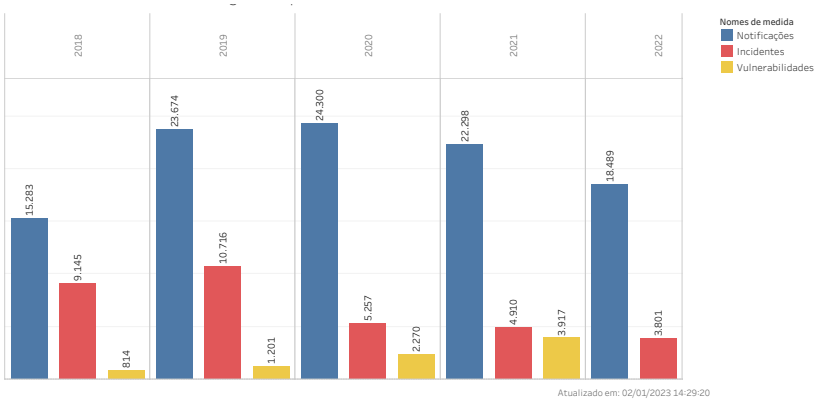


Figura 2 – Incidentes de segurança coletados pelo CTIR por ano. Extraído do portal (CTIR, 2022)

roubar informações do mesmo. De forma complementar, (BRANQUINHO; BRANQUINHO, 2021) enumera alguns exemplos de vulnerabilidade como: configurações malfeitas, portas de comunicação abertas e senhas fracas.

Alagoas é um estado com população de 3 milhões de habitantes segundo o último censo do IBGE. O estado possui um total de 102 municípios com mais de 1000 vereadores em atividade nas câmaras municipais. Os portais eletrônicos das câmaras disponibilizam informações sobre os vereadores em exercício e notícias sobre a câmara, além de conter o portal da transparência, que possui informações acerca das ações governamentais, execução orçamentária e financeira (receitas e despesas), movimento extra orçamentário, dentre outras informações de interesse do cidadão.

O governo federal disponibiliza um modelo de portal gratuito e de código fonte aberto chamado de Portal Modelo Interlegis ou Modelo Interlegis. Ele vem pronto para ser usado pelas câmaras municipais e assembleias legislativas e foi concebido com foco em usabilidade, acessibilidade e segurança. Devido a essas vantagens e as restrições orçamentárias, seria esperado uma adoção massiva entre as câmaras municipais.

Atualmente, não se sabe se existem vulnerabilidades de segurança em portais eletrônicos das câmaras municipais alagoanas, seja ele do tipo Interlegis ou não. Portanto, é imperativo ter uma análise de segurança destes portais, já que os cidadãos têm a possibilidade de acessar esses sites e é importante que os sites sejam seguros para não comprometerem os dispositivos eletrônicos usados no acesso, as informações dos cidadãos e informações relacionadas às câmaras em si.

Ademais, a integridade das informações das câmaras são de extrema importância para garantir que a Lei de Acesso à Informação (LAI) esteja sendo cumprida pelo município. Se houver vulnerabilidades de segurança nesses portais de câmaras, atacantes podem adulterar informações anteriormente íntegras e fazer com que a LAI seja descumprida, prejudicando o acesso a informações legítimas por parte dos cidadãos neste tipo de site.

Finalmente, o orçamento das câmaras municipais é estabelecido por repasses financeiros mensais do executivo a outras esferas de poder (Duodécimo). O Duodécimo é definido pelo

orçamento de cada município; tal orçamento é concebido a partir da LOA (Lei Orçamentária Anual) que antecipa as possíveis receitas e despesas do município para o ano em questão. Sendo assim, um importante indicador econômico levado em conta para o cálculo das receitas do município, e por consequência, das câmaras municipais, é o PIB (Produto Interno Bruto).

O PIB é a soma de todos os bens e serviços finais produzidos por uma determinada região geográfica geralmente em um ano. Ele mede a atividade econômica a fim de verificar o resultado do crescimento econômico do lugar avaliado. Com isso, como uma medida indireta de relação com o orçamento das câmaras, é pertinente verificar uma possível correlação entre o PIB das cidades alagoanas com o número de vulnerabilidades nos portais de câmara, isto é, cidades com maiores PIB talvez possuam um número menor de vulnerabilidades e vice-versa.

O presente trabalho tem como objetivo geral verificar eventuais vulnerabilidades de segurança existentes em portais das câmaras municipais do estado de Alagoas. Tem-se como objetivos específicos:

1. Analisar as vulnerabilidades conforme a classificação OWASP de criticidade;
2. Discutir possíveis soluções e medidas de prevenção para as vulnerabilidades encontradas;
3. Avaliar vulnerabilidades em portais de câmara que utilizam o Portal Modelo Interlegis do Governo Federal;
4. Avaliar vulnerabilidades conforme o PIB das cidades dos portais (correlação com o orçamento das câmaras de forma indireta)

Como algumas contribuições desse estudo, tem-se que o total de vulnerabilidades encontrado considerando todos os portais foi de 667. Os municípios que tiveram o maior número de vulnerabilidades (26) foram Maragogi e Penedo. Já os que tiveram o menor número de vulnerabilidades (1) foram Belém, Belo Monte, Cajueiro, Carneiros, Colônia Leopoldina, Girau do Ponciano, Maribondo, Satuba e Senador Rui Palmeira. Ademais, portais que utilizam o Portal Modelo Interlegis tiveram um número de vulnerabilidades médio maior que portais que não utilizam. Finalmente, os portais de câmara das dez cidades com maior PIB de Alagoas apresentaram um número de vulnerabilidades médio maior comparado com os outros portais.

O presente estudo encontra-se organizado em capítulos, onde se tem: 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, composto pela metodologia, vulnerabilidades encontradas e como corrigi-las, além das avaliações de vulnerabilidades referentes ao modelo Interlegis e ao PIB das cidades. Por fim, no capítulo 4 desenvolve-se a conclusão, a qual reúne as considerações finais obtidas na efetivação deste trabalho

2 Fundamentação Teórica

A medida em que os sistemas evoluem, ferramentas capazes de descobrir e explorar vulnerabilidades de segurança crescem na mesma proporção. Nesse sentido, a prática de atacar sistemas computacionais buscando identificar e explorar vulnerabilidades (*hacking*) pode ser considerada de fácil aprendizado (UTICA, 2020).

Dessa forma, pessoas especialistas em hacking podem usar seus conhecimentos para atacar sistemas ou protegê-los. Instituição públicas e privadas sofrem grandes prejuízos financeiros e tecnológicos quando tais conhecimentos são usados de forma antiética. Em 2017, por exemplo, um ransomware denominado *WannaCry* afetou diversos sistemas computacionais de várias empresas ao redor do mundo, ocasionando prejuízos milionários (MIMECAST, 2021).

Com relação à organização do restante desse capítulo, na seção 2.1 serão abordados alguns conceitos básicos de Segurança de Informação, onde, de modo breve, serão definidos aspectos como integridade, confidencialidade e disponibilidade. Na seção 2.2, será definido o conceito de vulnerabilidade, bem como as principais vulnerabilidades de segurança segundo a OWASP, organização sem fins lucrativos que tem como objetivo auxiliar desenvolvedores de sites e especialistas de segurança na proteção de aplicações web contra ciberataques. A seção 2.3 define scanners de vulnerabilidades. A seção 2.4 aborda conceitos relacionados a portais eletrônicos e câmaras municipais. Por fim, a seção 2.5 trata de trabalhos correlatos a este.

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 (KIM; SOLOMON, 2014), as características a serem preservadas podem ser definidas como:

- Confidencialidade: manter restrições sobre a divulgação e acesso de informações, com 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 cadastrados podem ser vazados, gerando 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 dados tenham sido alterados sem que se tenha mecanismos de backup, nesse caso o princípio da integridade das informações foi comprometido.

- Disponibilidade: garantir o acesso rápido e confiável a informação. Um ataque pode ser realizado contra algum site com o objetivo de torná-lo indisponível. Caso isso aconteça e o ataque seja bem-sucedido, a disponibilidade estará comprometida.

2.2 Vulnerabilidades

Segundo (MARTINELO; 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, 2021), sendo organizadas em uma lista com as dez vulnerabilidades mais críticas (OWASP Top 10). A OWASP ou *The Open Web Application Security Project* é uma fundação que tem como objetivo melhorar a segurança de softwares permitindo que indivíduos ou empresas tomem decisões embasadas. A classificação da OWASP consiste nos seguintes grupos:

1. Quebra de Controle de Acesso: ocorre quando o controle de acesso (permissões) de uma determinada aplicação web é frágil. O que permite que atacantes consigam acesso não autorizado a alguma parte do código ou informação privilegiada por meio de senhas ou *tokens* não-validados. Por exemplo, um atacante poderia ter controle total da conta de um usuário em um serviço de *streamming* que não possua um controle de acesso adequado.

2. Falhas Criptográficas: ocorre quando dados sensíveis como senhas, dados de cartão de crédito e registros médicos são armazenados sem criptografia ou com alguma forma de criptografia insegura. Um exemplo de ataque é um banco de dados que usa informações de forma não criptografada, uma falha de envio de arquivos ao banco poderia fazer com que a senha do banco fosse capturada e com isso o atacante poderia descobrir todas as senhas dessa base de dados.

3. Injeção: ocorre quando um atacante injeta parâmetros maliciosos em consultas a bases de dados. Tais parâmetros são capazes de modificar uma consulta anteriormente restritiva para conseguir acesso a informação indevidas ou fazer operações não autorizadas no banco de dados como modificar ou deletar dados. Um exemplo de ataque é um atacante modificar uma consulta de forma a obter todos os dados de uma tabela, o que não seria possível sem a injeção.

4. Design Inseguro: ocorre quando há más práticas de segurança implementadas em um software, com atenção voltada para a falta de modelagem de ameaças, não uso de padrões de design de segurança e arquiteturas de referência. Um exemplo de má prática se refere ao uso de perguntas e respostas na redefinição de senhas de usuários em uma aplicação.

5. Configuração Incorreta de Segurança: ocorre quando há: envio de requisições com *headers* inseguros, softwares desnecessários dentro de um servidor web que possuem vulnerabilidades conhecidas, uso de senhas e usuários padrão em aplicações, etc. Um exemplo de ataque seria uma escalada de privilégios por meio de aplicações vulneráveis dentro de um servidor web com credenciais padrão.

6. Componentes Vulneráveis e Desatualizados: ocorre quando não se tem conhecimento sobre versões de software que a aplicação usa, esses softwares podem estar vulneráveis a ataques se não há uma política de testes periódicos de vulnerabilidades ou acompanhamento de boletins de segurança. Um exemplo de ataque seria explorar uma aplicação a partir de uma vulnerabilidade conhecida e conseguir acesso total sobre ela.

7. Falhas de Autenticação e Identificação: ocorre quando é possível automatizar testes de usuários e senhas por meio de força bruta e listas de senhas conhecidas. Além disso, também ocorre quando se tem uso de senhas padrão na aplicação, quando uma sessão de usuário pode ser facilmente roubada, etc. Um exemplo de ataque é usar uma lista de senhas conhecidas para conseguir acesso a uma aplicação.

8. Falhas de integridade de software e dados: ocorre quando uma atualização de software por meio de um arquivo não possui nenhuma verificação de integridade ou quando dados serializados não assinados ou não encriptados são enviados para fontes duvidosas como bibliotecas e componentes não confiáveis. Por exemplo, um atacante poderia modificar uma imagem ISO de um Sistema Operacional inserindo código malicioso e distribuí-la em algum site.

9. Falhas de Monitoramento e Logs de segurança: ocorre quando não há um monitoramento adequado de logs de uma aplicação, quando os logs revelam informaçãos sensível ou informações insuficientes, quando os logs da aplicação não são capazes de detectar tentativas de login e requisições em excesso ou quando a detecção do uso de ferramentas de testes de penetração não existe.

10. Falsificação de requisições para o lado servidor (SSRF): Uso de requisições maliciosas para conseguir alguma informação indevida do lado servidor. Um exemplo de ataque é conseguir acessar metadados de serviços de armazenamento na nuvem ou conseguir comprometer serviços internos usando execução de código remoto ou ataques de negação de serviço.

2.3 Scanners de vulnerabilidades web

Segundo (LIS, 2019), *scanners* de vulnerabilidades web são ferramentas que testam aplicações web a procura de vulnerabilidades existentes e retornam as vulnerabilidades encontradas em formato de relatório para o usuário. Os scanners podem ser de dois tipos: *black-box*

e *white-box*. O primeiro realiza testes mandando requisições para interagir com a aplicação, nesse caso o código fonte da aplicação não está disponível. Já o segundo foca em encontrar vulnerabilidades de uma aplicação web por meio do seu código fonte.

Um *scanner black-box* tem uma arquitetura que se baseia na ideia de que a tarefa de testar uma aplicação web pode ser dividida em várias subtarefas. Dessa forma, a arquitetura de tais ferramentas é modular, permitindo fácil manutenção e extensibilidade do código fonte. A Figura 3 demonstra a estrutura de um *scanner* de vulnerabilidades *black-box* em 4 componentes:

1. Núcleo (*Core*): Componente responsável por orquestrar a execução dos outros componentes e monitorar o scan.

2. Coletor (*Crawler*): Componente responsável por fornecer os *endpoints* da aplicação a partir de uma URL base que serão usados na fase de auditoria. Ele funciona a partir de links em páginas conhecidas, sendo eles estáticos ou gerados de forma dinâmica.

3. Auditor (*Auditor*): Componente responsável por testar as URLs encontradas pelo coletor em busca de vulnerabilidades. Para isso, ele utiliza requisições com parâmetros aleatórios, inválidos ou inesperados e analisa as respostas recebidas da aplicação. Tal componente pode ser divido em módulos específicos de cada vulnerabilidade diferente.

4. Gerador de relatório (*Report generator*): Componente responsável por gerar relatórios em diferentes formatos e com informações precisas sobre as vulnerabilidades encontradas.

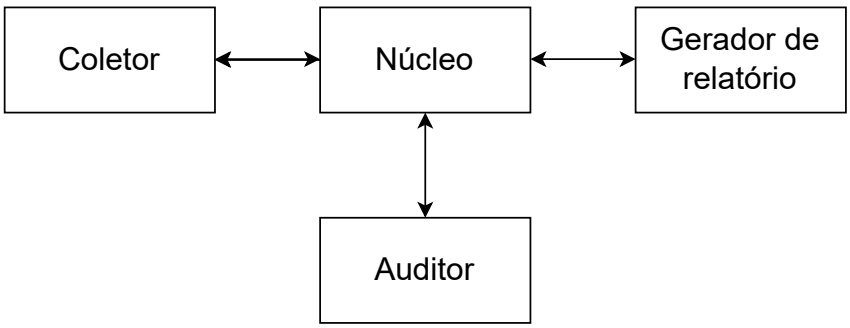


Figura 3 – Arquitetura de um scanner de vulnerabilidades black-box, adaptado de (LIS, 2019)

A ferramenta usada nesse estudo é o (WAPITI, 2022), um *scanner* de vulnerabilidades de aplicações web escrito em (PYTHON, 2022) que roda a partir de um terminal de comandos. A escolha da ferramenta está justificada nas seções Seção 2.5 e Seção 3.1. A execução do Wapiti têm duas partes: coleta de páginas do website e execução da auditoria de vulnerabilidades, ou seja, a execução dos módulos de ataque. Alguns exemplos de módulos são: Injeção SQL, XSS e execução de comandos. A ferramenta possui diversos parâmetros que se adequam a diversos casos de uso, como tempo máximo de *scan* de páginas, uso de *cookies* personalizados, uso de *proxys* e verificação ou não de SSL. Por fim, o relatório gerado pela ferramenta é detalhado e

pode ser exportado em diferentes formatos como: CSV, JSON, HTML e XML. A Figura 4 ilustra a ferramenta no terminal de comandos do Linux.

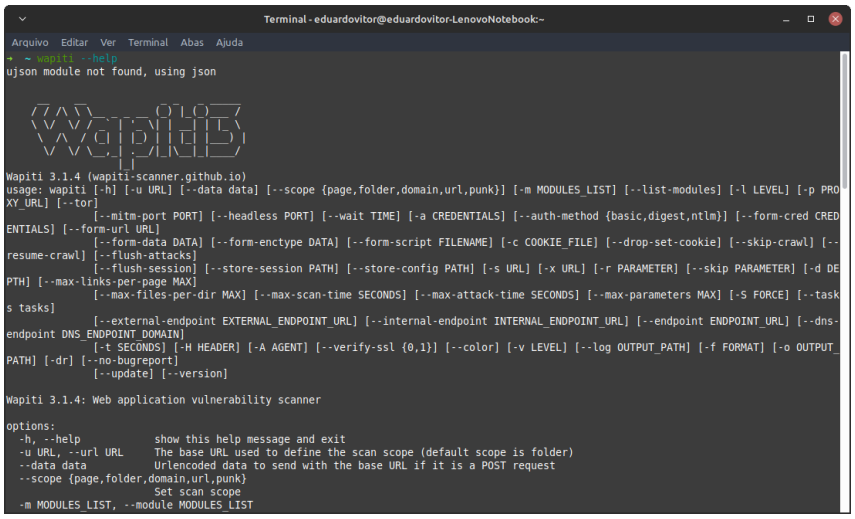


Figura 4 – Ferramenta Wapiti. Imagem produzida pelo autor.

2.4 Portal eletrônico de câmaras municipais

(PINHO, 2008) considera que o governo eletrônico, representado pela informatização de suas atividades internas e pela comunicação com o público externo, cidadãos, fornecedores e empresas têm se efetivado por meio da construção de portais governamentais. Por meio deles, os governos mostram sua identidade, seus propósitos, suas realizações e disponibilizam serviços e informações para os cidadãos.

Atualmente, legislativos de todas as esferas, inclusive da esfera municipal, que é o foco deste trabalho, apresentam suas páginas eletrônicas, disponibilizando informações, tais como notícias diárias sobre as ações parlamentares, ordem do dia, relatórios semestrais e anuais das ações desenvolvidas pelos parlamentares e pelas comissões, execução orçamentária, entre outros assuntos.

O governo federal disponibiliza um modelo de portal gratuito, chamado de Portal Modelo Interlegis ou Modelo Interlegis, ele é rápido e pronto para uso das Câmaras Municipais e Assembleias Legislativas. O Portal Modelo é uma customização do CMS Plone que fornece um portal completo para uma Casa Legislativa, com arquitetura de informação padronizada e pronto para receber conteúdos. Ele inclui ainda várias ferramentas e funcionalidades que facilitam o trabalho de uma instituição pública e a tornam aderente a padrões nacionais e internacionais e às leis brasileiras, como por exemplo a Lei de Acesso à Informação. A Figura 5 mostra o site do município de Pilar que utiliza o Portal Modelo Interlegis.

Figura 5 – Exemplo de Portal Modelo Interlegis do município de Pilar. Produzido pelo autor.

2.5 Trabalhos correlatos

2.5.1 Avaliação de *scanners* de vulnerabilidades

(ESCUDERO; ANDRADE; TERADA, 2020) realiza um estudo comparativo de *scanners* de vulnerabilidade gratuitos de código aberto sob a perspectiva de análise do módulo rastreador (*crawling*) dos *scanners*. Tal estudo avaliou 11 ferramentas, são elas: (W3AF, 2022), (PAROS, 2022), (WAPITI, 2022), (VEGA, 2022), (SKIPFISH, 2022) e (NIKTO, 2022) em um ambiente vulnerável conhecido na literatura, o WackoPicko, presente no trabalho de (DOUPÉ; COVA; VIGNA, 2010). As métricas usadas foram: recursos indexados, recursos acessados, páginas vulneráveis exploradas, interação entre os módulos, geração de páginas infinitas, realização de upload durante o ataque e realização de ataque baseado em dicionário.

Como contribuições do trabalho de (ESCUDERO; ANDRADE; TERADA, 2020), exceto o (WAPITI, 2022), todos os scanners geraram mais de 50% de falsos negativos, considerando apenas as páginas com vulnerabilidades que não foram rastreadas por cada ferramenta. Desse modo, o (WAPITI, 2022) foi a ferramenta que alcançou mais páginas vulneráveis e o (VEGA, 2022) indexou mais recursos. Ademais, o estudo concluiu que os resultados dos *scanners* de vulnerabilidades não foram eficazes, logo faz-se necessário executar testes manuais de aplicações web que irão entrar em produção a fim de achar vulnerabilidades. Finalmente, o trabalho afirma que *crawlers* eficientes são fundamentais para que se tenha resultados mais confiáveis/eficazes em relação à detecção de vulnerabilidades.

De forma análoga ao trabalho de (ESCUDERO; ANDRADE; TERADA, 2020), (FERRÃO; SÁ; KREUTZ, 2018) apresenta uma avaliação dos *scanners* de vulnerabilidade web gratuitos verificando a eficácia de tais ferramentas em um ambiente de vulnerabilidades controlado. A metodologia de pesquisa foi dividida em quatro etapas: a primeira foi caracterizada pela escolha, estudo e instalação dos *scanners*; na segunda etapa foram escolhidas as 10 vulnerabilidades do cenário controlado; na terceira, foi criado um ambiente no sistema Kali Linux o qual reproduzia o cenário vulnerável; por fim, a quarta etapa foi a execução dos *scanners* de vulnerabilidade no ambiente controlado.

Os resultados do trabalho de (FERRÃO; SÁ; KREUTZ, 2018) mostraram que os *scanners* de vulnerabilidade conseguiram detectar 50% das dez categorias de vulnerabilidade do cenário controlado proposto. Além disso, os *scanners* tiveram proporções diferentes de detecções de vulnerabilidade. O que reforça a ideia de que tais ferramentas são imperfeitas e podem apresentar falsos-positivos e falsos-negativos.

Já o trabalho de (VIEIRA; ANTUNES; MADEIRA, 2009) avalia vulnerabilidades de segurança de 300 serviços web públicos usando *scanners* de vulnerabilidade comerciais e não comerciais. A metodologia proposta seguiu quatro etapas: seleção dos *scanners* de vulnerabilidade e de um grande conjunto de serviços web, execução dos *scanners* nos serviços web, uso de testes manuais para confirmação das vulnerabilidades achadas e análise dos resultados obtidos e lições apreendidas.

A conclusão (VIEIRA; ANTUNES; MADEIRA, 2009) apontou que um grande número de vulnerabilidades foi encontrado nos serviços web. Além disso, os *scanners* de vulnerabilidades exibiram resultados diferentes de vulnerabilidades encontradas. Adicionalmente, o número de falsos-positivos foi alto, o que, conforme os autores, reduz a confiança na precisão das vulnerabilidades encontradas.

Em outra vertente, o estudo de (FONSECA; VIEIRA; MADEIRA, 2007) propôs um *benchmark* de *scanners* de vulnerabilidades web. O método utilizado consiste na injeção de vulnerabilidades reais em uma aplicação web. Tais falhas foram injetadas usando um programa denominado G-SWFIT, que emula os mais frequentes tipos de falha de alto nível. Os autores selecionaram duas aplicações web para análise: *MyReference*, ferramenta para controle e uso de referências; *BookStore*, loja de livros gerada por uma ferramenta de criação fácil de sites.

Para a execução dos experimentos, (FONSECA; VIEIRA; MADEIRA, 2007) usou um algoritmo inicial que testa a aplicação a procura de vulnerabilidades antes de rodar os *scanners* com o objetivo de verificar se já haviam vulnerabilidades nela, tal procedimento é chamado de *Golden run*. Posteriormente é feito um backup da aplicação sem alterações e então 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 do trabalho de (FONSECA; VIEIRA; MADEIRA, 2007) mostraram que os *scanners* de vulnerabilidade produzem diferentes resultados e podem não detectar vulnerabilidades existentes. Assim, existiu uma alta taxa de falsos-positivos variando de 20% a 70% nos experimentos realizados.

Finalmente, os trabalhos de (MOHAMMED, 2016) e (IDRISSI et al., 2017) analisaram *scanners* de vulnerabilidade focando em alguns tipos específicos de vulnerabilidade como: SQLI, XSS, RFI e LFI. A metodologia de ambos é similar quanto ao detalhamento do desempenho dos *scanners* conforme as métricas de: *recall* (número de vulnerabilidades detectadas corretamente sobre o número real total de vulnerabilidades), *precision* (número de vulnerabilidades detectadas

corretamente sobre o número total de vulnerabilidades detectadas) e *F-measure* (cálculo da média harmônica de *precision* e *recall*).

A metodologia difere quanto aos *scanners* avaliados e quanto a forma de calcular o número de falsos-positivos e falsos-negativos existentes. (MOHAMMED, 2016) utiliza testes manuais de vulnerabilidade para comparação com os resultados dos *scanners*, enquanto (IDRISSI et al., 2017) usa uma aplicação vulnerável conhecida e verifica quais vulnerabilidades realmente existem. Ambos os trabalhos de (MOHAMMED, 2016) e (IDRISSI et al., 2017) concluiram que a eficiência de cada *scanner* é diferente para cada tipo de vulnerabilidade. Adicionalmente, um importante aspecto desses estudos é considerar o *F-measure* como medida máxima de desempenho dos scanners levando em consideração o número de falsos-positivos e falsos-negativos.

Os trabalhos citados acima foram importantes para o entendimento do processo de coleta de vulnerabilidades por meio dos *scanners* e das limitações desse tipo de ferramenta. Além disso, as conclusões do trabalho de (ESCUDERO; ANDRADE; TERADA, 2020), em especial, foram um dos motivos para a escolha do *scanner* de vulnerabilidades usado neste trabalho, o (WAPITI, 2022).

2.5.2 Análise de vulnerabilidades em portais de governo eletrônico

(REIS et al., 2018) foca em analisar vulnerabilidades do sistema de Internet Banking de 20 instituições financeiras (IFs) tentando entender se há relação entre o patrimônio líquido de uma IF com um número maior ou menor de vulnerabilidades. As ferramentas usadas no estudo foram os *scanners* (VEGA, 2022) e (SKIPFISH, 2022). Para coletar as vulnerabilidades foi utilizada uma máquina virtual com o sistema operacional Linux SEEDUbuntu12.04 disponibilizado pelo Departamento de Computação da Universidade Syracuse.

O estudo de (REIS et al., 2018) negou a hipótese inicial e concluiu que nenhum indicativo mostrou que exista uma forte correlação entre o patrimônio líquido de uma IF e a segurança de seu portal de Internet Banking, porém foi possível encontrar uma correlação forte entre o número de acessos e a alta gravidade das vulnerabilidades encontradas nos respectivos sistemas.

De forma análoga ao trabalho de (REIS et al., 2018), (SENA et al., 2017) usa uma abordagem que faz uso de um único *scanner*, o (NETSPARKER, 2022), e os sites analisados compõem um total de 40 portais de prefeituras do Estado da Paraíba. O trabalho classificou as vulnerabilidades conforme a (OWASP, 2017). Os resultados encontrados no estudo revelaram um total de 822 vulnerabilidades, sendo 30% delas de alta criticidade, o que, conforme (SENA et al., 2017) indica fragilidade, por parte da Administração Pública, no gerenciamento e controle da segurança da informação dos portais analisados.

Semelhantemente ao trabalho de (SENA et al., 2017), o estudo de (COSTA et al., 2017) analisa a segurança dos portais de governos eletrônicos dos 26 estados, do distrito federal e do governo federal sob a perspectiva do número de vulnerabilidades encontrado em cada portal e verifica a hipótese referente a uma possível relação do poderio econômico dos estados donos dos portais com o número de vulnerabilidades encontrado. A abordagem usada por (COSTA et al.,

2017) utiliza dois *scanners* de vulnerabilidades, o (SKIPFISH, 2022) e o (UNISCAN, 2022). Em adição, o estudo utiliza o (NMAP, 2022) para varredura de portas abertas nos portais analisados.

O trabalho de (COSTA et al., 2017) detectou um grande número de vulnerabilidades nos portais e teve como conclusão o fato da hipótese ter sido negada, e, dessa forma, estados mais ricos não possuem um menor número de vulnerabilidades comparado com os mais pobres, pelo contrário, o estado que apresentou o maior número de vulnerabilidades foi o Rio de Janeiro.

Por fim, o estudo de (SANTOS et al., 2005) faz uso de um único *scanner*, o (NESSUS, 2022), 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., 2005) é chamado de *g-Quality*. Este método consiste da avaliação de sites considerando oito critérios: usabilidade, interoperabilidade, segurança, privacidade, veracidade da informação, agilidade do serviço e transparência.

Como contribuições do trabalho de (SANTOS et al., 2005), tem-se que os resultados, principalmente para prefeituras, não foram bons, isto é, houve um número médio alto de vulnerabilidades. Os estados e as capitais obtiveram um número médio menor de vulnerabilidades comparado aos municípios. Além disso, foi observado um grande número de portas abertas em determinados sites.

Os estudos mencionados acima respaldaram a metodologia deste trabalho e auxiliaram a estruturar uma análise de vulnerabilidades de portais de governos eletrônicos. Este estudo se assemelha muito, em particular, ao trabalho de (SENA et al., 2017), pois é usado apenas um *scanner* de vulnerabilidades e não há varredura de portas abertas nos portais; ademais, como pontos divergentes, este estudo analisa vulnerabilidades dos portais de todos os muncípios de Alagoas e foca em portais do Poder Legislativo, os quais são consideravelmente menos examinados neste tipo de análise.

3 Avaliação de segurança dos portais web das câmaras municipais alagoanas

3.1 Metodologia

Este trabalho é um estudo de caso que foca em analisar vulnerabilidades dos portais das câmaras municipais de Alagoas. A fim de alcançar esse objetivo, foram levantadas as URLs e as empresas/instituições desenvolvedoras dos portais. Com isso, tem-se o total de 100 URLs avaliadas, os municípios de São Brás e Traipu não possuem um portal dedicado a câmara legislativa municipal, logo não serão avaliados neste estudo. A metodologia é ilustrada em um diagrama na Figura 6.

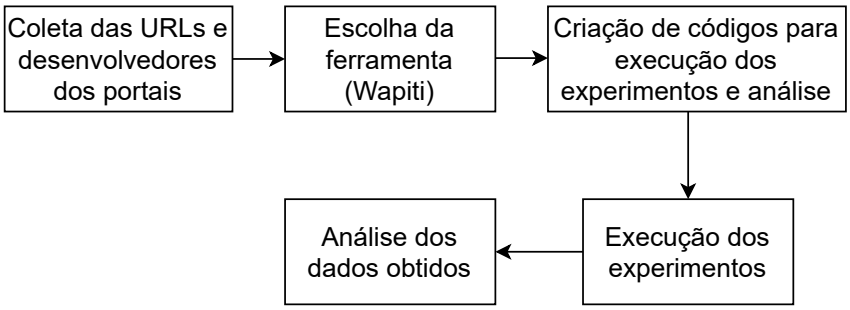


Figura 6 – Diagrama exibindo a metodologia utilizada. Produzido pelo autor.

Este trabalho busca analisar as vulnerabilidades dos portais das câmaras visando responder às perguntas descritas abaixo:

- Quais e quantas vulnerabilidades de segurança os portais das câmaras municipais têm?
- Os portais das câmaras municipais possuem vulnerabilidades críticas?
- Com relação à segurança, como os portais baseados no portal Modelo Interlegis se comportam em relação aos demais portais?
- Com relação à segurança, como os portais de cidades com maior PIB se comportam em relação aos demais portais?

A ferramenta de scanner de vulnerabilidades usada neste estudo foi o (WAPITI, 2022) na versão 3.1.4. Tal ferramenta foi escolhida por algumas razões, uma delas foram as conclusões do estudo de (ESCUDERO; ANDRADE; TERADA, 2020), ademais a ferramenta é gratuita e de código aberto, o que facilita o seu uso. Por fim, possui uma boa popularidade no meio acadêmico (823 artigos mencionam a ferramenta no Google Scholar) e possibilita exportar o relatório de vulnerabilidades em diversos formatos diferentes como HTML, JSON e XML (GOOGLE, 2023).

O relatório de vulnerabilidades do (WAPITI, 2022) foi exportado como JSON nos experimentos para ser possível enriquecer com informações adicionais importantes para a análise. Assim, foi criado um *script* que adiciona a rodada de execução do experimento, *timestamps* de início e fim de execução, cidade em que foi feita a varredura de vulnerabilidades e classificação da (OWASP, 2021) para os tipos de vulnerabilidade encontrados. Além disso, foi criado um *script* que gera arquivos CSV de análise a partir dos dados dos relatórios de vulnerabilidades. Desse modo, é possível analisar os dados usando linguagens de programação estatísticas e responder às perguntas levantadas por este trabalho.

Dessa forma, o experimento realizado para obtenção dos dados das vulnerabilidades para análise estatística posterior possui as vulnerabilidades como variáveis dependentes e as URLs dos portais das câmaras como único fator, com 100 possíveis níveis (tratamentos). As constantes do experimento são a ferramenta utilizada, uso dos módulos de ataque padrão, o ambiente experimental e os parâmetros da ferramenta: tempo máximo de *scan* em 720 segundos e tempo máximo de execução de cada módulo em 648 segundos.

Os valores de tempo máximo de *scan* e tempo máximo de execução de cada módulo foram definidos visando um tempo de execução máximo da ferramenta de 2 horas para cada portal de câmara em razão da média do tempo de execução de experimentos para URLs dos portais ser em média de 24 horas, o que poderia inviabilizar este trabalho.

Os experimentos não foram repetidos devido a atrasos e problemas na execução da ferramenta nos portais das câmaras quando contêineres (DOCKER, 2022) foram usados, possivelmente provocado por possíveis mecanismos de proteção dos sites contra múltiplas requisições vindas de um mesmo endereço IP.

O ambiente experimental utilizado foi 100 máquinas virtuais do tipo t2.micro do serviço EC2 da (AWS, 2022). Máquinas virtuais do tipo t2.micro possuem armazenamento EBS, 1 GB de memória RAM e 1 vCPU. O número de máquinas ser 100 se refere a quantidade de tratamentos. Foram realizadas 4 etapas de execução objetivando ter os relatórios de vulnerabilidade de cada URL, as primeiras três etapas executariam 30 URLs e a última 10 URLs.

Para executar as etapas foram usadas duas tecnologias - (TERRAFORM, 2022) e (ANSIBLE, 2022) - caracterizadas como ferramentas de Infraestrutura como código (IAC). O (TERRAFORM, 2022) foi usado para criar e iniciar a quantidade de máquinas necessária para a execução dos experimentos na AWS. Já o (ANSIBLE, 2022) foi utilizado para automatizar a instalação da ferramenta e dos experimentos nas máquinas. O repositório com os códigos usados neste trabalho está disponível no Apêndice A. Durante a execução das etapas com 30 URLs não foi possível executar todas as URLs simultaneamente, devido a limitações do Ansible.

Tal limitação acabou levando a mais atrasos na obtenção dos relatórios de vulnerabilidades dos portais das câmaras. Após a execução do (WAPITI, 2022) nas 100 URLs dos portais das câmaras para obtenção dos relatórios de vulnerabilidade foi feita a análise dos dados. A análise do número das vulnerabilidades coletadas foram realizadas considerando apenas o apontamento da ferramenta (WAPITI, 2022), desconsiderando o mérito de falso-positivos ou falso-negativos.

3.2 Vulnerabilidades encontradas e como corrigi-las

Em setembro, foram levantadas as URLs e as empresas/instituições desenvolvedoras dos portais das câmaras municipais alagoanas. Além disso, foi selecionada a ferramenta de detecção de vulnerabilidades. Em outubro foi feito um teste *end to end* com duas URLs de câmaras municipais para verificar a eficácia da ferramenta e o formato do relatório de vulnerabilidades gerado. Posteriormente, foram criados *scripts* de automatização de execução da ferramenta.

As varreduras de segurança acontecem em duas etapas: coleta de páginas da URL base do portal (exemplo: www.arapiraca.al.leg.br) e execução dos módulos de ataque nas páginas coletadas em busca de vulnerabilidades. Após a última etapa, são gerados os relatórios de vulnerabilidades. Durante a execução das varreduras, ocorreram problemas relacionados ao não escaneamento de páginas dos portais.

Desse modo, houveram atrasos nas varreduras de segurança, as quais foram concluídas apenas no fim de novembro. Os problemas ocorreram possivelmente em razão da presença de mecanismos de segurança presentes nos portais, como políticas específicas de *Firewall* ou adoção de Sistemas de Detecção/Prevenção de Intrusão.

Depois da resolução do problema relacionado ao escaneamento de páginas, foi obtida uma média de 53 minutos de duração de cada varredura. Apesar disso, houve um desvio padrão de 44 minutos, com o máximo chegando a 188. Já com relação a média de páginas escaneadas, ela foi de 287, não obstante, o desvio padrão foi de 773 páginas; 4 portais apresentaram um número de páginas escaneadas muito alto (mais de 1000), o que influenciou no alto número do desvio padrão.

A variação no número de páginas coletadas ocorreu devido a limitações do algoritmo de coleta de páginas da ferramenta. Em resumo, 5358 minutos foram necessários para a execução de todas as URLs, correspondendo a aproximadamente 90 horas ou quase 4 dias de execução ininterrupta. Além disso, 28743 páginas foram escaneadas considerando a execução em todos os portais.

O total de vulnerabilidades considerando todos os portais foi de 667. Os municípios que tiveram o maior número de vulnerabilidades (26) foram Maragogi e Penedo. Já os que tiveram o menor número de vulnerabilidades (1) foram Belém, Belo Monte, Cajueiro, Carneiros, Colônia Leopoldina, Girau do Ponciano, Maribondo, Satuba e Senador Rui Palmeira.

É importante ressaltar que existiram 9 portais de cidades nos quais não foi detectado nenhum tipo de vulnerabilidade. São elas: Arapiraca, Cacimbinhas, Chã Preta, Feliz Deserto, Jacaré dos Homens, Messias, Palestina, Piranhas e Teotônio Vilela. A Figura 7 ilustra o histograma da distribuição de vulnerabilidades entre as cidades analisadas. É possível notar barras mais altas no histograma que comunicam que 58% dos portais de cidades tiveram um número de vulnerabilidades menor ou igual a 5.

A Tabela 1 ilustra a distribuição entre as vulnerabilidades descobertas conforme a classificação da (OWASP, 2021). Pode ser observado que a vulnerabilidade com o maior número de ocorrências é a A5, seguido de perto por A4. A utilização da sigla A seguida do número se

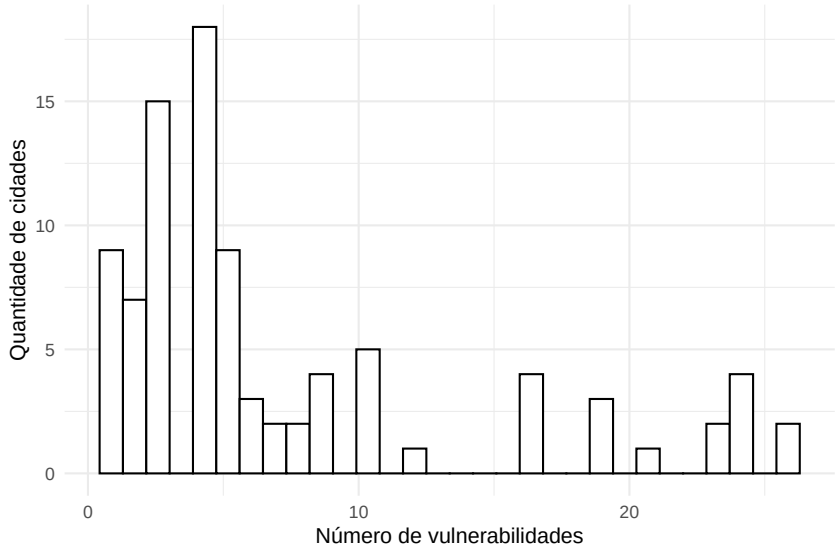


Figura 7 – Histograma da distribuição de vulnerabilidades entre as cidades. Produzido pelo autor.

refere a cada uma das vulnerabilidades citadas no Capítulo 2 sendo descritas abaixo:

- A1 - Quebra do Controle de Acesso
- A2 - Falhas Criptográficas
- A3 - Injeção
- A4 - Design Inseguro
- A5 - Configuração Incorreta de Segurança
- A6 - Componentes Vulneráveis e Desatualizados
- A7 - Falhas de Autenticação e Identificação
- A8 - Falhas de Integridade de Software e Dados
- A9 - Falhas de Monitoramento e Logs de Segurança
- A10 - Falsificação de requisições para o lado servidor

A execução dos experimentos gerou um número significativo de vulnerabilidades que estão resumidos nas tabelas da seção anterior. Os tipos detectados foram: Quebra do Controle de Acesso (A1), Injeção (A3), Design Inseguro (A4) e Configuração Incorreta de Segurança (A5). Em sua maioria as vulnerabilidades são de Configuração Incorreta de Segurança com 52,5%, seguido de Design Inseguro com 45,12% e por fim, Injeção com 2,25% e Quebra do Controle de Acesso com 0,13%. A Figura 8 ilustra essa distribuição de vulnerabilidades entre as categorias de forma proporcional.

Tabela 1 – Vulnerabilidades de acordo com a classificação da (OWASP, 2021)

Vulnerabilidades	Frequência
A1	1
A2	0
A3	15
A4	301
A5	350
A6	0
A7	0
A8	0
A9	0
A10	0

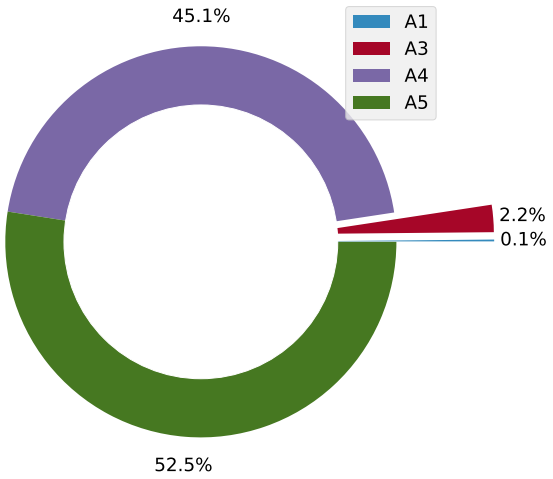


Figura 8 – Proporção de vulnerabilidades encontradas de acordo com a classficação OWASP. Produzido pelo autor.

3.2.1 Configuração Incorreta de Segurança

A vulnerabilidade “Configuração Incorreta de Segurança”, que possui a quinta posição da classificação da (OWASP, 2021) foi a mais frequente, com 350 ocorrências, ela foi encontrada em 81% dos portais de câmara analisados. Os portais que apresentaram o maior número (9) de vulnerabilidades desse tipo foram os de Jacuípe, Jequiá da Praia e Maravilha.

Outros portais que tiveram um número alto (8) de vulnerabilidades desse tipo foram os de Atalaia, Estrela de Alagoas, Porto Real do Colégio, Roteiro, Santana do Mundaú e Viçosa. No que diz respeito ao menor número (1) de vulnerabilidades desse tipo, apenas o portal de Carneiros obteve tal número.

Um exemplo de "Configuração Incorreta de Segurança"em alguns portais foi um cookie

sensível do navegador estar marcado sem a flag "HttpOnly". O que pode fazer com que um atacante consiga ler o conteúdo do cookie e obtenha a informação dele. Dessa forma, ele pode roubar a sessão do usuário no portal, conseguindo se passar por ele, tendo acesso a todas as suas informações.

A fim de eliminar as vulnerabilidades do tipo "Configuração Incorreta de Segurança"deve-se automatizar o processo de criação de sites com configurações de segurança adequadas, e se possível, implementar uma automatização de verificações de configurações de segurança em todos os sites.

Além disso, é necessário atentar-se para usar uma arquitetura de software segmentada e o mínimo de bibliotecas e *framewoks* necessário para a construção dos sites, evitando softwares desatualizados e por consequência componentes vulneráveis a ataques. Por fim, usar diretivas de segurança recomendadas, como *headers* seguros, estar sempre revisando e atualizando políticas de segurança e softwares essenciais para o funcionamento do site.

3.2.2 Design Inseguro

Sobre o segundo tipo de vulnerabilidade mais frequente “Design Inseguro”, ela possui a quarta posição da (OWASP, 2021) com 301 ocorrências sendo encontrada em 44% dos portais de câmara analisados. Os portais que apresentaram o maior número (19) de vulnerabilidades desse tipo foram os de Maragogi e Penedo.

Outros portais que tiveram um número alto (17) de vulnerabilidades desse tipo foram os de Barra de São Miguel, Joaquim Gomes, São Luís do Quintude e Porto Calvo. No que diz respeito ao menor número (1) de vulnerabilidades desse tipo, 20 portais obtiveram esse número, alguns foram deles foram os de: Satuba, Coqueiro Seco e Belém.

Um exemplo de "Design Inseguro"foi alguns portais das câmaras retornarem um erro interno do servidor (código de status 500) enquanto a ferramenta tentava injetar um *payload* em um parâmetro específico. Tal erro pode liberar informações sensíveis de logs do servidor, além de poder ser usado para ataques de negação de serviço.

Como forma de prevenção e remediação deste tipo de vulnerabilidade, é aconselhável implementar um ciclo de desenvolvimentos seguro de aplicações com profissionais da área de cibersegurança, usar bibliotecas de padrões de design seguro, limitar o consumo de recursos por usuário ou serviço. Além de tratar erros do site adequadamente para não incorrerem em liberação de informações sensíveis sobre os servidores onde o site está rodando.

3.2.3 Injeção

A terceira vulnerabilidade mais numerosa foi “Injeção”, que possui a terceira posição da classificação da (OWASP, 2021), com 15 ocorrências, ela foi encontrada em 10% dos portais de câmara analisados. O portal que apresentou o maior número de vulnerabilidades (3) deste tipo foi Delmiro Gouveia. Em seguida, outros portais que tiveram um número significativo de

vulnerabilidades desse tipo (2) foram os de Estrela de Alagoas, Roteiro e Santana do Mundaú. Um exemplo de "Injeção"foi alguns dos portais permitirem injeções de códigos JavaScript em um parâmetro específico (Cross Site Scripting - XSS). Fazendo com que um atacante consiga, por exemplo, realizar ações não autorizadas em nome do usuário que está acessando o site.

Para evitar e corrigir vulnerabilidades do tipo é preferível que se use uma API segura que não utilize o interpretador do banco de dados diretamente e impossibilite a execução de scripts maliciosos. Ademais, usar sanitização, filtro ou validação no lado do servidor é crucial para eliminar vulnerabilidades deste tipo. Nesse sentido, é importante escapar caracteres especiais de consultas e usar controles do SQL como o comando *LIMIT* a fim de mitigar possíveis vazamentos de dados.

3.2.4 Quebra do Controle de Acesso

O tipo de vulnerabilidade menos frequente foi "Quebra de Controle de Acesso", que possui a primeira posição da classificação da (OWASP, 2021), com apenas uma detecção em um portal de câmara específico. O portal afetado por tal vulnerabilidade foi o de Delmiro Gouveia. Exemplificando esse tipo de vulnerabilidade, hipoteticamente um site usa uma entrada externa para construir um nome de caminho que deve estar num diretório restrito, mas não neutraliza adequadamente sequências como ".."que podem ir para um local fora desse diretório. Isso permite que invasores atravessem o sistema de arquivos para acessar arquivos ou diretórios que estão fora do diretório restrito.

Para extinguir uma vulnerabilidade do tipo "Quebra do Controle de Acesso", faz-se fundamental tomar algumas medidas como: negar acesso a quaisquer recursos não públicos dentro do site, logar falhas de controle de acesso para os administradores do site, desabilitar listagem de diretórios e garantir que metadados ou arquivos de backup não estejam disponíveis para acesso público. Finalmente, limitar o número de requisições ao site em um mesmo período evita ataques por meio de ferramentas automatizadas.

3.2.5 Perspectiva de criticidade de vulnerabilidades

Dez portais de câmaras ou 10% de todos os portais analisados apresentaram vulnerabilidades críticas, as quais são, neste estudo, as três primeiras classificações da (OWASP, 2021). São eles: Água Branca, Atalaia, Boca da Mata, Cajueiro, Delmiro Gouveia, Estrela de Alagoas, Porto Real do Colégio, Roteiro, Santana do Mundaú e Viçosa. As vulnerabilidades presentes nesses portais são, conforme as categorias de vulnerabilidade do Wapiti, de XSS (Reflected Cross Site Scripting), de execução de comandos e de travessia de diretórios (Path Traversal). Segundo a classificação (OWASP, 2021), tais vulnerabilidades são de duas categorias: A3 Injeção e A1 Quebra do Controle de Acesso. A Figura 9 mostra a proporção de portais com vulnerabilidades críticas, não-críticas e sem vulnerabilidades.

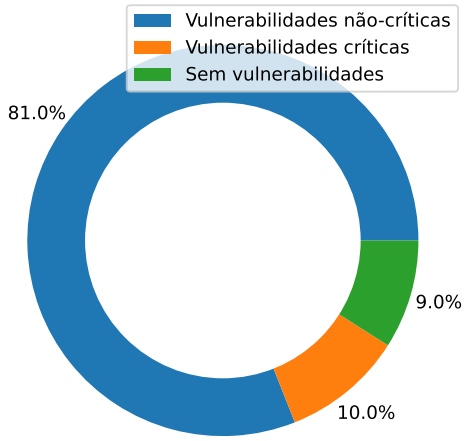


Figura 9 – Proporção de portais com vulnerabilidades críticas, não-críticas e sem vulnerabilidades. Produzido pelo autor.

3.3 Avaliação de vulnerabilidades em portais do tipo Interlegis

O governo federal disponibiliza um modelo de portal gratuito, rápido e pronto para uso das Câmaras Municipais e Assembleias Legislativas chamado de Portal Modelo Interlegis. Segundo o site do Portal Modelo, o portal está de acordo com os padrões web exigidos para portais públicos: usabilidade, acessibilidade e segurança. O padrão web que este trabalho avalia é a segurança, buscando entender se, de fato, há um bom nível de segurança nesse tipo de site em termos de número de vulnerabilidades.

Nesse sentido, os portais de câmara que utilizam esse modelo são: Fleixeiras, Japaratinga, Jaramataia, Junqueiro, Marechal Deodoro, Murici, Olho D'água Das Flores, Paripueira, Pilar e Santa Luzia do Norte. Foram detectadas em média 12 vulnerabilidades não-críticas dos tipos: Design Inseguro (80%) e Configuração Incorreta de Segurança (20%). A Tabela 2 mostra parâmetros estatísticos da distribuição de vulnerabilidades quanto a portais com e sem o modelo Interlegis do Governo Federal.

Tabela 2 – Vulnerabilidades quanto ao tipo de site de câmara

	Interlegis	Não-Interlegis
Média	12	7
Máximo	19	26
Desvio padrão	5,73	6,89

Percebe-se a partir dos dados apresentados que câmaras com modelo Interlegis de portal possuem um maior número de vulnerabilidades médio (cerca de 1,7x maior em média) em

comparação aos portais que não utilizam o modelo.

3.3.1 Criticidade de vulnerabilidades conforme o Portal Modelo Interlegis

Nenhum portal que usa o modelo Interlegis apresentou vulnerabilidades críticas de segurança. Já com relação aos demais portais que não usam Interlegis, cerca de 11% deles apresentaram alguma vulnerabilidade de segurança crítica, os portais com vulnerabilidades críticas foram citados anteriormente na Subseção 3.2.5.

3.4 Avaliação de vulnerabilidades em portais conforme o PIB das cidades

A Tabela 3 mostra parâmetros estatísticos da distribuição de vulnerabilidades quanto aos portais de cidades no top 10 de cidades de alagoas em termos de PIB de acordo com (WIKIPEDIA, 2022) e as que não estão. O Apêndice A mostra uma tabela com o top 10 de cidades alagoanas em PIB.

Tabela 3 – Vulnerabilidades quanto ao PIB das cidades

	Top 10 cidades	Outras
Média	11	7
Máximo	26	26
Desvio padrão	9,23	6,5

A partir dos dados expostos, nota-se que as cidades que estão no top 10 em PIB têm portais com número médio de vulnerabilidades de 11, número 1,6x maior do que o restante dos portais de cidade.

3.4.1 Criticidade de vulnerabilidades conforme o PIB das cidades

Dentre as cidades com maior PIB, o portal de Delmiro Gouveia apresentou um total de 4 vulnerabilidades críticas. Já com relação aos demais portais, cerca de 10% deles apresentaram alguma vulnerabilidade de segurança crítica, os portais com vulnerabilidades críticas foram citados anteriormente na Subseção 3.2.5.

4 Conclusão

Primeiramente, o presente trabalho alcançou seu objetivo com êxito ao verificar a existência de possíveis vulnerabilidades de segurança nos portais das câmaras municipais alagoanas. Além disso, este estudo descreveu e classificou as vulnerabilidades encontradas conforme a (OWASP, 2021) e expôs a proporção de portais de câmara com vulnerabilidades críticas e não críticas.

O número total de vulnerabilidades encontrado foi de 667, com relação a proporção de vulnerabilidades críticas: 81% dos portais apresentaram vulnerabilidades não críticas, 10% apresentaram vulnerabilidades críticas e 9% não apresentaram vulnerabilidades. As vulnerabilidades encontradas foram dos tipos: Quebra do Controle de Acesso (A1) com 0,13%, Injeção (A3) com 2,25%, Design Inseguro (A4) com 45,12% e Configuração Incorreta de Segurança (A5) com 52,5%.

Os municípios que tiveram o maior número de vulnerabilidades (26) foram Maragogi e Penedo. Já os que tiveram o menor número de vulnerabilidades (1) foram Belém, Belo Monte, Cajueiro, Carneiros, Colônia Leopoldina, Girau do Ponciano, Maribondo, Satuba e Senador Rui Palmeira.

Em seguida, este trabalho avaliou vulnerabilidades de portais de câmaras que utilizam o Portal Modelo Interlegis em comparação aos que não utilizam e concluiu que câmaras que usam o modelo obtiveram um número médio de vulnerabilidades maior (1,7x) do que câmaras que não usam. Todavia, não foram encontradas vulnerabilidades de segurança críticas em portais que usam o modelo. O número de vulnerabilidades não-críticas médio em portais Interlegis foi de 12 e os tipos encontrados foram Design Inseguro (80%) e Configuração Incorreta de Segurança (20%).

Finalmente, este trabalho avaliou vulnerabilidades nos portais sob a perspectiva das 10 cidades alagoanas com maior PIB (Dados sobre PIB disponíveis no Apêndice A). Os portais de câmara das cidades com maior PIB apresentaram um número médio de vulnerabilidades maior do que as outras câmaras. Contudo, apenas o portal de Delmiro Gouveia apresentou vulnerabilidades críticas (4).

Como trabalhos futuros, pode-se verificar a existência de possíveis vulnerabilidades nos sites das prefeituras dos municípios alagoanos ou verificar a existência de vulnerabilidades em câmaras municipais de outro estado. Uma nova perspectiva de análise pode ser adotada considerando o uso de mais de uma ferramenta.

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.

AGRA, A. D.; BARBOZA, F. F. M. **Segurança de sistemas da informação**. [S.l.]: Grupo A, 2019. Citado na página 11.

ANSIBLE. **Ansible**. 2022. Disponível em: <https://www.ansible.com/>. Citado na página 24.

AWS. **Amazon EC2**. 2022. Disponível em: <https://aws.amazon.com/pt/ec2/>. Citado na página 24.

BRANQUINHO, T.; BRANQUINHO, M. **Segurança Cibernética Industrial**. [S.l.]: Alta Books, 2021. Citado na página 12.

CERT.BR. **Estatísticas do CERT.br - Incidentes**. 2020. Disponível em: <https://www.cert.br/stats/incidentes/>. Citado 2 vezes nas páginas 7 e 11.

COSTA, J. V. P. et al. Análise de vulnerabilidades de segurança em portais de governos eletrônicos. Universidade Federal de Uberlândia, 2017. Citado 2 vezes nas páginas 21 e 22.

CTIR. **Estatísticas resultantes do trabalho de detecção, triagem, análise e resposta a incidentes cibernéticos**. 2022. Disponível em: <https://www.gov.br/ctir/pt-br/assuntos/ctir-gov-em-numeros/visao-geral>. Citado 2 vezes nas páginas 7 e 12.

DOCKER. **Docker**. 2022. Disponível em: <https://www.docker.com/>. Citado na página 24.

DOUPÉ, A.; COVA, M.; VIGNA, G. Why johnny can't pentest: An analysis of black-box web vulnerability scanners. In: SPRINGER. **International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment**. [S.l.], 2010. p. 111–131. Citado na página 19.

ESCUDERO, D. P.; ANDRADE, E. R.; TERADA, R. Estudo comparativo do módulo rastreador de scanners de vulnerabilidade web de código aberto. In: SBC. **Anais da XVIII Escola Regional de Redes de Computadores**. [S.l.], 2020. p. 154–160. Citado 3 vezes nas páginas 19, 21 e 23.

FERRÃO, I.; SÁ, G. N. B.; KREUTZ, D. L. Detecção de vulnerabilidades em ambientes web controlados. **Anais do Salão Internacional de Ensino, Pesquisa e Extensão**, v. 10, n. 2, 2018. Citado 2 vezes nas páginas 19 e 20.

FONSECA, J.; VIEIRA, M.; MADEIRA, H. Testing and comparing web vulnerability scanning tools for sql injection and xss attacks. In: IEEE. **13th Pacific Rim international symposium on dependable computing (PRDC 2007)**. [S.l.], 2007. p. 365–372. Citado na página 20.

GOOGLE. **Google Scholar**. 2023. Disponível em: <https://scholar.google.com.br/scholar?hl=pt-BR&as_sdt=0%2C5&q=wapiti+vulnerability+scanner&btnG=>. Citado na página 23.

IDRISSI, S. et al. Performance evaluation of web application security scanners for prevention and protection against vulnerabilities. **International Journal of Applied Engineering Research**, Research India Publications, v. 12, n. 21, p. 11068–11076, 2017. Citado 2 vezes nas páginas 20 e 21.

KIM, D.; SOLOMON, M. G. Fundamentos de segurança de sistemas de informação. **Rio de Janeiro: LTC**, 2014. Citado na página 14.

LIS, A. **Comparison and analysis of web vulnerability scanners**. Dissertação (B.S. thesis), 2019. Citado 3 vezes nas páginas 7, 16 e 17.

MARTINELO, C. A. G.; BELLEZI, M. A. Análise de vulnerabilidades com openvas e nessus. **Revista TIS**, v. 3, n. 1, 2014. Citado na página 15.

MIMECAST. **What is WannaCry Ransomware and How Does It Work?** 2021. Disponível em: <https://www.mimecast.com/blog/all-you-need-to-know-about-wannacry-ransomware/>. Citado na página 14.

MOHAMMED, R. Assessment of web scanner tools. **International Journal of Computer Applications**, Foundation of Computer Science, v. 133, n. 5, p. 1–4, 2016. Citado 2 vezes nas páginas 20 e 21.

NESSUS. **Nessus Vulnerability Scanner**. 2022. Disponível em: <https://www.tenable.com/products/nessus>. Citado na página 22.

NETSPARKER. **Netsparker – Web Application Security Scanner**. 2022. Disponível em: <https://www.100security.com.br/netsparker>. Citado na página 21.

NIKTO. **Nikto Web Server Scanner**. 2022. Disponível em: <https://github.com/sullo/nikto>. Citado na página 19.

NMAP. **The Network Mapper**. 2022. Disponível em: <https://github.com/nmap/nmap>. Citado na página 22.

OWASP. **OWASP Top 10 - 2017**. 2017. Disponível em: <https://owasp.org/www-project-top-ten/2017/Top_10>. Citado na página 21.

OWASP. **OWASP Top 10 - 2021**. 2021. Disponível em: <https://owasp.org/Top10/>. Citado 8 vezes nas páginas 8, 15, 24, 25, 27, 28, 29 e 32.

PAROS. **Paros**. 2022. Disponível em: <https://gitlab.com/kalilinux/packages/paros>. Citado na página 19.

PINHO, J. A. G. d. Investigando portais de governo eletrônico de estados no brasil: muita tecnologia, pouca democracia. **Revista de Administração Pública**, SciELO Brasil, v. 42, p. 471–493, 2008. Citado na página 18.

PYTHON. **Python**. 2022. Disponível em: <https://www.python.org/>. Citado na página 17.

REIS, A. P. d. et al. Análise de vulnerabilidades de segurança em sistemas de internet banking utilizando ferramentas de código aberto. Universidade Federal de Uberlândia, 2018. Citado na página 21.

SANTOS, M. C. et al. Segurança em sítios de governo eletrônico brasileiros: um estudo de caso. **Instituto de Computação–Universidade Federal Fluminense (UFF)**, 2005. Citado na página 22.

SENA, A. S. d. et al. Portais de governo eletrônico em municípios do estado da paraíba: análise sob a óptica da segurança da informação. Universidade Federal da Paraíba, 2017. Citado 2 vezes nas páginas 21 e 22.

SENADO. **Brasil é 2º no mundo em perdas por ataques cibernéticos, aponta audiência**. 2022. Disponível em: <https://www12.senado.leg.br/noticias/materias/2019/09/05/brasil-e-2o-no-mundo-em-perdas-por-ataques-ciberneticos-aponta-audiencia>. Citado na página 11.

SKIPFISH. **Skipfish Web Application Security Scanner**. 2022. Disponível em: <https://gitlab.com/kalilinux/packages/skipfish>. Citado 3 vezes nas páginas 19, 21 e 22.

TERRAFORM. **Terraform**. 2022. Disponível em: <https://www.terraform.io/>. Citado na página 24.

UNISCAN. **Uniscan Web Vulnerability Scanner**. 2022. Disponível em: <https://www.kali.org/tools/uniscan/>. Citado na página 22.

UTICA. **Ten Ways Evolving Technology Affects Cybersecurity**. 2020. Disponível em: <https://programs.online.utica.edu/resources/article/ten-ways-evolving-technology-affects-cybersecurity>. Citado na página 14.

VEGA. **Vega Vulnerability Scanner**. 2022. Disponível em: <https://subgraph.com/vega/>. Citado 2 vezes nas páginas 19 e 21.

VIEIRA, M.; ANTUNES, N.; MADEIRA, H. Using web security scanners to detect vulnerabilities in web services. In: IEEE. **2009 IEEE/IFIP International Conference on Dependable Systems & Networks**. [S.l.], 2009. p. 566–571. Citado na página 20.

W3AF. **W3af Web Vulnerability Scanner**. 2022. Disponível em: <https://github.com/andresriancho/w3af>. Citado na página 19.

WAPITI. **Wapiti - Web Vulnerability Scanner**. 2022. Disponível em: <https://github.com/wapiti-scanner/wapiti>. Citado 5 vezes nas páginas 17, 19, 21, 23 e 24.

WENDT, E.; JORGE, H. V. N. **Crimes Cibernéticos (2a. edição): Ameaças e procedimentos de investigação**. [S.l.]: Brasport, 2013. Citado na página 11.

WIKIPEDIA. **Lista de municípios de Alagoas por PIB**. 2022. Disponível em: <https://pt.wikipedia.org/wiki/Lista_de_munic%C3%ADpios_de_Alagoas_por_PIB>. Citado 3 vezes nas páginas 7, 31 e 39.

Apêndices

APÊNDICE A – Repositório de códigos usados

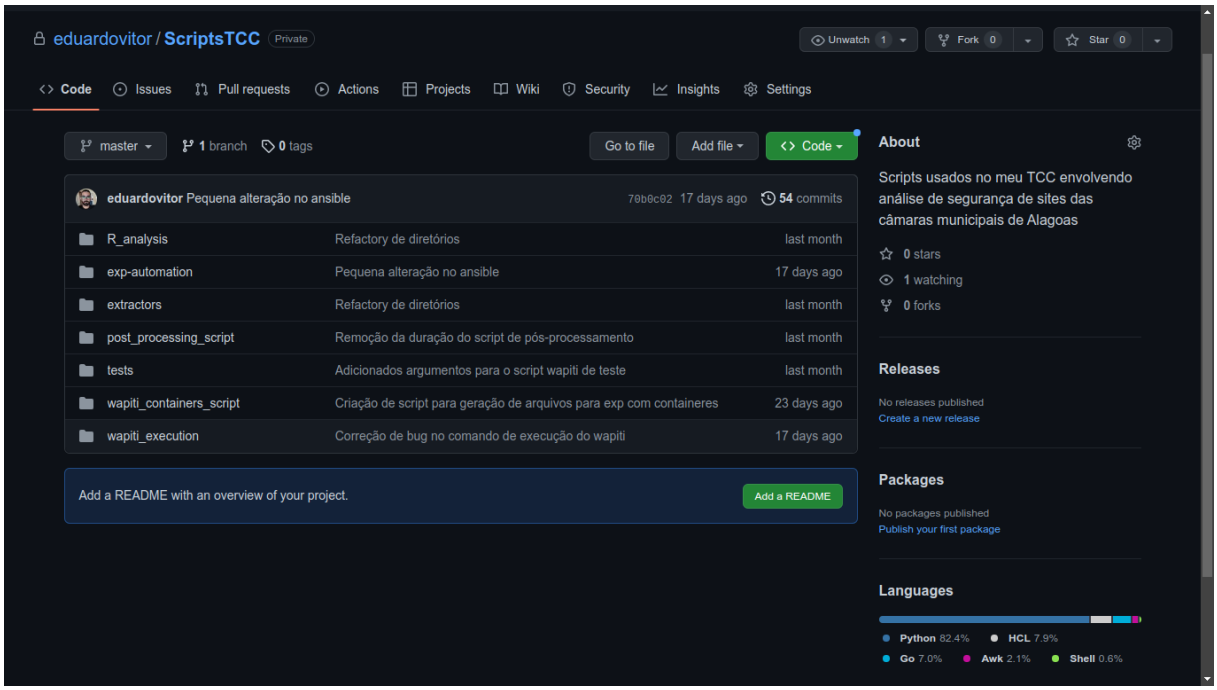


Figura 10 – Repositório do Github com os códigos usados. Imagem produzida pelo autor.

Anexos

ANEXO A – Produto Interno Bruto (PIB) dos municípios Alagoanos

[ocultar]

Lista de municípios de Alagoas por PIB

Adicionar línguas

Artigo Discussão Ler Editar Ver histórico

Origem: Wikipédia, a enciclopédia livre.

Esta é uma lista dos municípios de Alagoas por PIB, segundo estimativas do IBGE em 2012, 2011 e em 2010.[1][2][3]

PIB em 2012

Posição	Mapa	Município	PIB (mil R$)
1		Maceió	**13 694 808**
2		Arapiraca	2 416 888
3		Marechal Deodoro	1 122 913
4		São Miguel dos Campos	884 362
5		Coruripe	729 741
6		Rio Largo	603 644
7		União dos Palmares	535 734
8		Palmeira dos Índios	516 042
9		Penedo	433 241
10		Delmiro Gouveia	392 709
11		Pilar	314 974
12		São Luís do Quitunde	305 825
13		Atalaia	277 360
14		Campo Alegre	274 317
15		Teotônio Vilela	263 853
16		Santana do Ipanema	258 177
17		Igreja Nova	238 738
18		Boca da Mata	213 593
19		Porto Calvo	200 391
20		São José da Laje	190 901

Figura 11 – PIB dos municípios alagoanos. Extraído do artigo de (WIKIPEDIA, 2022)