INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO NORTE

GILENO CORDEIRO DUARTE

ANÁLISE DE VULNERABILIDADES SOBRE A APLICAÇÃO WEB DO CINTE-RN: UM ESTUDO DE CASO

NOVA CRUZ / RN

2022

GILENO CORDEIRO DUARTE

ANÁLISE DE VULNERABILIDADES SOBRE A APLICAÇÃO WEB DO CINTE-RN: UM ESTUDO DE CASO

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, em cumprimento às exigências legais, como requisito parcial à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientadora: Ma. Maria Jane de Queiroz.

NOVA CRUZ / RN

2022

Catalogação da publicação na fonte
Biblioteca do Instituto Federal de Educação, Ciência e Tecnologia do RN *Campus Nova Cruz*

D812a DUARTE, Gileno Cordeiro.

Análise de vulnerabilidades sobre a aplicação web do CINTE- RN: um estudo de caso. / Gileno Cordeiro Duarte. – Nova Cruz/RN, 2022.
49f.

Orientador: Profª. Ma. Maria Jane de Queiroz - Monografia (Monografia em ciências exatas e da terra). – Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte – Nova Cruz/RN, 2022.

1. Aplicação web – Monografia. 2. Segurança de software – Monografia. 3. Vulnerabilidade de software - Monografia. I. QUEIROZ, Maria Jane de. II. Título.

IF
R
N

CDU: 004.415.53(0813.2)

GILENO CORDEIRO DUARTE

ANÁLISE DE VULNERABILIDADES SOBRE A APLICAÇÃO WEB DO CINTE-RN: UM ESTUDO DE CASO

Trabalho de Conclusão de Curso apresentado ao Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, em cumprimento às exigências legais como requisito parcial à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Aprovado em: ___/___/_____.

BANCA EXAMINADORA

_____

Ma. Maria Jane de Queiroz – Orientadora

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

_____

Dr. Francisco Ary Alves de Souza – Examinador

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

_____

Ma. Aislânia Alves de Araujo – Examinadora

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

Dedico esse projeto a todos que estiveram comigo e me apoiaram durante esse tempo todo, em especial a minha mãe que lutou para que iniciasse minha vida acadêmica e deu início a tudo que construí até hoje.

RESUMO

Este trabalho teve como objetivo realizar uma análise de vulnerabilidades sobre a aplicação web do Centro de Inovações Tecnológicas do Rio Grande do Norte (CINTE-RN), bem como sobre o servidor web que a hospeda, induzindo uma reflexão a respeito da importância da segurança da informação no âmbito do desenvolvimento e manutenção de software. A escolha do tema surgiu diante da relevância do assunto, dado o fato de que a segurança nem sempre é considerada durante o processo de desenvolvimento de *software*. Desta forma, foi realizada uma pesquisa bibliográfica sobre o tema e, em seguida, um estudo de caso aplicado, no qual foi adotada a metodologia para *pentest* denominada "caixa preta", bem como as ferramentas para *scanner* de vulnerabilidades NMAP e Nessus à aplicação web do CINTE. Corroborando com a hipótese inicialmente levantada, os resultados da análise identificaram um total de 216 possíveis vulnerabilidades, classificadas da seguinte forma: 5% de nível crítico (11), 16% de nível médio (34), 3% de nível baixo (7) e 76% de informativos (164). Além da análise da aplicação, foi realizada uma análise de vulnerabilidades no servidor *web* que apresentou problemas como portas e versões de serviços expostos e serviços desnecessários operando na máquina. Após a análise, foi elaborado um ambiente de testes e sugeridas soluções para a prevenção de ataques que pudessem explorar as vulnerabilidades encontradas, tanto na aplicação web quanto no servidor de hospedagem. Ao final do trabalho, conclui-se que a maioria das vulnerabilidades de nível médio e crítico poderiam ser evitadas com a aplicação de *patches* de segurança e a atualização de tecnologias, além de configurações simples no servidor para dificultar a obtenção de informações pelos atacantes.

Palavras-chave: Análise; Aplicação; Segurança; Vulnerabilidades.

ABSTRACT

This work aimed to perform a vulnerability analysis on the web application of the Center for Technological Innovations of Rio Grande do Norte (CINTE-RN), as well as on the web server that hosts it, inducing a reflection on the importance of information security in the scope of software development and maintenance. The choice of the meme arose in view of the relevance of the subject, given the fact that security is not always considered during the software development process. Thus, a bibliographical research was carried out on the subject and then an applied case study, in which the pentest methodology called "black box" was adopted, as well as the tools for scanner vulnerabilities NMAP and Nessus to the web application of CINTE. Corroborating the hypothesis initially raised, the results of the analysis identified a total of 216 possible vulnerabilities, classified as follows: 5% of critical level (11), 16% of medium level (34), 3% of low level (7) and 76% of informative (164). In addition to the application analysis, a vulnerability analysis was performed on the web server that presented problems such as ports and versions of exposed services and unnecessary services operating on the machine. After the analysis, a testing environment was developed and solutions were suggested to prevent attacks that could exploit the vulnerabilities found, both in the web application and on the hosting server. At the end of the work, it is concluded that most medium-level and critical vulnerabilities could be avoided by applying security patches and updating technologies, as well as simple settings on the server to make it difficult for attackers to obtain information.

Keywords: Analysis; Application; Safety; Vulnerabilities.

LISTA DE FIGURAS

Figura 1 – Principais Vulnerabilidades em Sistemas Web 17
Figura 2 – Topologia com DMZ 19
Figura 3 – Pagina Inicial cinte.com.br 26
Figura 4 – Scan com flag -O 28
Figura 5 – Scan com flag -sV 29
Figura 6 – Registros MX CINTE 31
Figura 7 – Domínio Mx-Vip-02.Kinghost.Net 31
Figura 8 – Domínio mx-vip-01.kinghost.net 32
Figura 9 – Resultado de consulta utilizando whois 33
Figura 10 – Resultado do teste utilizando a ferramenta Nessus 35
Figura 11 – PHPMyAdmin vulnerabilidade de SQLInjection 36
Figura 12 – PHPMyAdmin vulnerabilidade de CSRF 37
Figura 13 – Identificação de arquivo phpinfo.php 38
Figura 14 – Detecção de vulneralbilidade HSTS 39
Figura 15 – Máquina SRV-01 (servidor) 41
Figura 16 – Máquina SRV-02 (atacante) 41
Figura 17 – Teste no Servidor sem Alterações 42
Figura 18 – Alteração de porta padrão do SSH 43
Figura 19 – Configuração no arquivo httpd.conf 44
Figura 20 – Teste NMAP com serviços configurados 44
Figura 21 – Configuração no arquivo 000-default.conf 46

LISTA DE QUADROS

Quadro 1 – REQUISITOS ESPECÍFICOS DE SEGURANÇA

LISTA DE SIGLAS

API	*Application Programing Interface*
CERT.br	*Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil*
CINTE	*Centro de Inovações Tecnológicas*
CSRF	*Cross Site Request Forgery*
FTP	*File Transfer Protocol*
HSTS	*Strict-Transport-Security*
HTTP	*HyperText Transfer Protocol*
HTTPS	*HyperText Transfer Protocol Secure*
IFRN	Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte
IMAP	*Internet Message Access Protocol*
LFI	*Local File Incursion*
LGPD	*Lei Geral de proteção de dados*
NMAP	*Network Mapper*
OWASP	*Open Web Application Security Project*
PHP	*Hypertext Preprocessor*
POP 3	*Post Office Protocol*
SGD	Secretaria de Governo Digital do Ministério da Economia
SMTP	*Simple Mail Transfer Protoco*l
SQL	*Structured Query Language*
SSH	*Open Secure Shell*

W3AF *Open Source Web Application Security Scanner*

SUMÁRIO

1 INTRODUÇÃO **12**

1.1 OBJETIVOS 13

1.1.1 Objetivo Geral **13**

1.1.2 Objetivos Específicos **13**

1.2 ESTRUTURA DO TRABALHO 14

2 REFERENCIAL TEÓRICO **15**

2.1 VULNERABILIDADE, AMEAÇA E ATAQUE 15

2.2 PARÂMETROS PARA UMA APLICAÇÃO SEGURA 17

2.3 TRABALHOS RELACIONADOS 22

3 PROCEDIMENTOS METODOLÓGICOS **24**

4 ESTUDO DE CASO **25**

4.1 APLICAÇÃO ALVO 25

4.2 MÁQUINA DE ORIGEM DA VARREDURA 26

4.3 ANÁLISE DE VULNERABILIDADES 27

4.3.1 No Servidor de Hospedagem **27**

4.3.2 No Website da CINTE-RN **34**

5 SUGESTÕES PARA PREVENÇÃO DE ATAQUES NO SISTEMA ANALISADO **40**

5.1 NO SERVIDOR DE HOSPEDAGEM 40

5.2 NO WEBSITE DA CINTE-RN 45

6 CONSIDERAÇÕES FINAIS **47**

REFERÊNCIAS **48**

APÊNDICE A – Permissão para testes no ambiente **52**

1 INTRODUÇÃO

A criação de sistemas *web*, assim como o desenvolvimento de *software* em geral, é um processo complexo, com várias etapas: desde o levantamento de requisitos e análise de requisitos, projeto, implementação, testes e implantação, além de um possível período inicial de *feedback* e correção de pequenos *bugs* que podem estar contidos na aplicação.

Segundo Montanheiro (2018, p. 11), "As principais falhas em aplicações web ocorrem durante o desenvolvimento do projeto, a fase de codificação do sistema muitas das vezes é feita às pressas e sempre extrapolando o prazo". Dada a pressão pela entrega de um *software* funcional e o curto prazo para a codificação, a segurança no desenvolvimento de uma aplicação é comumente negligenciada, tais condições exigem que as equipes de desenvolvimento sejam obrigadas a reduzir as fases do processo, focando mais na construção e desenvolvimento do código em si, amenizando a importância de testes e avaliações tanto de desempenho quanto de segurança sem revisá-lo e/ou aprimorá-lo.

Segundo Akamai (2022, p. 6), ataques a aplicações *web* triplicaram no ano de 2021, em comparação ao ano anterior que havia atingido o recorde. Na lista de ataques praticados, os três primeiros da lista, portanto, os mais frequentes são: *SQL Injection* (6,2 bilhões de tentativas), *Local File Inclusion (LFI)* (3,3 bilhões de tentativas) e *Cross-Site Scripting* (1,019 bilhão de tentativas).

As novas tecnologias tendem a ser cada vez mais presentes no cotidiano, com o surgimento de novas ferramentas *online* que auxiliam a população em tarefas de seu dia a dia, porém, com isso há um crescimento no número de pessoas que se mantêm conectadas e com dados pessoais e até bancários na rede, despertando o interesse dos *crackers* para invasões.

O intuito desses ataques podem ser diversos, além de ataques a aplicações, existem também os ataques direcionados aos servidores que hospedam estes sistemas. Em uma publicação feita pelo Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil (CERT.br, 2020) quanto aos incidentes reportados entre janeiro e dezembro de 2020, foi possível observar que a maioria consistia em "*scans"* (59,85%), que se tratam de varreduras feitas em redes de computadores em busca de máquinas com sistemas operacionais ou serviços vulneráveis, portanto, este procedimento também é conhecido como análise de vulnerabilidades.

Após uma varredura e a descoberta de uma vulnerabilidade, a próxima etapa é o deferimento de um ataque. Um ataque acontece a partir da exploração de uma ou mais vulnerabilidades presentes no código, no banco de dados e/ou no servidor onde a aplicação esteja hospedada.

A depender do objetivo, o ataque receberá uma nomenclatura apropriada. Por exemplo, um ataque *web* visa comprometer um servidor *web* (responsável pela hospedagem de uma aplicação) a partir de ações como a desfiguração de páginas ou o furto de dados confidenciais. Conforme o estudo do CERT.br, ataques *web* corresponderam a uma parcela de 3,99% dos ataques realizados no Brasil em 2020.

Dada a relevância e a urgência do tema, este trabalho apresenta um estudo de caso relacionado à segurança de uma aplicação *web* específica: o sistema *web* do Centro de Inovações Tecnológicas do Rio Grande do Norte (CINTE-RN).

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Realizar uma análise de vulnerabilidades sobre a aplicação web do Centro de Inovações Tecnológicas do Rio Grande do Norte (CINTE-RN) e sobre o serviço que a hospeda, a fim de suscitar uma reflexão sobre a importância da segurança tanto nas etapas de desenvolvimento quanto na hospedagem de uma aplicação web.

1.1.2 Objetivos Específicos

- Analisar a aplicação web do Centro de Inovações Tecnológicas do Rio Grande do Norte (CINTE-RN) quanto à existência de vulnerabilidades;
- Apresentar soluções para as possíveis vulnerabilidades detectadas;
- Refletir sobre a importância de boas práticas de segurança durante a codificação e hospedagem de aplicações web.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está organizado da seguinte forma: o capítulo 1 é composto pela introdução, motivação e objetivos do trabalho; o capítulo 2 apresenta o referencial teórico e trabalhos relacionados a este; o capítulo 3 apresenta os procedimentos metodológicos empregados; o capítulo 4 é composto pelo estudo de caso realizado a partir da análise de vulnerabilidades na aplicação web do CINTE-RN, bem como os resultados obtidos a partir desta análise; o capítulo 5 apresenta sugestões para a prevenção de ataques na aplicação analisada e o capítulo 6 é composto pelas considerações finais.

2 REFERENCIAL TEÓRICO

Nos próximos tópicos serão apresentados conceitos relacionados ao desenvolvimento deste trabalho.

2.1 VULNERABILIDADE, AMEAÇA E ATAQUE

De acordo com CERT.br (2021),

> Uma vulnerabilidade é definida como uma condição que, quando explorada por um atacante, pode resultar em uma violação de segurança. Exemplos de vulnerabilidades são falhas no projeto, na implementação ou na configuração de programas, serviços ou equipamentos de rede.

A existência de vulnerabilidades e, consequentemente, a existência da possibilidade de explorá-las é conhecida como ameaça. Segundo Stallings (2014, p. 11) uma ameaça é a "chance de violação da segurança que existe quando há uma circunstância, capacidade, ação ou evento que poderia quebrar a segurança e causar danos".

Um ataque, por sua vez, é uma derivação "de uma ameaça inteligente; ou seja, um ato inteligente que é uma tentativa deliberada (especialmente no sentido de um método ou técnica) de fugir dos serviços de segurança e violar a política de segurança de um sistema" (STALLINGS, 2014, p. 11).

Desta forma, fica perceptível a ligação direta entre os três termos, ou seja, uma ameaça é a existência de uma vulnerabilidade que pode ser explorada por alguém mal-intencionado, originando assim um ataque.

Para exemplificar os termos citados anteriormente, pode-se citar o recente caso da vulnerabilidade encontrada na biblioteca Log4j (THE APACHE SOFTWARE FOUNDATION, 2021). Esta API (*Application Programming Interface*) baseada em Java foi originalmente desenvolvida pela *Apache Software Foundation* com o objetivo de armazenar registros (*logs*) relacionados ao comportamento de aplicações, acessos feitos pelos usuários e alterações de código.

No entanto, a vulnerabilidade encontrada no dia 09 de dezembro de 2021, denominada CVE-2021-44228 e mais conhecida como Log4Shell (TREND MICRO, 2021), é uma ameaça a grandes empresas, como Minecraft, IBM, Oracle, AWS e Cloudflare, as quais utilizam esta biblioteca, portanto, estão suscetíveis a um possível ataque (KORN, 2021).

Um ataque possível a partir da exploração da vulnerabilidade da API Log4j consiste na execução remota de código, permitindo o acesso de pessoas mal intencionadas a servidores para finalidades potencialmente prejudiciais a empresas e usuários (TREND MICRO, 2021).

A partir deste exemplo é possível observar que a existência de uma vulnerabilidade (CVE-2021-44228) em um *software* (biblioteca Log4j) representa uma ameaça ao sistema, pois esta vulnerabilidade pode ser explorada, culminando em um ataque que possibilite o acesso remoto ao servidor por parte do atacante.

Uma vez que o ataque desferido seja bem-sucedido, o atacante pode ainda praticar outros crimes como o furto ou o sequestro de dados, a infecção de outras máquinas da rede por vírus, a aplicação de golpes a usuários legítimos, dentre outros danos.

Apesar de ser considerado crime, ataques continuam a ocorrer e de forma ainda mais frenética nos últimos tempos, sendo esta uma preocupação recorrente das grandes corporações. Por isso, diferentes metodologias foram criadas para a realização dos chamados testes de penetração (*Penetration testing*) ou *pentest*.

Um *pentest* é um conjunto de testes que permitem a análise e exploração de vulnerabilidades, aplicados por uma empresa de segurança da informação – devidamente contratada e com permissão para tal serviço – a uma rede de computadores, aos sistemas operacionais e *softwares* utilizados em um órgão público ou privado (MORENO, 2015, p. 57).

O objetivo do teste de invasão, como também é conhecido o *pentest*, é identificar, ativamente, possíveis vulnerabilidades existentes na rede ou *software*, a fim orientar a equipe de Tecnologia da Informação (TI) na tomada de decisões para mitigar tais falhas.

Este trabalho apresenta uma análise de vulnerabilidades que aborda apenas a investigação superficial do ambiente, mapeando vulnerabilidades sem realmente as explorar. Segundo Moreno (2015, p. 23), "é similar ao pentest, pois também realiza uma auditoria para encontrar falhas; embora ela faça somente um levantamento das vulnerabilidades, e algumas etapas não são executadas, para não comprometer o sistema auditado".

2.2 PARÂMETROS PARA UMA APLICAÇÃO SEGURA

Quando se trata de estabelecer parâmetros para se elaborar e manter uma aplicação segura, é imprescindível citar o *Open Web Application Security Project* (OWASP) ou, em português, Projeto Aberto de Segurança em Aplicações Web, que é uma comunidade *online* sem fins lucrativos que disponibiliza material referente à segurança de aplicações *web.*

Este material compreende *softwares open source*, palestras, metodologias e material escrito para que pesquisadores, programadores, administradores de redes e quaisquer pessoas que tenham interesse no tema possam se informar sobre segurança em aplicações *web*.

A cada quatro anos, o OWASP lança uma pesquisa referente às principais vulnerabilidades *web* exploradas na *Internet*. O último lançamento (*release*, em inglês) foi em 2021, cuja lista é apresentada na Figura 1.

Figura 1 - Principais Vulnerabilidades em Sistemas Web

Fonte: OWASP.org , 2021

A Secretaria de Governo Digital do Ministério da Economia (SGD) desenvolveu o Guia de Segurança em Aplicações Web 1.0, tendo como objetivo o auxílio aos profissionais de desenvolvimento e manutenção de aplicações, no âmbito da implementação de software seguro Nitto et. al (2021, p 7), tema que entrou ainda mais em evidência com o estabelecimento da Lei Geral de Proteção de Dados (LGPD).

O documento define dois requisitos para a proteção de aplicações web: gerenciamento de ambiente e proteção do perímetro das aplicações. O gerenciamento de ambiente está

relacionado principalmente à manutenção, não apenas da aplicação, mas também das tecnologias que auxiliam no seu funcionamento.

Para um *software* funcionar, há várias tecnologias integradas além de sua própria linguagem de programação. Estas tecnologias sofrem atualizações, sendo preciso que tais atualizações sejam aplicadas às suas versões em produção. Isso porque *v*ersões antigas de *softwares* geralmente possuem vulnerabilidades que foram descobertas e que podem vir a ser exploradas, colocando em risco a aplicação *web* em produção.

Segundo Nitto et. al (2021, p 10),

> Patches de segurança e novas versões são lançadas frequentemente para corrigir as vulnerabilidades que são descobertas ao longo dos ciclos de vida das tecnologias. Por isso, é essencial monitorar as vulnerabilidades das tecnologias utilizadas pela aplicação, aplicar patches ordenados e oportunos em todos os sistemas afetados, bem como utilizar versões seguras.

No que se refere ao segundo ponto do Guia de Segurança em Aplicações Web, intitulado proteção do perímetro da aplicação, o guia aponta a necessidade de se realizar uma delimitação entre a rede interna em que se encontra a aplicação e a Internet ou até mesmo outra rede interna. Neste cenário, estarão presentes *firewall*, *proxy* e outros tipos de serviços para filtragem de pacotes na rede.

De acordo com Nitto et. al (2021, p 10),

> A proteção do perímetro da aplicação consiste na aplicação de controles e regras para o tráfego que flui entre a fronteira da rede interna e externa. Esses controles de segurança visam bloquear o tráfego de rede suspeito ou malicioso, limitar o acesso às aplicações somente a endereços confiáveis e necessários, registrar informações sobre os pacotes de rede que atravessam a fronteira, de modo a identificar possíveis incidentes de segurança, impedindo-os ou reportando-os ao responsável pela segurança da aplicação.

Esta proteção do perímetro da aplicação pode caracterizar-se como uma *Demilitarized Zone* (DMZ) ou, em português, Zona Desmilitarizada, em que parte da rede de uma empresa encontra-se isolada das demais redes internas, cujo acesso passa por um *Firewall* para filtragem de pacotes.

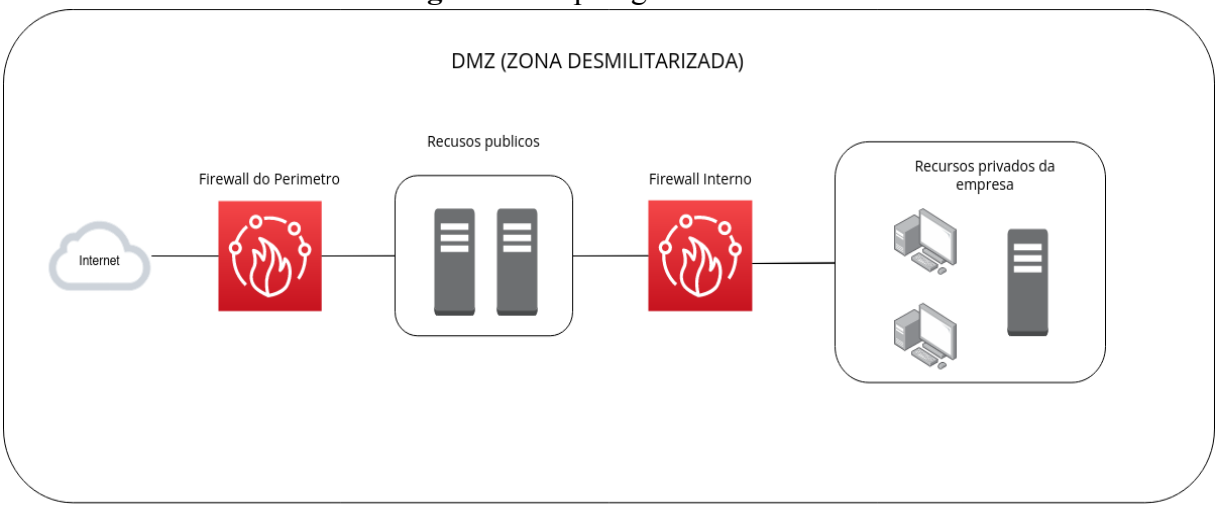
Acácio et al. (2020, p 4) explicam:

> A abordagem da implementação de zonas desmilitarizadas possibilita a conexão segura de redes com distintas políticas de segurança. De modo que o firewall, que

está no limite entre a zona corporativa e a desmilitarizada , não permite o tráfego de informações impróprias a rede corporativa.

Isolar serviços (recursos públicos cujo acesso externo é necessário) atrás de uma zona desmilitarizada é uma estratégia válida para aumentar a segurança da rede local (recursos privados da empresa), conforme ilustra a Figura 2. Uma aplicação *web*, hospedada em um serviço *web*, é um exemplo de aplicação isolada da rede local, mas acessível a ela e a usuários externos a partir da Internet.

Figura 2 - Topologia Com Dmz



Autor: Adaptado do artigo: o que é uma DMZ (Duarte, 2020)

Além destes requisitos gerais, o Guia de Segurança em Aplicações Web 1.0 define requisitos específicos baseados no Guia de Melhores Práticas de Codificação Segura OWASP, conforme Quadro 01.

QUADRO 01 - REQUISITOS ESPECÍFICOS DE SEGURANÇA

Requisitos	Detalhes
Validação dos dados de entrada	Consiste em auditar os dados inseridos em sua aplicação, evitando que os usuários enviem cadeias de caracteres que possam ser identificadas como código e executadas no servidor.
Codificação de dados de saída	Com a diversidade de arquiteturas modernas de aplicações web, a realização da codificação de

	saída, ou seja, a padronização dos dados enviados para outra aplicação, realizando previamente o tratamento de caracteres especiais que poderiam ser interpretados inadequadamente pela aplicação que está recebendo esses dados, é muito importante. O uso de APIs se torna mais constante, portanto, consultas parametrizadas, estruturas de modelagem com escape automático ou codificação de saída cuidadosamente escolhida é fundamental para a segurança da aplicação.
Autenticação e gerenciamento de credenciais	Esse requisito está associado ao requisito de gerenciamento de sessões e ao de controle de acesso em aplicação, garantindo a autoria de quem realiza determinadas ações e a certeza de que apenas um usuário autorizado possa realizar tais ações.
Gerenciamento de sessões	Praticamente toda sessão de usuário é implementada por meio de um *token* que identifica a sessão e que é concedido após o usuário se autenticar. Os principais problemas surgem a partir da forma como o *token* é criado – o que pode permitir a um atacante adivinhar o *token* de outros usuários. Daí a importância do planejamento de como serão gerenciadas as sessões dos usuários.
Controle de acesso	Uma das principais etapas de tratamento de acesso do usuário é averiguar se cada solicitação individual de acesso deve ser permitida ou negada. É necessário garantir que cada usuário só tenha acesso ao que lhe foi permitido no sistema.
Criptografia	As aplicações precisam de uma arquitetura de criptografia forte para proteger seus dados. Criptografar todas as informações de um sistema não é viável, mas optar por não usar criptografia pode expor os dados do sistema a riscos.
Tratamento de erros e *logs*	O objetivo do tratamento de erros e *logs* é fornecer informação útil para usuários, administradores e demais membros das equipes. Os *logs* devem ser de alta qualidade e claros, de forma a evitar a criação massiva deles.

Proteção de dados	É necessário que a aplicação proteja os dados tratados por ela, de forma que o acesso às suas informações se restrinja ao mínimo necessário.
Segurança nas comunicações	Para a transmissão de dados e informações, é ideal que se utilizem canais de comunicação seguros, o que pode ser implementado utilizando o protocolo TLS ou outra cifra forte.
Configuração do sistema	A configuração de uma aplicação recém lançada deve ser segura o suficiente para estar publicada na Internet, significando uma configuração segura por padrão, seguindo critérios para segurança e adotando metodologias para configuração como *Security by Default*[1].
Segurança em Banco de Dados	É fundamental que as organizações consigam garantir a proteção dos dados armazenados em banco de dados contra acessos indevidos, ações de *hackers* e incidentes técnicos.
Gerenciamento de Arquivos	Constantemente, as aplicações recebem arquivos enviados pelos usuários. Por isso é necessário que esses arquivos sejam tratados para evitar que usuários mal-intencionados enviem arquivos maliciosos que possam executar código do lado do servidor.
Gerenciamento de memória	É necessário que os desenvolvedores garantam que a aplicação faça o bom gerenciamento dos recursos de memória, a fim de evitar que atacantes utilizem técnicas que possam impactar na sua utilização, tais como o *Buffer Overflow*.
Práticas Gerais de Codificação	Seguir práticas gerais para codificação segura desde o início do desenvolvimento do projeto para evitar danos futuros aos produtos da empresa.

Fonte: Adaptado de Nitto, et. al. (2021)

Apesar da existência destes projetos, comunidades e guias de definição de parâmetros para a elaboração de aplicações seguras, bem como para a manutenção desta segurança, os

[1] *Security By Default*
lt: conceito que estabelece que as configurações padrão do software precisam ser as configurações mais seguras possíveis.

trabalhos apresentados na seção a seguir apontam a existência de vulnerabilidades nas aplicações analisadas.

2.3 TRABALHOS RELACIONADOS

Esta seção apresenta dois trabalhos relacionados ao tema de pesquisa deste Trabalho de Conclusão de Curso.

O primeiro trabalho analisado, intitulado Vulnerabilidades em Sistemas Web, foi desenvolvido por Oliveira (2019) e teve como objetivo investigar as principais formas de ataques, ameaças e vulnerabilidades em sistemas Web, tomando como cenário de testes o *site* www.manacialdasaguas.com.br. Para o desenvolvimento do trabalho, foram empregadas as ferramentas *Nikito* e *SkipFish*. A princípio, foi utilizada a ferramenta *skipfish* para a realização de experimentos por um período de aproximadamente 30 minutos e 54 segundos, buscando possíveis falhas. Durante este período, foram escaneados 277 arquivos, sendo reportados pela ferramenta 05 alertas relacionados a vulnerabilidades de nível médio e 03 relacionados a vulnerabilidades de nível baixo. Após esta primeira análise, foram realizados experimentos com a ferramenta *Nikito*, onde o período de varredura foi de 23 minutos e 20 segundos. Foram escaneados 6544 itens, não sendo reportadas falhas pela ferramenta. Como resultado, o autor indicou dois pontos de falha na segurança da aplicação web analisada: 1) um formulário de contato em HTML, a partir do qual um código malicioso pode ser inserido e enviado ao servidor da aplicação; 2) uma página de localização que permitiria a injeção de código malicioso a partir do *Javascript* presente no *site*.

O segundo trabalho analisado foi desenvolvido por Aparecido (2014) e tem como título “Estudo e Análise de Vulnerabilidades Web”. Este trabalho teve como objetivo a exploração passiva de 10 sites brasileiros de diferentes categorias, sendo elas: comércio eletrônico, religiosos, acadêmicos, grandes portais, sítios que utilizam gerenciadores de conteúdo (do inglês, *Content Management System* ou CMS), governamentais, regionais e de conteúdo adulto.Com os resultados, buscou-se realizar uma investigação mais detalhada sobre as principais vulnerabilidades *Web.* Após essa fase, as vulnerabilidades foram analisadas em

um ambiente controlado, sendo propostas contramedidas às mesmas. A busca das vulnerabilidades foi orientada pelo *OWASP Top Ten* do ano de 2013, tendo como objetivo identificar uma ou mais vulnerabilidades desta lista nas aplicações analisadas. Para isso, foram utilizados *scanners* como o Vega e o W3AF, além de testes práticos para explorar falhas, incluindo testes mais intrusivos como injeção de *scripts* para análise de resultados. Como resultados, o autor apontou que 33% das vulnerabilidades encontradas nos sites analisados eram classificadas como severas e de fácil exploração, conforme definições do OWASP. Desta forma, Aparecido (2014) concluiu que a maioria das vulnerabilidades web resultam de falhas durante o desenvolvimento de *software*. Estas falhas, segundo o autor, podem ser exploradas facilmente por pessoas com pouco conhecimento técnico utilizando-se ferramentas livres. Para mitigar estes problemas, o autor sugere, dentre outras medidas, o estudo e exploração de cada falha descrita no *Top Ten OWASP* em trabalhos distintos, bem como o estabelecimento de métricas de segurança que auxiliem desenvolvedores e mantenedores de aplicações web na gestão de riscos à segurança.

3 PROCEDIMENTOS METODOLÓGICOS

O presente trabalho consiste em uma pesquisa qualitativa de natureza aplicada, utilizando-se, para tanto, dos procedimentos de pesquisa bibliográfica e estudo de caso. A pesquisa bibliográfica foi realizada em livros, artigos e monografias disponíveis em repositórios digitais como Google Acadêmico e *ACM Digital Library*.

Para a realização do estudo de caso, foi escolhida uma aplicação web do Centro de Inovações Tecnológicas do Rio Grande do Norte (CINTE-RN) como objeto de análise de vulnerabilidades, bem como o servidor de hospedagem que a mantém.

Tal análise foi realizada utilizando-se uma abordagem que simula a fase de reconhecimento de um *pentest* do tipo "caixa preta". Sobre o termo, GIAVAROTO e SANTOS (2013, p. 25)explicam que a:

> Definição no português de caixa preta caracteriza a falta de conhecimento prévio de toda a infraestrutura do sistema-alvo que será testado. Portanto, é necessária a pormenorização de todos os dados analisados, a fim de determinarmos a sua localização e dimensão dos sistemas e aplicativos envolvidos, antes de podermos aplicar as técnicas de análises pretendidas ao sistema-alvo.

No contexto deste trabalho, significa que a aplicação será analisada apenas com base nos resultados obtidos a partir das ferramentas utilizadas na realização das varreduras, sendo o cenário (tipo de sistema operacional, servidor de hospedagem, tecnologias utilizadas pela aplicação, endereços IPs e demais informações) inicialmente desconhecido para o pesquisador.

As ferramentas escolhidas para a realização da análise de vulnerabilidades foram os *scanners* NMAP (*Network Mapper* ou, em português, Mapeador de Rede) versão 7.80 e Nessus versão 10.0.1. O motivo da escolha destas duas ferramentas se deu por serem amplamente conhecidas, aprovadas e utilizadas por profissionais da área de segurança e perícia forense computacional, sendo o NMAP utilizado para detecção de portas abertas, nome e versão de serviços e sistemas operacionais em execução na(s) máquina(s) analisada(s), enquanto o Nessus é uma ferramenta utilizada para obter dados mais detalhados

a respeito das vulnerabilidades da aplicação web e quais as possíveis consequências, caso essas vulnerabilidades sejam exploradas por terceiros.

Cabe salientar que o estudo de caso foi realizado visando verificar a existência de vulnerabilidades na aplicação web do CINTE-RN e no serviço que a hospeda, suscitar uma reflexão sobre a importância da segurança tanto nas etapas de desenvolvimento quanto na hospedagem de uma aplicação web, não tendo sido realizadas incursões no sistema que pudessem ocasionar problemas à sua utilização.

4 ESTUDO DE CASO

Este capítulo apresenta o cenário, descrição e resultados obtidos a partir da realização de um estudo de caso que consistiu na análise de vulnerabilidades do website da empresa CINTE-RN. As seções 4.1 e 4.2 descrevem o cenário de realização do estudo de caso (informações sobre a aplicação alvo e especificações técnicas sobre a máquina de origem da varredura); a seção 4.3 apresenta os resultados e discussões gerados a partir da análise de vulnerabilidades.

4.1 APLICAÇÃO ALVO

A aplicação a ser analisada é um *web site* empresarial desenvolvido com o intuito de ser utilizado para apresentação dos serviços do Centro de Inovações Tecnológicas (CINTE), um Provedor de Serviços de Internet (do inglês, *Internet Service Provider* ou ISP) que fornece serviços de acesso à *Internet* para clientes residenciais e empresas privadas há mais de 10 anos, atendendo Natal e outras 23 cidades do interior do estado.

O *site* também fornece uma "Área do Cliente" reservada para acesso e *download* de boletos para pagamento de faturas. Isso é feito através da plataforma, havendo a necessidade de que o cliente já possua um cadastro na empresa e que realize o *login* para ter acesso às

informações de suas faturas. A Figura 3 apresenta a imagem da página inicial do *Web Site* da empresa.

Figura 3 – Página Inicial cinte.com.br

Fonte: cinte.com.br (2022)

4.2 MÁQUINA DE ORIGEM DA VARREDURA

A máquina utilizada para a realização da varredura foi um *notebook* de marca Dell *Inspiron* i5 com processador de sétima geração, 8 *GB* de memória RAM e SSD de 240 *GB* de armazenamento conectado à Internet via rede sem fio com velocidade de 50 Mbps. O sistema operacional utilizado foi o Ubuntu 20.04 LTS.

Os testes foram realizados remotamente, ou seja, a máquina de origem dos disparos se encontrava fora da rede local da empresa, para simular um cenário o mais próximo possível de um ataque real.

Para o escaneamento, foram utilizadas as ferramentas Nessus e Nmap. Além dos programas citados, também foram utilizadas ferramentas de resolução de nomes como *whois*, *dig* e *nslookup* para testar hipóteses originadas a partir dos resultados dos escaneamentos ou varreduras.

4.3 ANÁLISE DE VULNERABILIDADES

Esta seção apresenta os resultados e discussões a partir da execução de uma análise de vulnerabilidades na aplicação web da empresa CINTE-RN. Para tanto, ela está dividida em duas partes.

A primeira apresenta os resultados do uso da ferramenta *nmap,* portanto, é mais focada em uma análise de vulnerabilidades do servidor de hospedagem da aplicação. Nesta primeira parte também foram utilizadas ferramentas de resolução de nomes.

A segunda parte apresenta os resultados obtidos a partir do uso da ferramenta Nessus, portanto, é voltada à análise de vulnerabilidades a nível de código, ou seja, da aplicação web.

4.3.1 No Servidor de Hospedagem

Inicialmente, foram realizados testes utilizando a ferramenta Nmap via linha de comandos para permitir uma análise da segurança do servidor onde a aplicação web da CINTE - RN está hospedada. O uso desta ferramenta segue o padrão:

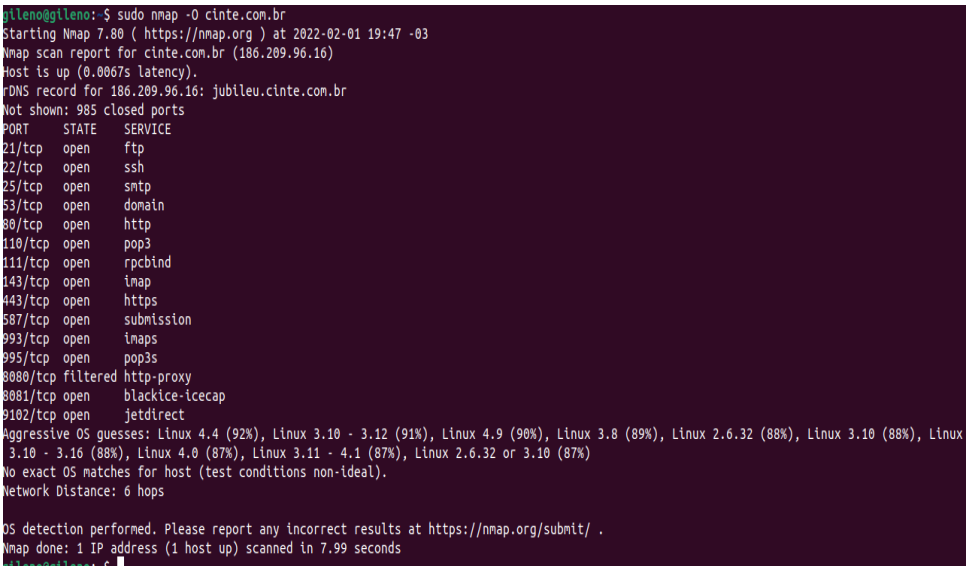
nmap [Opções] {especificação do alvo}

Onde *nmap* é o comando utilizado para o mapeamento da rede/serviços, *[Opções]* são os parâmetros que podem ser utilizados para especificar quais informações são esperadas

como resultado e *{especificação do alvo}* deve ser substituído pelo endereço IP ou nome de domínio do alvo da varredura.

No primeiro teste com a ferramenta nmap, foi utilizada a flag -O que, além de escanear as portas, também intenta determinar o sistema operacional que está operando no *host,* conforme pode ser observado na Figura 4.

Figura 4- Scan com flag -O



Fonte: Autoria própria (2022).

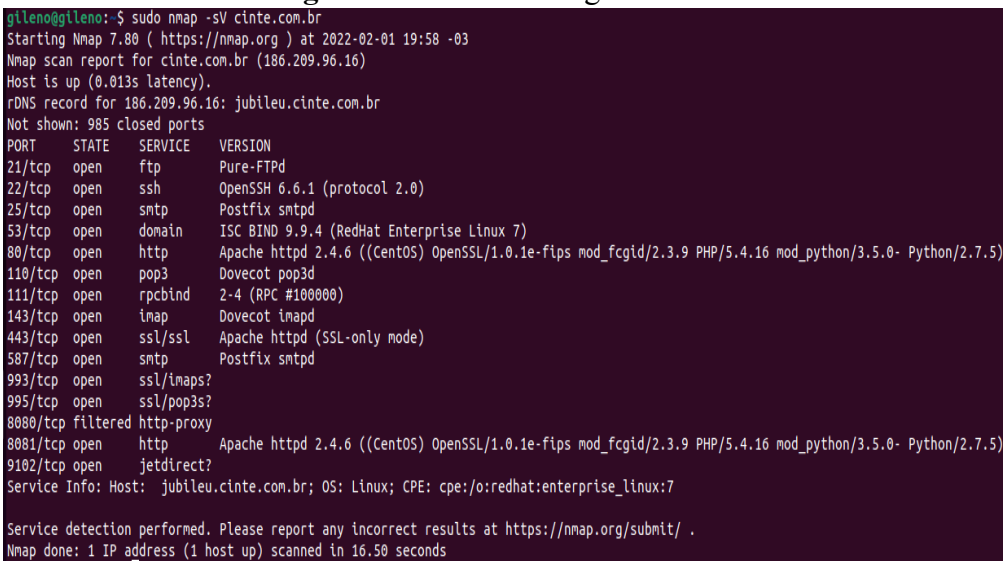
O teste não obteve sucesso na descoberta do sistema operacional do *host,* conseguindo apenas indicar, com 92% de certeza, que a versão do Kernel Linux utilizada é a 4.4. Apesar disso, um atacante poderia utilizar esta informação para tentar explorar possíveis vulnerabilidades existentes nesta versão de kernel.

No segundo teste, foram usadas as opções -sV para realizar um escaneamento das portas abertas e identificar os serviços e versões que estavam em funcionamento na máquina alvo.

Como resultado, observou-se que vários serviços estavam ativos (Figura 5), dentre eles, o serviço de hospedagem de site (dado o uso do protocolo HTTP – *Hypertext Transfer*

Protocol ou, em português, Protocolo de Transferência de Hipertexto); o serviço de resolução de nomes (indicado pelo termo *domain*, presente na coluna *Service*, e pelo termo ISC BIND na coluna *Version*); o serviço de acesso remoto (indicado pelo uso do protocolo SSH ou *Secure Shell*, que em português significa Shell Seguro), o serviço de transferência de arquivos (dado o uso do protocolo FTP ou *File Transfer Protocol*, que em português significa Protocolo de Transferência de Arquivo) e o serviço de correio eletrônico (dado o uso dos protocolos SMTP ou *Simple Mail Transfer Protoco*l – Protocolo de Transferência de Correio Simples, POP3 ou *Post Office Protocol* – Protocolo de Correios e, por fim, IMAP ou *Internet Message Access Protocol* – Protocolo de Acesso a Mensagens na Internet).

Figura 5 - Scan com flag -sV



Fonte: Autoria própria (2022).

Assim como a versão do kernel, os nomes e versões de protocolos/serviços instalados na máquina constituem uma fonte de recursos para atacantes. Moreno (2015, p. 58) explica que estas informações podem ser exploradas (a partir da identificação de vulnerabilidades de cada versão de software descoberta) por criminosos virtuais para a realização de ataques.

Uma vez que um ataque seja bem-sucedido e o acesso ao servidor seja concedido, o atacante pode realizar outras tentativas de ataques à rede local a partir do servidor invadido ou

realizar ações maléficas como a desfiguração da página hospedada, furto de dados confidenciais de usuários ou clientes da empresa.

Para além da problemática da exposição das versões, o excesso de serviços disponíveis no servidor de hospedagem do website da empresa CINTE suscitou o seguinte questionamento: além dos protocolos e serviços necessários à hospedagem de sites (como HTTP e SSL, por exemplo), qual a necessidade dos demais protocolos e serviços presentes neste servidor?

Para responder a esta pergunta, foi feita uma análise dos demais protocolos e serviços presentes no servidor. Como primeiro caso há o protocolo POP3 que possui uma função que permite que um cliente baixe e-mails do servidor para sua máquina, além de permitir a exclusão de mensagens (Heinlein e Hartleben, 2008, p.23).

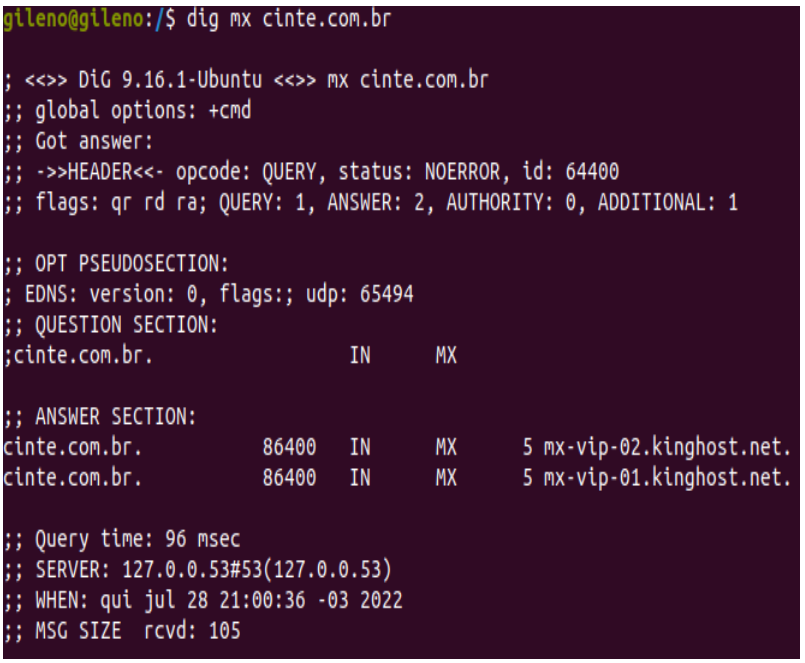
O IMAP é outro protocolo para e-mail, com funções adicionais aos do POP3, como a possibilidade do gerenciamento de e-mails de múltiplos usuários (Heinlein e Hartleben, 2008, p.28).

Ambos os serviços supracitados estão relacionados a manipulação e gerenciamento de mensagens de e-mail, não sendo necessários a uma máquina que não tenha esta finalidade, como no caso do servidor de hospedagem do site da CINTE.

Outro protocolo relacionado ao correio eletrônico e disponível no servidor de hospedagem analisado é o SMTP, responsável pela transferência de e-mails entre os servidores de correio eletrônico, como Thunderbird (Heinlein e Hartleben, 2008, p.17). A partir desta informação, surgiu o interesse em investigar se a empresa CINTE-RN utilizava de fato um servidor de correio eletrônico.

Para descobrir isso, foi utilizado o comando *dig* (*Domain Information Groper*) (INTERNET SYSTEMS CONSORTIUM, 2010). Como se pode observar na Figura 6, consultas feitas aos registros mx (*mail exchange*) do domínio cinte.com.br indicaram a existência de dois servidores de correio eletrônico mantidos para servir à empresa CINTE: **mx-vip-02.kinghost.net.** e **mx-vip-01.kinghost.net**. Estes servidores são mantidos pela empresa Kinghost, como se pode observar a partir de seus nomes de domínio.

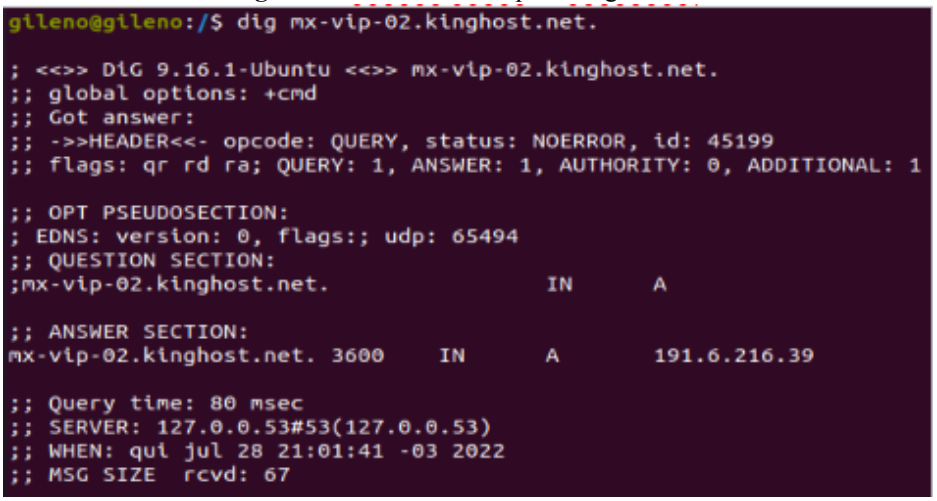
Figura 6 - Registros MX para o domínio da empresa CINTE



Autor: Autoria Própria (2022)

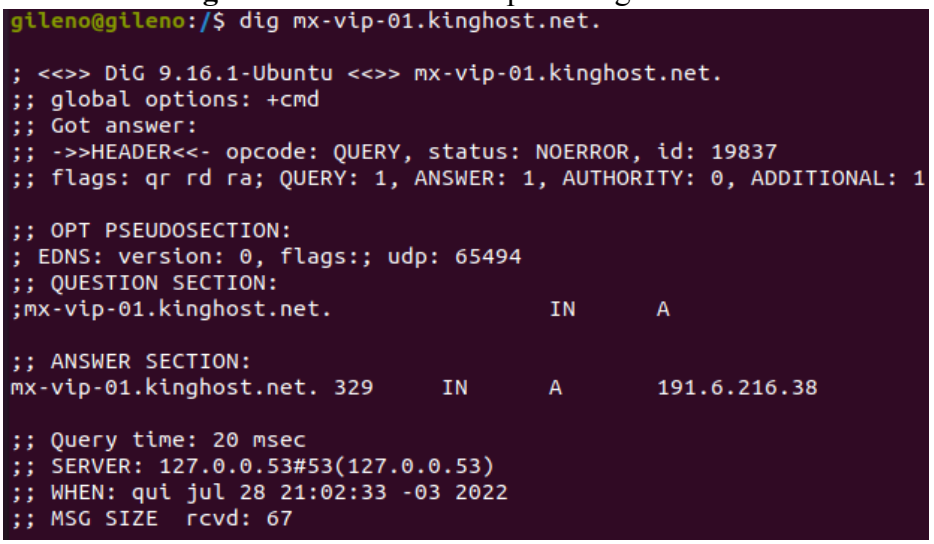
Ao utilizar o comando dig novamente, especificando tais nomes de domínio, foi possível identificar os endereços IPs atribuídos a cada um dos servidores, sendo eles 191.6.216.39 (associado ao domínio **mx-vip-02.kinghost.net**, conforme Figura 7) e 191.6.216.38 (associado ao domínio **mx-vip-01.kinghost.net**, conforme Figura 8).

Figura 7 - Dominio mx-vip-02.kinghost.net



Autor: Autoria Própria (2022)

Figura 8- Dominio mx-vip-01.kinghost.net



Autor: Autoria Própria (2022)

Conforme pode ser observado na Figura 4 e na Figura 5, o endereço IP do servidor que hospeda o website da CINTE é 186.209.96.16, ou seja, o serviço de correio eletrônico instalado neste servidor não está sendo utilizado pela empresa, já que os nomes de domínio do tipo *mx* registrados apontam para endereços IPs diferentes do servidor da CINTE.

Os fatos anteriormente relatados comprovam que não é necessária a utilização do serviço de correio eletrônico instalado no servidor de hospedagem do *website* da CINTE, uma vez que o servidor de correio eletrônico utilizado de fato pela empresa é mantido por uma empresa terceirizada, denominada Kinghost.

A situação descrita anteriormente se repete para o serviço Bind, ativo no servidor que hospeda o site da CINTE. Bind é o nome do serviço de resolução de nomes (responsável por implementar o DNS - *Domain Name System* ou Sistema de Nomes de Domínio) no mundo GNU/Linux (INTERNET SYSTEMS CONSORTIUM, 2022).

Com a utilização do comando *whois* (D'ITRI, 2019), o qual permite obter informações de contato e DNS utilizados por entidades da Internet, foi possível identificar que a máquina analisada não possui a função de servidor DNS para a empresa, tal como explicitado na Figura 9.

Figura 9 - Resultado de consulta utilizando whois

Autor: Autoria Própria (2022)

Os servidores responsáveis pela tradução de nomes e IPs da empresa CINTE são ns1.cinte.com.br e ns2.cinte.com.br, os quais possuem os respectivos endereços IPv4: 186.209.96.10 e 186.209.96.11. Nenhum deles corresponde ao endereço IPv4 do servidor que hospeda o website da CINTE (186.209.96.16), portanto o serviço de nomes instalado neste servidor também é desnecessário.

Com relação ao serviço FTP (serviço de transferência de arquivos), este pode estar instalado por dois motivos: para atualizações no código da aplicação hospedada ou para a transferência de arquivos que irão compor ou ser disponibilizados pela aplicação hospedada.

Com relação às atualizações no código do website hospedado, estas podem ser implementadas através do uso do GitHub (disponível em https://github.com/) ou outro sistema para versionamento de código, em conjunto com a configuração de chaves públicas para acesso e atualização do repositório local do servidor. Esta é uma solução segura e alternativa ao uso do FTP para a atualização do código da aplicação.

Com relação à transferência de arquivos para compor ou serem disponibilizados pela aplicação web hospedada, pode-se optar pelo uso do SSH – que inclusive já está instalado no

servidor *web* da CINTE – e utilizar o *security copy protocol* (scp) para fazer a transferência de arquivos entre máquinas de forma segura.

Cardim (2020, p. 5) explica:

> Para reduzir os riscos de exposição, o protocolo SSH pode ser utilizado para substituir serviços de emulação de terminal, como Telnet, e transferência de arquivos, como FTP. Sua implementação garante segurança adicional quando dados são transferidos de forma criptografada. Porém, para isso é importante que o serviço de SSH esteja sempre atualizado e na versão mais recente do produto.

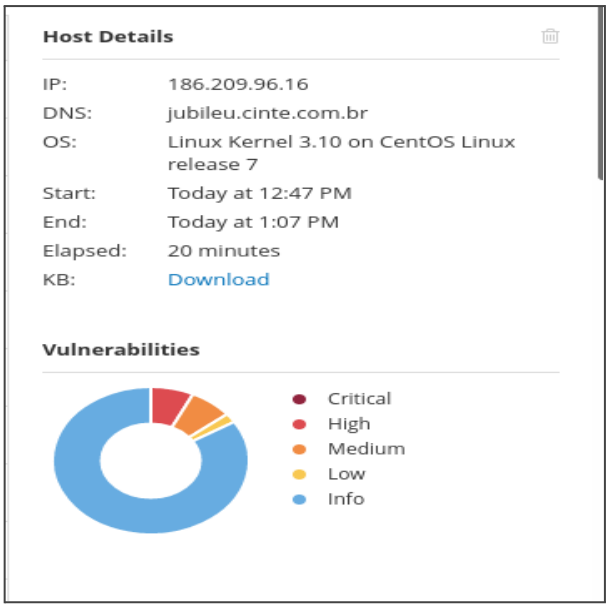
Desta forma, o FTP poderia ser descartado, evitando o acúmulo de mais um serviço no servidor *web* da CINTE e diminuindo as chances de ataques a serviços possivelmente mal configurados, executando em versões antigas e/ou com vulnerabilidades exploráveis.

4.3.2 No Website da CINTE-RN

Uma vez finalizada a análise com a ferramenta Nmap, foi iniciada a análise com o auxílio do Nessus, um *scanner* de aplicações muito utilizado no meio corporativo.

Ao realizar o escaneamento da aplicação *web* da empresa CINTE - RN, o qual durou 20 minutos, o Nessus apresentou o gráfico presente na Figura 10, com a porcentagem de vulnerabilidades encontradas por categoria, sendo: 5% de nível crítico (11), 16% consideradas de nível médio (34), 3% de nível baixo (7) e 76% de ocorrências de informativos (164).

Figura 10 - Resultado do teste utilizando a ferramenta Nessus



Autor: autoria própria (2022)

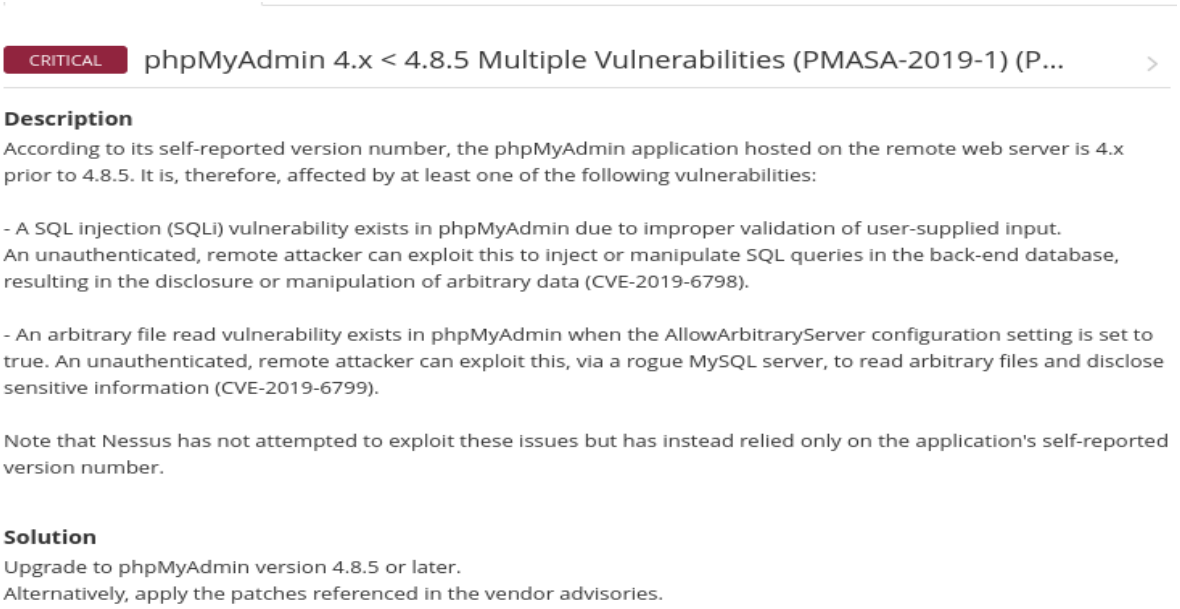
Como se pode observar, a maioria das vulnerabilidades encontradas são do tipo Info (informativos), ou seja, são pouco ou nada prejudiciais ao sistema. No entanto, a existência de vulnerabilidades das categorias média e crítica indicam a existência de ameaças que podem causar prejuízos ao sistema, caso estas vulnerabilidades sejam exploradas por um atacante.

As vulnerabilidades categorizadas como média e/ou crítica podem ser evitadas com a aplicação de *patchs*, ou atualização de bibliotecas, conforme relatório apresentado pela ferramenta Nessus. Por isso, foram escolhidas quatro vulnerabilidades (SQL Injection, CSRF, exposição de dados sensíveis através de um arquivo de debug e uma falha de configuração de HSTS), sendo uma categorizada como crítica e três de nível médio para apresentação neste trabalho, a fim de exemplificar vulnerabilidades que necessitam de configurações extras, além da simples atualização das versões das tecnologias utilizadas pela aplicação web.

Além da categorização das vulnerabilidades, também foi possível identificar, a partir dos resultados do Nessus, o sistema operacional em funcionamento na máquina alvo (CentOS 7) conforme Figura 10 — algo que o Nmap não conseguiu realizar.

Na categoria crítica, durante a varredura foi identificada uma vulnerabilidade presente no PHPMyAdmin e conhecida como *SQL Injection.* Caso seja explorada, esta falha de segurança permite que um atacante obtenha dados sensíveis do banco de dados, como nome de usuário e senha. A descrição da vulnerabilidade pode ser vista na Figura 11.

Figura 11 - PHPMyAdmin vulnerabilidade de *SQL Injection*



Autor: autoria própria (2022)

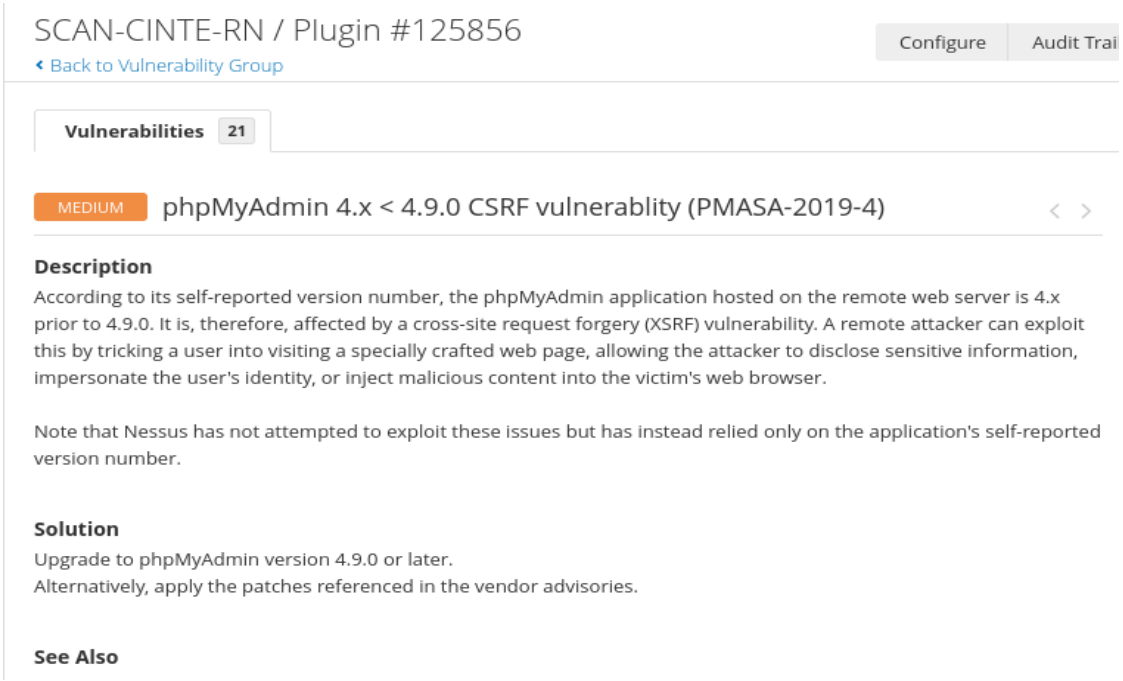
Segundo Alenezi, Nadeem e Asif (2021, p. 1021),

> "Nos últimos anos, as injeções de SQL surgiram como um dos tipos mais perigosos de ataques à sistemas web e são classificados em primeiro lugar entre os Open Web Application Security Project's (OWASP) top 10 vulnerabilidades [1]. Os ataques de injeção SQL (SQLIAs) são lançados inserindo caracteres maliciosos nos campos de entrada de aplicativos web, resultando em uma consulta SQL modificada."

Assim, uma falha de *SQL Injection* é uma violação à integridade dos dados do sistema, sendo prioridade para correção. A ferramenta Nessus recomenda a atualização do PHPMyAdmin para a versão 4.8.5 ou superior, ou a aplicação de *patches* para as correções.

Na categoria média, foi identificada outra vulnerabilidade na ferramenta PHPMyAdmin, conhecida como CSRF (*Cross-Site Request Forgery,* ou em português, falsificação de solicitações entre sites) conforme Figura 12.

Figura 12 - PHPMyAdmin vulnerabilidade de CSRF



Autor: autoria própria (2022)

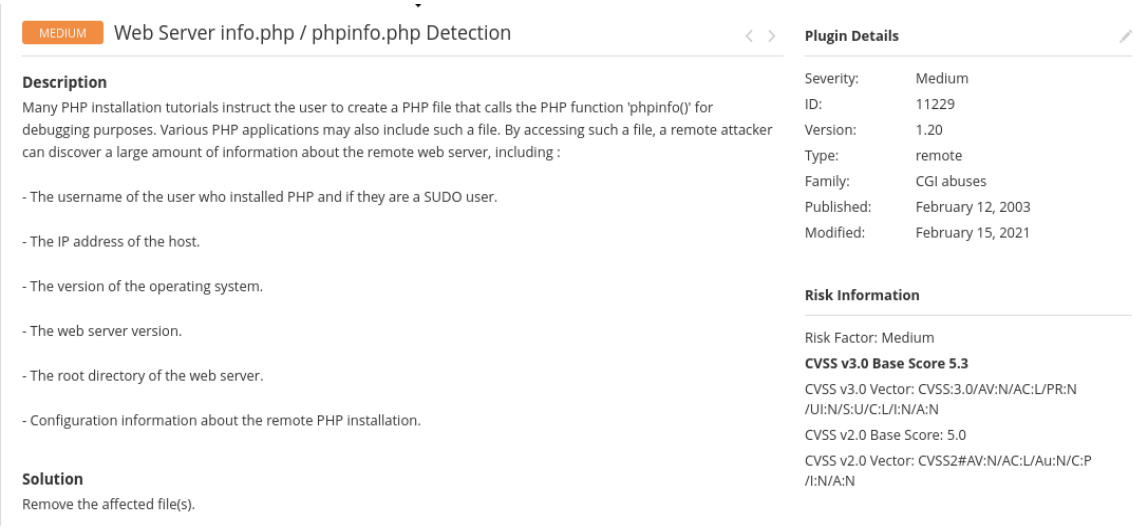
Segundo Chavarría (2018, p. 37), um ataque CSRF consiste em explorar sessões que um usuário abriu e não finalizou. A vítima costuma acessar um *site* malicioso que se utiliza de algum elemento que, ao ser carregado, realiza uma solicitação HTTP. Esta solicitação é encaminhada para outro sistema que realiza uma determinada ação, se passando pelo usuário.

O PHPMyAdmin é um Sistema de Gerenciamento de Banco de Dados (SGBD) que serve para realizar consultas e atualizações em bancos do tipo MySQL ao qual esteja conectado. Portanto, um ataque CSRF a um *software* que manipula o banco de dados da aplicação pode gerar danos ao funcionamento desta, possibilitando acesso a dados sensíveis que por ventura estejam armazenados neste banco.

Como pode ser observado na Figura 12, o Nessus sugere que para resolver este problema, seja feita uma atualização da ferramenta para a versão 4 ou superior, ou ainda que seja feita a aplicação de *patches* de correção.

A segunda vulnerabilidade da categoria média indicada pela ferramenta Nessus estava associada ao PHP que, assim como outras linguagens de programação, possui elementos para depuração de informações, como a função phpinfo (Figura 13).

Figura 13 - Identificação de arquivo phpinfo.php



Autor: Autoria Própria (2022)

The PHP Group (2022) explica que a função phpinfo():

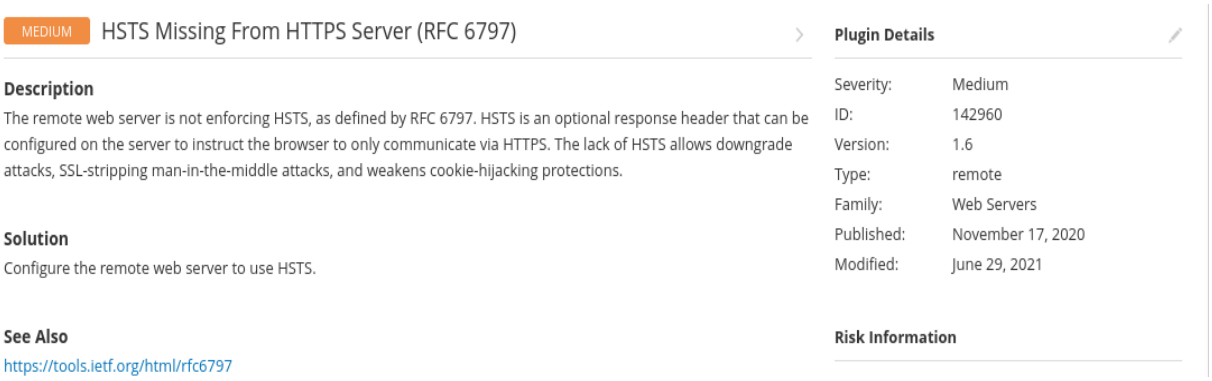
> Mostra uma grande quantidade de informações sobre o estado atual do PHP. Isto inclui informações sobre as opções de compilação do PHP e extensões, a versão do PHP, informações do servidor e ambiente (se compilado como um módulo), o ambiente PHP, informação da versão do SO, caminhos, valores principais e locais das opções de configuração, cabeçalhos HTTP e a licença do PHP.
> Devido a configuração em cada sistema ser diferente, a função phpinfo() é normalmente utilizada para conhecer as definições de configuração e as variáveis pré-definidas que estejam disponíveis no sistema. phpinfo() é também uma ferramenta valiosa para eliminação de erros já que contém todos os dados de EGPCS (Environment, GET, POST, Cookie, Server).

Dessa forma, a simples existência do arquivo phpinfo.php em um diretório acessível via *browser,* caso seja acessado, concederia uma grande quantidade de informações a possíveis atacantes, como o nome do usuário responsável pela instalação do PHP e se este pode utilizar o comando *sudo*, a versão da linguagem, as bibliotecas e módulos instalados, a versão do serviço web para a operação do *website*, dentre outras informações que podem

facilitar a realização de ataques ao servidor e à aplicação hospedada. Para evitar esse tipo de problema, o Nessus recomenda a remoção desse arquivo do sistema.

A terceira e última vulnerabilidade do tipo média que será abordada neste trabalho e que foi identificada no website da CINTE-RN consistiu em uma falta de configuração do *Strict-Transport-Security* (HSTS), que é um cabeçalho que pode ser habilitado no servidor para permitir apenas conexões seguras via HTTPS, como pode ser observado na Figura 14.

Figura 14 - Detecção de vulnerabilidade HSTS.



Autor: autoria própria (2022)

Esta opção estava desabilitada no servidor de hospedagem do *website* da empresa CINTE-RN, possibilitando problemas como o apresentado no portal MOZZINLA.org (2022):

> "Se um site aceitar uma conexão por meio de HTTP e redirecionar para HTTPS, os visitantes poderão se comunicar inicialmente com a versão não criptografada do site antes de serem redirecionados, se, por exemplo, o visitante digitar http://www.foo.com/ ou até mesmo apenas foo.com. Isso cria uma oportunidade para um ataque man-in-the-middle. O redirecionamento pode ser explorado para direcionar os visitantes a um site mal-intencionado em vez da versão segura do site original."

Como solução para esse problema, o Nessus sugere a configuração do servidor para o uso do HSTS.

5 SUGESTÕES PARA PREVENÇÃO DE ATAQUES NO SISTEMA ANALISADO

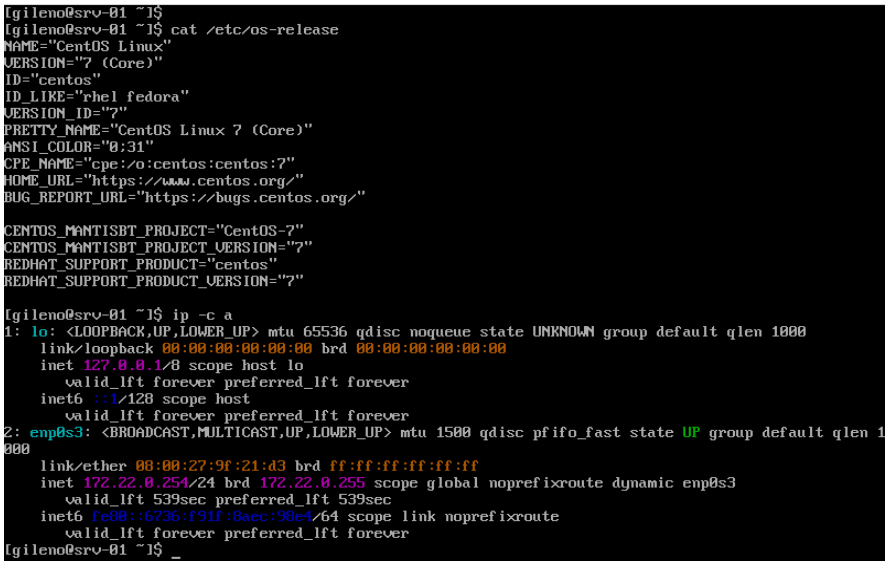
Após a análise dos resultados obtidos a partir do estudo de caso apresentado anteriormente, esta seção apresenta sugestões a fim de evitar possíveis tentativas de ataques.

5.1 NO SERVIDOR DE HOSPEDAGEM

Para começar a análise do ponto de vista do servidor, sugere-se a desinstalação dos protocolos e serviços desnecessários à hospedagem do website da CINTE-RN, conforme indicado anteriormente na seção 4.3.1, a fim de manter uma instalação mínima, preservando apenas os serviços indispensáveis para operação e manutenção da aplicação web.

Com relação às informações expostas sobre os serviços, identificadas pelo Nmap, este problema pode ser corrigido de forma simples, alterando parâmetros nos arquivos de configuração dos serviços. Para exemplificar estas configurações, foi elaborado um ambiente de testes utilizando-se o software VirtualBox para a criação de duas máquinas virtuais: uma que serviu como cliente e outra que fez o papel de servidor do cenário de simulação.

Figura 15 - Máquina SRV-01 (servidor)

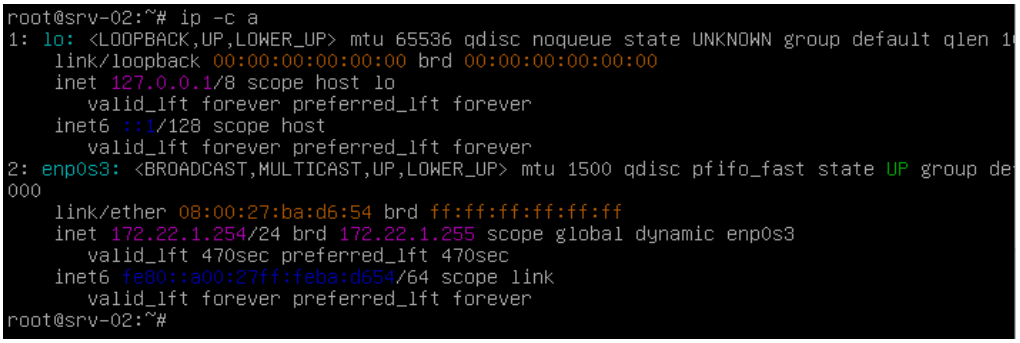


Autor: Autoria Própria (2022)

A máquina virtual que fez papel de servidor foi criada com 512M de memória RAM, 10.94 GB de armazenamento, sistema operacional CentOS 7 e endereço IP 172.22.0.254/24, conforme Figura 15. Esta máquina recebeu o *hostname* SRV-01 e foram instalados nela os serviços SSH e Apache2.

Para simular um atacante externo à rede, foi criada uma máquina virtual com sistema operacional Debian 11, 256 MB de memória RAM, 10 GB de armazenamento, endereço IP 172.22.1.254/24 e *hostname* SRV-02, conforme Figura 16.

Figura 16 - Máquina SRV-02 (atacante)



Autor: Autoria Própria (2022)

Antes de realizar as alterações nos arquivos, foi efetuado um teste no SRV01 a fim de analisar os resultados, conforme Figura 17.

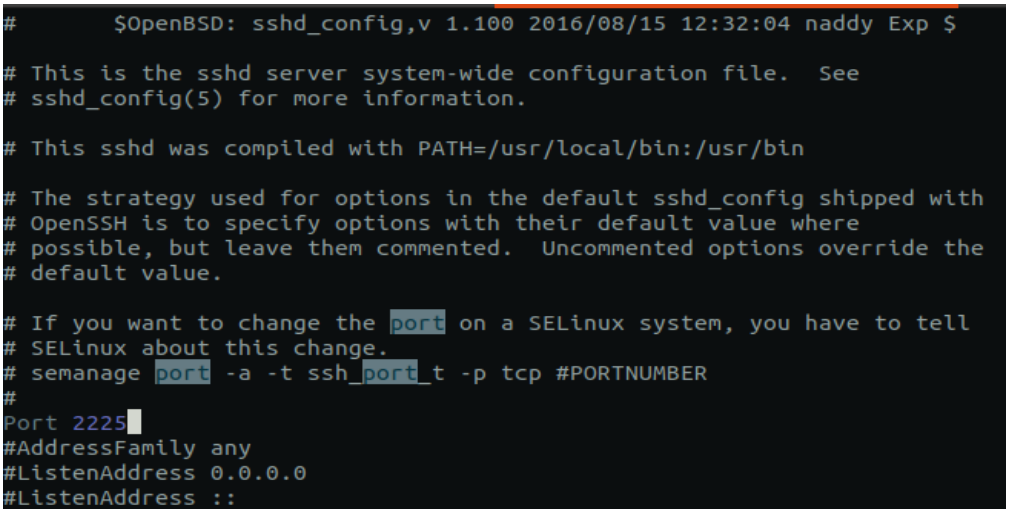
Figura 17 - Teste no Servidor sem Alterações

Autor: Autoria Própria (2022)

Como pode ser observado, as informações dos serviços encontram-se expostas ao realizar um teste com o Nmap, assim como no servidor real da aplicação analisada neste trabalho.

Para solucionar este problema, a primeira configuração realizada no servidor de simulação consistiu na alteração da porta padrão do serviço SSH (de 22 para 2225), conforme Figura 18. Esta alteração foi feita no arquivo de configuração denominado sshd_config, disponível no diretório /etc/ssh/.

Figura 18 - Alteração de Porta SSH



Autor: Autoria Própria (2022)

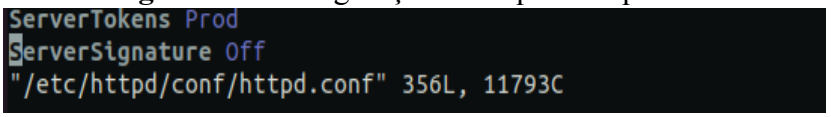
Além disso, é indicado que não se faça acesso ao servidor SSH com senhas, mas sim por meio de chaves públicas de acesso, que identificam as máquinas de cada usuário. Tais chaves são geradas através de um utilitário chamado ssh-keygen, responsável por gerar uma chave pública que será inserida no servidor e uma chave privada que identifica o usuário, sendo que qualquer máquina que não possua a chave privada relacionada à chave pública não conseguirá realizar *login* na máquina. DEVMEDIA (2013) explica:

> Diante da existência da possibilidade de um invasor conseguir descobrir sua senha, o SSH oferece um tipo de autenticação baseado em duas chaves. Isto melhora o nível de segurança, pois desta forma não dependemos apenas de uma senha para o login. Ao invés disso, teremos duas chaves: uma chave pública instalada no servidor SSH, e uma chave privada protegida por um *passphrase*. Com estas duas chaves será necessário saber o *passphrase* para estabelecer a conexão e ainda ter a chave privada instalada em seu computador.

No caso do serviço web, como a porta 443 e a porta 80 encontram-se liberadas para acesso, ainda é possível identificar sua versão.

Para evitar isso, é necessário desabilitar a apresentação desses dados no serviço web Apache, editando o arquivo httpd.conf. Este arquivo está localizado no diretório /etc/httpd/conf/httpd.conf. Nele, é preciso adicionar as diretivas: **ServerTokens Prod** e **ServerSignature Off**, conforme Figura 18.

Figura 19- Configuração no arquivo httpd.conf



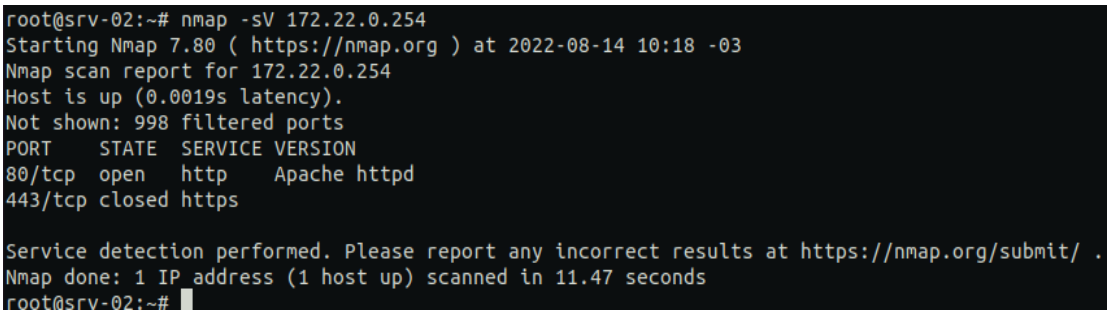
Autor: Autoria Própria (2022)

A primeira diretiva informa ao Apache para retornar apenas “Apache” no cabeçalho do servidor em todas as páginas solicitadas. Além desta, existem outras diretivas citadas na documentação do Apache com o mesmo objetivo, porém a **ServersTokens Prod** é a que mais omite informações.

A segunda diretiva – **ServerSignature Off** – evita que o serviço exiba sua versão em páginas de erro ou qualquer página que ele gere.

Com as configurações anteriores aplicadas, foi realizado um novo teste de varredura na máquina, obtendo-se o resultado apresentado na Figura 20.

Figura 20- Teste NMAP com serviços configurados



Autor: Autoria Própria (2022)

Como foi possível observar na Figura 20, o serviço SSH foi omitido da saída do comando Nmap devido à mudança de sua porta padrão para a porta 2225 (Figura 18). O serviço web continua funcionando e recebendo requisições na porta 80 (Figura 20), porém a versão do Apache já não aparece em virtude das configurações realizadas no arquivo httpd.conf, como apresentado na Figura 19.

Uma vez que as sugestões para a solução de problemas no servidor de hospedagem foram apresentadas, seguem as sugestões para mitigar possíveis tentativas de ataques a partir da exploração das vulnerabilidades encontradas na aplicação web do CINTE-RN.

5.2 NO WEBSITE DA CINTE-RN

Para solucionar os problemas relacionados a *SQL Injection* e CSRF, sugere-se realizar a atualização do PHPMyAdmin.

Além do PHPMyAdmin, sugere-se a atualização das demais tecnologias utilizadas no desenvolvimento e manutenção da aplicação web, como banco de dados, bibliotecas e linguagem do sistema, pois as vulnerabilidades presentes nas versões utilizadas são problemas já solucionados em versões atuais dessas tecnologias.

Para evitar que um atacante identifique o diretório onde se localiza o arquivo phpinfo.php, pode-se optar por removê-lo do servidor ou configurar o arquivo .htaccess que deve ficar localizado na raiz da aplicação, geralmente no apache sendo o diretório /var/www/html/. No arquivo deve ser inserido a seguinte diretiva:

```
<IfModule mod_rewrite.c>
    Options -Indexes
</IfModule>
```

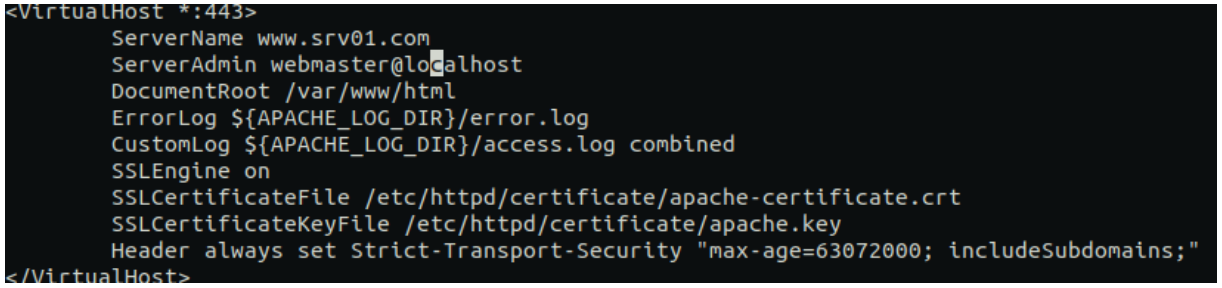
O .htaccess é um arquivo de configuração usado por servidores *web* para informar certos comportamentos ao servidor. Neste caso, a configuração indicada anteriormente impede a indexação de diretórios, o que evitaria o acesso a arquivos como o phpinfo.php e também a diretórios presentes no servidor via navegador.

Por fim, é necessário habilitar o transporte seguro de dados no servidor Apache, como indicado pelo Nessus, para solucionar o problema relacionado ao HSTS. Para isso, é preciso editar o arquivo 000-default.conf presente no diretório /etc/httpd/sites-available, adicionando a seguinte linha na seção da declaração do HTTPS:

Header always set Strict-Transport-Security "max-age=63072000; includeSubdomains;

A configuração do arquivo 000-default.conf pode ser observada na Figura 21.

Figura 21- Configuração no arquivo 000-default.conf



Autor: Autoria Própria (2022)

A adoção das medidas discutidas anteriormente diminui a visibilidade de atacantes em relação à infraestrutura de hospedagem da aplicação web e em relação às tecnologias de codificação utilizadas em seu desenvolvimento, dificultando a exploração de um serviço desatualizado ou mal configurado.

6 CONSIDERAÇÕES FINAIS

A negligência com a segurança em projetos de desenvolvimento de *software*, tende a gerar falhas que podem ser exploradas, tendo como consequência sérios problemas às empresas e usuários finais. Para tanto, existem procedimentos para verificação dessas vulnerabilidades que podem ser aplicados e amenizar tais riscos, seja no momento da codificação, optando por boas práticas no desenvolvimento, seja após a implementação do projeto, com a realização de testes de intrusão no sistema para validar sua robustez.

No cenário aqui abordado levou-se em consideração a aplicação web utilizada no CINTE-RN, implementando soluções plausíveis para correção de algumas das falhas identificadas durante o processo de análise de vulnerabilidades. Simultaneamente, levantou-se uma reflexão a respeito da importância da segurança da informação no ciclo de desenvolvimento de um *software*, utilizando de um exemplo real e apresentando vulnerabilidades tanto no sistema web, quanto no servidor utilizado para hospedagem.

Os resultados obtidos mostraram que a maioria das vulnerabilidades de nível crítico e médio poderiam ter sido resolvidas com a aplicação de *patches* de segurança ou atualização de bibliotecas. Com relação ao servidor web, configurações simples como a mudança de porta padrão e informações presentes no banner dos serviços contribuem para dificultar a vida de um possível atacante. Esses procedimentos relativamente simples, apresentados brevemente no capítulo 5, infelizmente muitas vezes são ignorados pelos desenvolvedores e administradores de sistemas.

Como trabalhos futuros, sugere-se a realização de um trabalho de conscientização junto à empresa CINTE-RN quanto à importância da segurança em seus sistemas, bem como um aprofundamento sobre este tema com trabalhos que incentivem e divulguem boas práticas de segurança na codificação de software e incentivem a realização de *hardening* em servidores.

REFERÊNCIAS

ACÁCIO, A. A. ; BECKER, L.; QUINTINO, F. L.; EDUARDO, J. **CRITÉRIOS DE SEGURANÇA PARA SISTEMAS SCADA EM REDE CORPORATIVA** . [*s. l.*], 2020. Disponível em: https://www.sba.org.br/open_journal_systems/index.php/cba/article/view/475. Acesso em: 7 abr. 2022.

Akamai. **API: a superfície de ataque que conecta todos nós**.[*s. l.*], v. 7, ed. 4, 2021.

ALENEZI, Mamdouh; NADEEM, Muhammad; ASIF, Raja. SQL injection attacks countermeasures assessments. **Indonesian Journal of Electrical Engineering and Computer Science**, [*S. l.*], p. 1121-1131, 2 fev. 2021. Disponível em: https://www.researchgate.net/profile/Mamdouh-Alenezi-2/publication/344597081_SQL_Injection_Attacks_Countermeasures_Assessments/links/5fcc5c6345851568d142b19a/SQL-Injection-Attacks-Countermeasures-Assessments.pdf. Acesso em: 23 mar. 2022.

MONTEVERDE, W. A. **ESTUDO E ANÁLISE DE VULNERABILIDADES WEB**. 2014. Trabalho de Conclusão de Curso (Tecnologia em Sistemas para Internet) - UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ, [*S. l.*], 2014. Disponível em: http://repositorio.utfpr.edu.br/jspui/bitstream/1/6476/3/CM_COINT_2013_2_02.pdf. Acesso em: 15 maio 2022.

CARDIM, Bruno. **GESTÃO DE SEGURANÇA EM DISPOSITIVOS IOTS**. Orientador: Ms. Jefferson Jesus Hengles Almeida. 2020. Trabalho de Conclusão de Curso (Engenharia Elétrica) - Universidade Presbiteriana Mackenzie, [*S. l.*], 2020. Disponível em: https://dspace.mackenzie.br/handle/10899/29255. Acesso em: 10 jul. 2022.

CAREY, M. CRISCUOLO, P. PETRUZZI, M. **Nessus Networking Auditing.** Segunda Edição. Burlington: Elsevier Inc, 2008.

CERTBR. Computadores. [*S. l.*], 20 jul. 2021. Disponível em: https://cartilha.cert.br/fasciculos/computadores/fasciculo-computadores-slides-notas.pdf. Acesso em: 26 jun. 2022.

GIAVAROTO, S. C. R. ; SANTOS, G. R. **BACKTRACK LINUX AUDITORIA E TESTE DE INVASÃO EM REDES DE COMPUTADORES**. Rio de Janeiro: Editora Ciência Moderna, 2013. Disponível em: http://rakuten.livrariacultura.com.br/imagem/capitulo/30757884.pdf. Acesso em: 7 out. 2022.

CHAVARRÍA, G.V. **Estudio de los ataques contra website. OWASP**. Palma: [*s. n.*], 2018. Disponível em: https://dspace.uib.es/xmlui/bitstream/handle/11201/151259/Memoria_EPSU0643.pdf?sequence=1&isAllowed=y. Acesso em: 15 fev. 2022.

DEVMEDIA. **SSH – Protocolo seguro para acesso remoto - Revista Infra Magazine 9**. [*S. l.*], 2013. Disponível em: https://www.devmedia.com.br/ssh-protocolo-seguro-para-acesso-remoto-revista-infra-magazine-9/26785. Acesso em: 4 out. 2022.

D'ITRI, M. whois(1) - Linux man page. 2019. Disponível em: https://linux.die.net/man/1/whois.

FERRAREZI, Guilherme de C.; GROSSI, Igor Rafael F. Del; MARCHI, Késsia Rita. Firewall IPTables e Exemplo de Implementação no Ambiente Corporativo. **Universidade Paranaense**, [*s. l.*], 2015. Disponível em: https://www.researchgate.net/publication/267803705_Firewall_IPTables_e_Exemplo_de_Implementacao_no_Ambiente_Corporativo. Acesso em: 14 ago. 2022.

FRANCO OLIVEIRA, Bruno. **Vulnerabilidades em Sistemas Web**. 2019. Trabalho de Conclusão de Curso (Bacharel em Sistemas de Informação) - UNIVERSIDADE FEDERAL DE UBERLÂNDIA, [*S. l.*], 2019.

HEINLEIN, Peer; HARTLEBEN, Peer. **The Book of IMAP**: Building a Mail Server with Courier and Cyrus. San Francisco: Open Source Press GmbH, 2008. Disponível em: https://books.google.com.br/books?id=YUUSdUtay1QC&printsec=frontcover&hl=pt-BR#v=onepage&q&f=false. Acesso em: 10 out. 2022.

INCIDENTES Reportados ao CERT.br -- Janeiro a Dezembro de 2020. [*S. l.*], 2020. Disponível em: https://cert.br/stats/incidentes/2020-jan-dec/tipos-ataque.html. Acesso em: 9 fev. 2022.

INTERNET SYSTEMS CONSORTIUM. dig(1)- Linux man page. 2010. Disponível em: https://linux.die.net/man/1/dig.

INTERNET SYSTEMS CONSORTIUM. BIND 9. Disponível em: https://www.isc.org/bind/. Acesso em: 27 set. 2022.

KORN. **Falha de segurança do Log4j pode afetar toda a Internet; o que você precisa saber.** [S.I] 31 nov 2021. disponível em: https://www.cnnbrasil.com.br/tecnologia/falha-de-seguranca-do-log4j-pode-afetar-toda-a-internet-o-que-voce-precisa-saber/

MORENO, Daniel. **Introdução ao pentest**. 1. ed. São Paulo: Novatec, 2015.

MONTANHEIRO, LUCAS SOUZA. **PRIMEIROS PASSOS PARA O DESENVOLVIMENTO SEGURO DE APLICAÇÕES WEB**. 2018. Trabalho de Conclusão de Curso (Gradução) - INSTITUTO FEDERAL GOIANO, [*S. l.*], 2018. Disponível em: https://repositorio.ifgoiano.edu.br/bitstream/prefix/1266/1/TCC-Montanheiro-v9-FichaOK%20lucas.pdf. Acesso em: 29 mar. 2022.

MOZZINLA.ORG. Strict-Transport-Security. *In*: **Strict-Transport-Security**. [*S. l.*], 2022. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Headers/Strict-Transport-Security. Acesso em: 27 mar. 2022.

nmap.org. **Introdução.** [S.I] 11 nov 2021. disponível em: https://nmap.org/

OWASP. **OWASP Top 10 2017.** [S.I].25 out 2021. disponivel em: https://owasp.org/www-pdf-archive/OWASP_Top_10-2017_%28en%29.pdf.pdf

PRESSMAN, ROGER S. **Engenharia de Software**. Quinta Edição. New York: The McGraw-Hill Companies, Inc, 2002.

Silva. **Segurança da Informação no Desenvolvimento de Aplicações Web.** [S.I] 20 fev 2022. disponível em: http://ric.cps.sp.gov.br/bitstream/123456789/916/1/20151S_SILVALuanMenezes_CD2169.pdf

STALLINGS, William. **Criptografia e Segurança de Redes Princípios e práticas**. 6. ed. [*S. l.*]: Pearson Education do Brasil Ltda, 2014. Ebook(p. 11)

The Apache Software Foundation. **Apache Log4j 2.** [S.I] 31 nov 2021. disponível em: https://logging.apache.org/log4j/2.x/

TREND Micro. **Log4jShell: Entenda a Falha e Veja Como Se Proteger.** [S.I] 31 nov 2021. disponível em: https://blog.trendmicro.com.br/log4jshell-entenda-a-falha-e-veja-como-se-proteger/?gclid=CjwKCAiAiKuOBhBQEiwAId_sKzTpoh4W2hPnfRctOyI9BWu8rGAoSSldc1Hts5nNtghm0-Y5nKMROhoC8AUQAvD_BwEs

THE PHP GROUP. Phpinfo. *In*: **Phpinfo**. [*S. l.*], 2022. Disponível em: https://www.php.net/manual/pt_BR/function.phpinfo.php. Acesso em: 16 jul. 2022.

MORENO, Daniel. Pentest: O que é Pentest. *In*: INTRODUÇÃO ao Pentest. São Paulo: Novatec, 2015.

APÊNDICE A – Permissão para testes no ambiente

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

Eu, [redacted] cujo o cargo é Gerente Técnico e CPF [redacted] estou ciente do estudo a ser feito no sistema de endereço **cinte.com.br** hospedado em minha empresa [redacted], cujos objetivos e justificativas são: a emissão de um relatório a respeito dos possíveis pontos vulneráveis na aplicação que está hospedada em um servidor sob meu domínio e a disponibilização desses dados como base para a pesquisa do TCC do solicitante Gilvano Cardoso de CPF [redacted] com telefone [redacted]

Fui alertado de que a pesquisa a se realizar é composta por testes com scanners que trarão resultados a respeito de minha aplicação e que os mesmos não serão intrusivos ou prejudiciais ao serviço hospedado. Da mesma forma foi me assegurado que um relatório a respeito dos possíveis pontos vulneráveis me será entregue ao término da pesquisa.

Também fui informado de que posso me recusar a participar do estudo, ou retirar meu consentimento a qualquer momento, desde que antes dos resultados serem publicados em pesquisa.

É assegurada a assistência durante toda pesquisa, bem como me é garantido o livre acesso a todas as informações e esclarecimentos adicionais sobre o estudo e suas consequências, enfim, tudo o que eu queira saber antes, durante e depois da minha participação.

Enfim, tendo sido orientado quanto ao teor de todo o aqui mencionado e compreendido a natureza e o objetivo do já referido estudo, manifesto meu livre consentimento em participar, estando totalmente ciente de que não há nenhum valor econômico, a receber ou a pagar, por minha participação.

Rio Grande Do Norte, de, de 2021.

Nome e assinatura do participante

Gilvano Cardoso Duarte

Nome(s) e assinatura(s) do(s) pesquisador(es) responsável(responsáveis)