Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

NATHAN RODRIGUES DOS SANTOS

ASSEGURANDO A SEGURANÇA DA INFORMAÇÃO CORPORATIVA:
UMA ABORDAGEM ATRAVÉS DE TESTES DE PENETRAÇÃO SIMULADOS

Assis/SP
2024

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

NATHAN RODRIGUES DOS SANTOS

ASSEGURANDO A SEGURANÇA DA INFORMAÇÃO CORPORATIVA:
UMA ABORDAGEM ATRAVÉS DE TESTES DE PENETRAÇÃO SIMULADOS

Projeto de pesquisa apresentado ao curso de Bacharelado em Ciência de Computação do Instituto Municipal de Ensino Superior de Assis – IMESA e a Fundação Educacional do Município de Assis – FEMA, como requisito parcial à obtenção do Certificado de Conclusão.

Orientando(a): Nathan Rodrigues dos Santos
Orientador(a): Fábio Éder Cardoso

Assis/SP
2024

FICHA CATALOGRÁFICA

Santos, Nathan Rodrigues dos

S237a Assegurando a segurança da informação corporativa: uma abordagem através de testes de penetração simulados / Nathan Rodrigues do Santos. -- Assis, 2024.

44p.

Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) -- Fundação Educacional do Município de Assis (FEMA), Instituto Municipal de Ensino Superior de Assis (IMESA), 2024.

Orientador: Prof. Dr. Fábio Éder Cardoso.

1. Segurança cibernética. 2. Segurança de dados. 3. Vulnerabilidades. I Cardoso, Fábio Éder. II Título.

CDD 004

ASSEGURANDO A SEGURANÇA DA INFORMAÇÃO CORPORATIVA: UMA ABORDAGEM ATRAVÉS DE TESTES DE PENETRAÇÃO SIMULADOS

NATHAN RODRIGUES DOS SANTOS

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação, avaliado pela seguinte comissão examinadora

Orientador: _____
Fábio Éder Cardoso

Examinador: _____
Claudinei Moreira da Silva

Assis/SP
2024

DEDICATÓRIA

Dedico esse trabalho especialmente à minha namorada, por ter me acompanhado nessa jornada, nos momentos felizes e nos momentos difíceis. E também aos meus amigos mais próximos pela companhia e apoio nessa etapa desafiadora da minha vida.

AGRADECIMENTOS

Gostaria de agradecer expressar meu sentimento de gratidão por esses quatro anos de aprendizagem e árduos desafios nessa instituição. Foi uma jornada sem igual, que me proporcionou conhecimentos e experiências incríveis.

Agradeço aos meus professores, que com paciência e sabedoria me guiaram nessa jornada acadêmica de maneira formidável. Foi uma honra tê-los como mestres, estamos realmente sobre ombros de gigantes.

Também agradeço aos meus colegas de classe, que trilharam esses anos juntamente à mim, deixando o percurso muito mais animado e leve.

Essa é uma etapa que se finaliza em minha vida, e uma das maiores e mais proveitosas experiências que já tive. Irei me lembrar de cada momento que passei nessa jornada, sejam momentos de vitória ou momentos difíceis que me fizeram persistir e seguir tentando.

RESUMO

Este estudo se concentra na análise da segurança cibernética em empresas, visando identificar vulnerabilidades e ameaças comuns no meio corporativo através de testes de penetração simulados. A proposta desse trabalho foi desenvolver o simulador "*Inside"*, que é capaz de automatizar os testes de penetração em sistemas alvos, identificando vulnerabilidades e explorando-as. E por fim oferece ao usuário final uma análise detalhada dos acontecimentos através de relatórios.

Palavras-chave: Segurança cibernética, Empresas, Teste de penetração, Vulnerabilidades

ABSTRACT

This study focuses on the analysis of cybersecurity in companies, aiming to identify common vulnerabilities and threats in the corporate environment through simulated penetration tests. The purpose of this work was to develop the "Inside" simulator, which is capable of automating penetration tests on target systems, identifying vulnerabilities, and exploiting them. Finally, it provides the end user with a detailed analysis of the events through reports.

Keywords: Cybersecurity, Companies, Penetration testing, Vulnerabilities

Lista de ilustrações

SUMÁRIO

1. INTRODUÇÃO

A evolução constante das ameaças digitais torna essencial oferecer orientações e soluções práticas para fortalecer a segurança cibernética de corporações. *“Muitas vezes, tais empresas são incapazes de compreender a extensão das estruturas de comunicação complexas de hoje e efetuam pouco ou até nenhum controle sobre elas”* (Apud, BERTOGLIO, ZORZO, 2015, p. 2).

Este trabalho de conclusão de curso (TCC) visa aprofundar a análise dos ataques mais comuns enfrentados por empresas, identificando as vulnerabilidades de segurança presentes e oferecendo maneiras práticas de mitigar os riscos.

Seus principais objetivos envolvem identificar os tipos predominantes de ataques no ambiente corporativo e desenvolver uma aplicação especializada para a realização de testes de penetração (*pentest*). Essa aplicação terá como finalidade simular ataques cibernéticos e avaliar proativamente as defesas das empresas, contribuindo para o fortalecimento de sua segurança.

A prática do *pentest* tem como objetivo auditar a segurança da informação de alguma rede ou aplicação, utilizando ferramentas comumente utilizadas por atacantes maliciosos, também conhecidos como *Black Hat*. Pode ser descrito como *“[...] tentativa controlada de penetrar um sistema ou rede a fim de detectar vulnerabilidades. Ela emprega as mesmas técnicas que são utilizadas em um ataque propriamente dito.”* (Apud, BERTOGLIO, ZORZO, 2015, p. 2).

No estado da arte em segurança da informação no meio empresarial, o desenvolvimento de ferramentas de *pentest*, e utilização de diversos *frameworks* como *Open Web Application Security Project* (OWASP) ou *Pentesting Framework* (PTF), tem se destacado como uma abordagem eficaz para avaliar a postura de segurança de uma organização.

Ainda nesse contexto, o Metasploit, um *framework* amplamente reconhecido e utilizado por profissionais de segurança em todo o mundo, tem se destacado como uma ferramenta importante para realização de testes de intrusão e testes de rede.

> *“O Metasploit é um framework criado pelo especialista em segurança de redes norte-americano HD Moore. Tem como principal objetivo criar ferramentas de*

> *explorações de segurança. É utilizado por profissionais do mundo todo para realização de testes de intrusão e testes de rede. Foi desenvolvido na linguagem de programação Ruby e alguns componentes são escritos na linguagem C."* (VIEIRA, 2018, p. 38).

Além do desenvolvimento de ferramentas, adotar políticas de segurança interna se mostra um grande aliado no combate às ameaças digitais. Nesse sentido, a ferramenta desenvolvida por meio desse TCC pode conscientizar as empresas a melhorarem suas politicas de segurança.

> Segundo Vieira *"A política de segurança são normas implementadas em uma organização, com o objetivo de gerenciar todos os recursos tecnológicos de uma organização como também os humanos e culturais, levando em consideração as normas e processos, negócios e leis locais. Ela é o alicerce da segurança na organização, pois sem uma política de segurança, é praticamente impossível aplicar normas que garantam a segurança dos ativos da empresa." (Apud, NAKAMURA, GEUS, 2010)*

Diante desse contexto, o presente trabalho visa contribuir para o estado da arte em segurança da informação no meio empresarial. Ele preenche uma lacuna ao fornecer uma ferramenta eficaz para avaliar abrangente e precisamente o estado da segurança nas empresas estudadas, permitindo identificar vulnerabilidades através de testes de penetração que simulam ataques reais já registrados e catalogados anteriormente.

A lacuna identificada neste trabalho é a ausência de um mecanismo eficiente e prático para avaliar o estado da segurança da informação nas empresas, especialmente com base em incidentes reais já ocorridos no contexto empresarial.

Embora existam normas, diretrizes e boas práticas estabelecidas, muitas organizações ainda encontram dificuldades em identificar vulnerabilidades, mensurar o nível de segurança e implementar melhorias adequadas em seus sistemas e processos.

No presente, as empresas enfrentam um ambiente complexo e em constante evolução, no qual as ameaças cibernéticas se tornam cada vez mais sofisticadas. Nesse contexto, é fundamental contar com um método de avaliação que permita uma análise abrangente e sistemática da segurança da informação.

A segurança da informação empresarial é de extrema importância atualmente, devido à dependência crescente de sistemas e dados digitais. Proteger as informações contra ameaças cibernéticas é essencial para evitar prejuízos financeiros, danos à reputação e violações de leis de proteção de dados. Para garantir a confidencialidade, integridade e disponibilidade dos dados, é necessário implementar medidas como controle de acesso, criptografia, backups e conscientização dos funcionários.

A confidencialidade dos dados empresariais é crucial para manter a competitividade e a confiança dos clientes. Isso envolve a adoção de medidas como autenticação robusta, criptografia e monitoramento de atividades suspeitas para evitar acesso não autorizado a informações estratégicas e dados de clientes. Além disso, a integridade dos dados é essencial para evitar alterações não autorizadas ou corrupção, o que poderia prejudicar a confiança dos clientes e a reputação da empresa.

1.1. OBJETIVO

O objetivo deste TCC é realizar uma análise aprofundada dos ataques mais comuns às empresas e desenvolver um simulador capaz de testar a segurança das empresas contra essas ameaças do mundo real.

Objetivos Específicos:

- Identificar os ataques mais comuns enfrentados por empresas atualmente.
- Desenvolver um software que permite simular diferentes tipos de ataques baseados em uma análise personalizada da infraestrutura da empresa, através de ferramentas de escaneamento de redes e aplicações.
- Realizar uma análise detalhada das lacunas de segurança presentes nessas empresas através do simulador desenvolvido, apontando vulnerabilidades e indicando pontos mais fracos da rede interna da empresa.

Esse trabalho busca contribuir para o aprimoramento do estado da arte em segurança da informação empresarial, preenchendo lacunas ao fornecer uma ferramenta robusta para melhorar a postura de segurança das empresas estudadas.

1.2. JUSTIFICATIVA

A análise da segurança cibernética em empresas é de extrema importância devido a uma série de fatores. Primeiramente, muitas organizações menores frequentemente enfrentam desafios devido à falta de recursos financeiros e técnicos necessários para a implementação de medidas robustas de segurança cibernética. Essa escassez os torna alvos atrativos para ataques cibernéticos, já que suas defesas muitas vezes são menos sofisticadas em comparação com empresas maiores ou mais capitalizadas.

Paralelamente, à medida que as tecnologias avançam e a conectividade aumenta, as ameaças cibernéticas tornam-se mais complexas e frequentes. As empresas, muitas vezes, não estão completamente preparadas para enfrentar essas ameaças, o que as torna particularmente vulneráveis a diferentes formas de ataques.

> *"Atualmente há muitas pessoas que dedicam seu tempo em buscar formas de romper as barreiras de segurança de sistemas diversos como bancos, grandes empresas e até órgãos do governo. Um dos ataques mais recentes e que movimentou a mídia em todo o mundo foi por conta do ransomware WannaCry, que sequestrou os dados de várias empresas ao redor do planeta solicitando um resgate em troca da liberação das informações. [...]"* (GOMES, 2017, p. 9).

Em setembro de 2022 a empresa Rockstar Games sofreu um ataque hacker focado em revelar trechos de seu jogo em desenvolvimento *Grand Theft Auto VI (GTA VI).* Esse ataque revelou mais de 50 minutos de conteúdo confidencial de um produto ainda em desenvolvimento. Esse conteúdo foi publicado pelo hacker em um fórum online para o público. Além dos vídeos referentes ao GTA VI, também foi publicado todo o código fonte do seu antecessor, *Grand Theft Auto V (GTA V),* permitindo que informações sensíveis sobre como a empresa desenvolve seus títulos vazassem para o público geral. (CPOMAGAZINE , 2022)

No mês de dezembro de 2023, outro grande nome em desenvolvimento de jogos digitais foi alvo de ataques hackers. O ataque sofrido foi de proporções inéditas, revelando informações sobre produções ainda em andamento, tais como código fonte de títulos ainda em desenvolvimento, trechos de conteúdo em vídeo e estratégias de negócio da empresa para os próximos anos. O ataque foi realizado pelo grupo hacker *Rhysida,* que pediu aproximadamente 10 milhões de reais em bitcoins em resgate, e estavam dispostos a vender para qualquer organização ou indivíduo que pagasse o preço. (DECRYPT, 2023)

Essas ocorrências evidenciam as potenciais consequências desastrosas que os ataques cibernéticos podem acarretar para empresas, incluindo perdas financeiras, danos à reputação, violações de dados e interrupções operacionais, ressaltando a urgência na compreensão e mitigação desses riscos.

1.3. MOTIVAÇÃO

A motivação por trás deste TCC é fundamentada na necessidade premente de abordar e mitigar as vulnerabilidades de segurança cibernéticas enfrentadas por empresas. Com a crescente dependência dessas organizações em relação à tecnologia e a falta de recursos ou conhecimentos suficientes para implementar defesas robustas, elas se tornam alvos suscetíveis a uma ampla gama de ameaças cibernéticas.

Este estudo é motivado pela necessidade de contribuir para a melhoria da segurança cibernética em ambientes empresariais. Pretende-se oferecer uma melhor segurança nos sistemas da empresa realizando testes simulados para validar existem serviços vulneráveis que podem ser utilizados por criminosos para prejudicar o funcionamento da empresa ou roubar seus dados. Dessa forma a empresa pode agir de forma pró-ativa contra essas ameaças e evitar que os ataques aconteçam de fato.

1.4. PERSPECTIVA

O estudo da segurança cibernética em empresas abre perspectivas significativas para diversas áreas.

Segurança e Proteção de Dados: Uma compreensão aprofundada das vulnerabilidades e ameaças cibernéticas enfrentadas por essas empresas possibilita o desenvolvimento de estratégias mais eficazes para proteger informações sensíveis e dados corporativos. Isso abre caminho para implementar medidas preventivas e corretivas mais direcionadas, garantindo a integridade, confidencialidade e disponibilidade das informações.

Conformidade Regulatória: Ao alinhar-se com os regulamentos de proteção de dados, as empresas estarão melhores preparadas para atender aos requisitos legais, evitando penalidades por não cumprimento das normativas, como o GDPR e a LGPD. Isso também estabelece bases sólidas para construir a confiança com clientes e parceiros, demonstrando comprometimento com a proteção dos dados.

Desenvolvimento Tecnológico: A compreensão aprofundada das ameaças cibernéticas enfrentadas pelas empresas pode impulsionar o desenvolvimento de tecnologias e soluções adaptadas a esses ambientes. Isso pode resultar na criação de ferramentas mais acessíveis e direcionadas para ajudar as organizações a fortalecer suas defesas cibernéticas.

Portanto, a análise e ações propostas neste estudo abrem perspectivas promissoras para aprimorar a segurança cibernética, garantir conformidade regulatória e impulsionar o desenvolvimento de estratégias mais eficazes, contribuindo para a proteção e estabilidade das empresas.

1.5. METODOLOGIA

A metodologia para análise da segurança cibernética corporativa será baseada em etapas estruturadas para garantir uma avaliação completa e eficaz:

Levantamento e Identificação de Ameaças:

- Pesquisa aprofundada para identificar as ameaças mais comuns enfrentadas por empresas ao redor do mundo.
- Revisão de dados de ataques recentes e tendências em segurança cibernética para entender os tipos de ameaças mais prevalentes.

Testes de Penetração Autorizados ou Simulados:

- Realização de testes de penetração autorizados ou em ambientes simulados para avaliar a eficácia das medidas de segurança.
- A simulação dos ataques permitirá uma avaliação realista das defesas e a identificação de áreas que precisam de melhorias.

Elaboração do Simulador de Ataques:

- Desenvolvimento de um simulador que permita reproduzir diferentes tipos de ataques, visando avaliar as defesas e controles de segurança das organizações estudadas.
- Este simulador tem como finalidade mapear a rede da empresa, identificar vulnerabilidades e simular possíveis ataques que agentes mal-intencionados poderiam realizar contra a organização.
- Após o levantamento de vulnerabilidades e ataques, o software vai levantar os pontos fracos da rede ou produto alvo através de relatórios das simulações realizadas, indicando possíveis melhorias.

A metodologia adotada será estruturada com base nas normas NBR 16167, ISO 27001 e ISO/IEC NBR-17799, garantindo uma abordagem abrangente de todos os aspectos da segurança cibernética em uma empresa. Isso inclui desde a identificação e classificação de ameaças e vulnerabilidades até a implementação de controles de segurança, com base nos princípios de confidencialidade, integridade e disponibilidade de informações.

1.6. ESTRUTURA DO TRABALHO

Esse trabalho foi estruturado da seguinte maneira:

- **Capítulo 1 – Introdução:** Esse capítulo foca em contextualizar a área da segurança da informação, definindo o objetivo, justificativa, motivação e perspectivas com relação ao trabalho, além da metodologia empregada ao desenvolvê-lo.
- **Capítulo 2 – Estado da Segurança Cibernética:** Nesse capítulo é abordado um estudo realizado sobre ataques em empresas durante o ano de 2023 e 2024, corroborando com a importância da proteção dos ativos digitais de uma corporação.
- **Capítulo 3 – Tecnologias:** Esse capítulo foca em explicar as tecnologias empregadas na construção da aplicação “Inside”, bem como a importância do seu uso ao decorrer desse trabalho.
- **Capítulo 4 – Construção da Aplicação:** Neste capítulo são apresentadas as metodologias aplicadas para o desenvolvimento da aplicação “Inside”, desde sua arquitetura até o desenvolvimento do código fonte.
- **Capítulo 5 – Resultado da Aplicação:** O objetivo desse capítulo é demonstrar a versão final da aplicação e suas funcionalidades.
- **Capítulo 6 – Conclusão:** Nesse capítulo é abrangido a importância desse trabalho, as vantagens do simulador “Inside” e os desafios enfrentados.
- **Referências**

2. ESTADO DA SEGURANÇA CIBERNÉTICA

Quando falamos de ataques cibernéticos em meios corporativos, o tipo mais comum é o *Ransomware*, que, segundo Souza "[...] pode ser definido como um software desenvolvido para acessar um sistema, bloqueando o dispositivo ou/e criptografando seus dados." (SOUZA, 2017, p. 7). Esses dados sequestrados então são utilizados para extorquir essas organizações, solicitando um resgate em dinheiro.

Utilizando de base o estudo realizado pela empresa de cibersegurança Sophos *"The State of Ransomware 2024"*, podemos ter uma boa estimativa de qual é o estado atual em que as empresas se encontram no presente momento quando se trata de segurança da informação. Esse estudo abrangeu 14 países, em uma sumarização de 5000 empresas lideres em cibersegurança/TI respondendo a pesquisa.

Um dos pontos desse estudo aponta que 99% das organizações afetadas por *ransomware* conseguiram identificar a causa raiz do ataque, sendo exploração vulnerabilidades sendo o ponto de partida mais comum. Podemos entender com isso que tratar vulnerabilidades nos sistemas corporativos é de suma importância para a proteção e integridade dos dados.

Na figura 1 podemos verificar um gráfico indicando os maiores causadores de sistemas comprometidos atingidos por *ransomware*, segundo o estudo da Sophos.

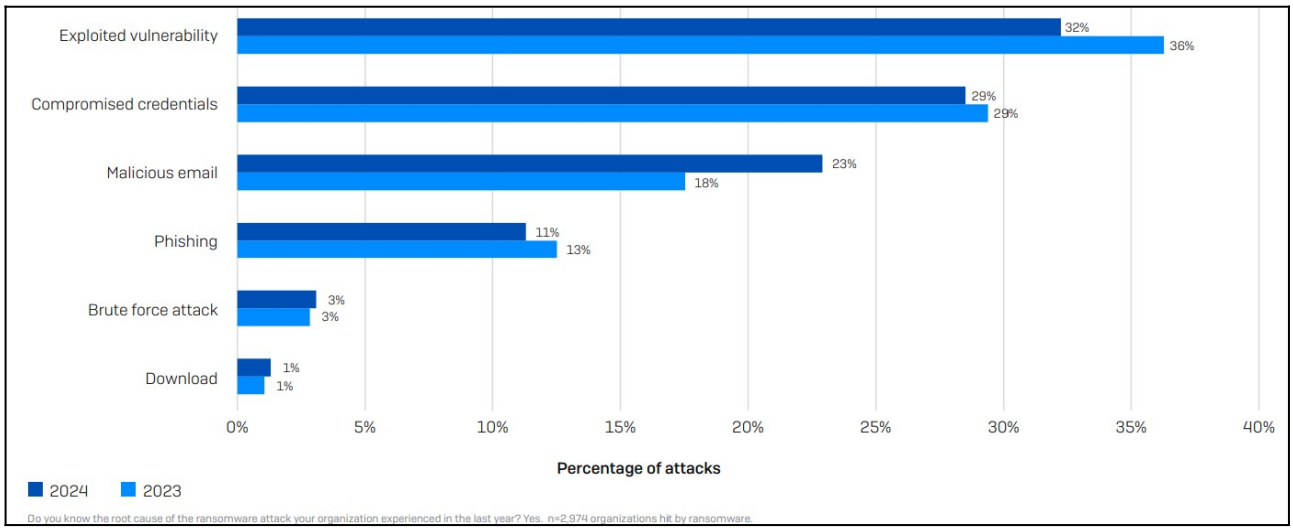

Figura 1: Gráfico de tipos de ataques sofridos por empresas
Fonte: https://assets.sophos.com/X24WTUEQ/at/9brgj5n44hqvgsp5f5bqcps/sophos-state-of-ransomware-2024-wp.pdf

Esse estudo ainda aponta que dentre as 1.701 organizações que tiveram seus dados criptografados e compartilharam o valor inicial do resgate, a média exigida pelos atacantes foi de aproximadamente 24 milhões de reais, enquanto a mediana foi de 11 milhões de reais. Um dos achados mais notáveis é que 63% das exigências de resgate ultrapassam R$ 5 milhões, e 30% são de aproximadamente R$ 28 milhões ou mais.

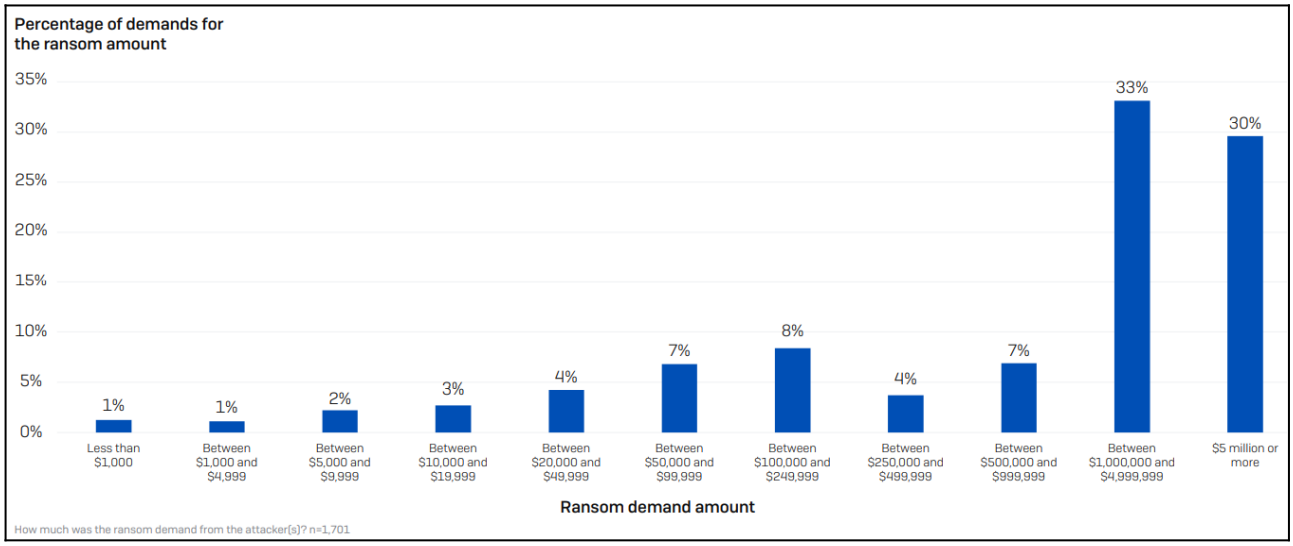

Figura 2: Gráfico de demandas de resgate por dados criptografados
Fonte: https://assets.sophos.com/X24WTUEQ/at/9brgj5n44hqvgsp5f5bqcps/sophos-state-of-ransomware-2024-wp.pdf

Isso reforça a necessidade de práticas rigorosas de segurança cibernética e da adoção de soluções proativas para mitigar riscos, garantindo a integridade dos dados e a continuidade dos negócios. A implementação de simuladores, como o “Inside”, apresenta-se como uma ferramenta eficaz e de baixo custo para auxiliar na identificação e correção de vulnerabilidades antes que possam ser exploradas por agentes mal-intencionados.

3. TECNOLOGIAS

3.1. NMAP

O Nmap (*Network Mapper*) é uma ferramenta de código aberto usada para exploração de rede e auditoria de segurança. Originalmente, foi desenvolvido para fazer varreduras de portas em uma rede para identificar quais estão abertas e quais serviços estão sendo executados em uma máquina, mas suas funcionalidades foram ampliadas ao longo do tempo.

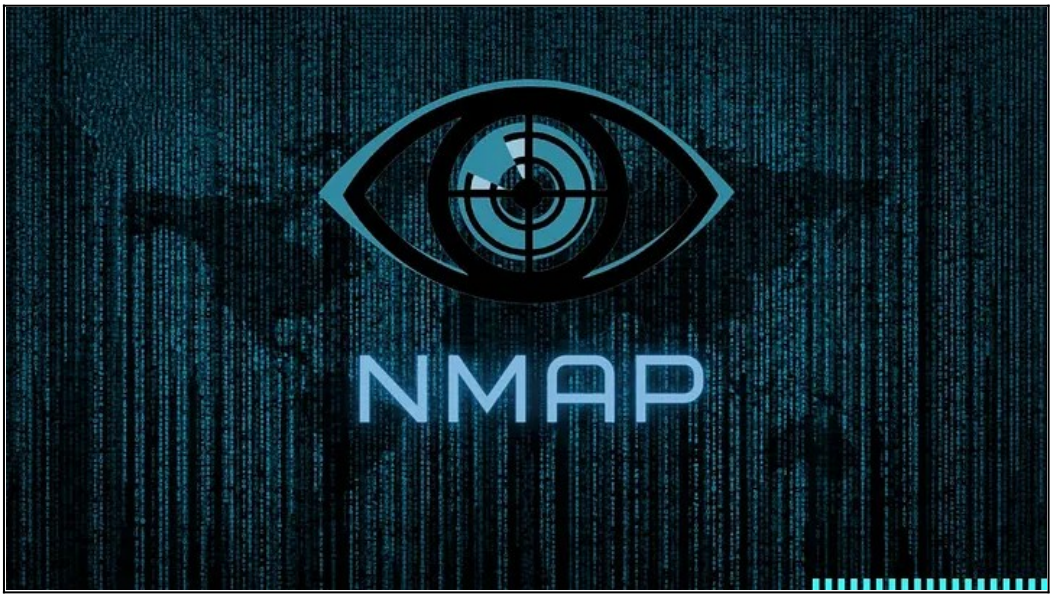

Figura 3: Nmap
Fonte: https://medium.com/@gauravbhandari072/nmap-unleashing-the-power-of-network-scanning-ba511cb774f0

É uma ferramenta poderosa capaz de identificar dispositivos conectados em uma rede (descoberta de *hosts*), determinar quais portas de comunicação (TCP/UDP) estão abertas (varredura de portas), e identificar os serviços executados nessas portas, como HTTP, FTP, ou SSH (detecção de serviços). Além disso, o Nmap pode tentar identificar o sistema operacional do host (detecção de SO) e verificar a existência de vulnerabilidades conhecidas em serviços utilizando scripts (detecção de vulnerabilidades). Por definição de escaneamento de rede, podemos entender que

> *"[..] um escaneamento de conexão TCP tenta fazer conexões TCP com todas as portas de um sistema remoto. Nesse tipo de escaneamento, o host-alvo transmite mensagens quanto ao sucesso da conexão (connection-succeeded) para portas ativas e mensagens*

> *sobre a indisponibilidade (host-unreachable) do host para portas inativas."* (BASTA et al, 2014 p. 46)

A ferramenta é amplamente utilizada por administradores de rede e profissionais de segurança para testar a segurança de redes e sistemas, bem como por hackers para identificar alvos em potenciais ataques. O Nmap pode ser executado a partir de uma interface de linha de comando ou através de uma interface gráfica chamada Zenmap.

Neste trabalho, o Nmap é utilizado na fase de escaneamento, onde ocorre o mapeamento da rede adversária e a identificação dos serviços em uso. A partir desse mapeamento, o sistema é capaz de detectar a presença de serviços vulneráveis, que possam ser explorados. Caso sejam identificadas vulnerabilidades, o processo avança para a fase de ataque.

3.2. KALI LINUX

Kali Linux é uma distribuição do sistema operacional Linux baseada em Debian, amplamente utilizada para testes de penetração, segurança da informação, e análise forense digital. Desenvolvido pela Offensive Security, Kali Linux vem pré-instalado com centenas de ferramentas voltadas para segurança, incluindo ferramentas para testes de invasão, engenharia reversa, análise de vulnerabilidades, e exploração de redes.

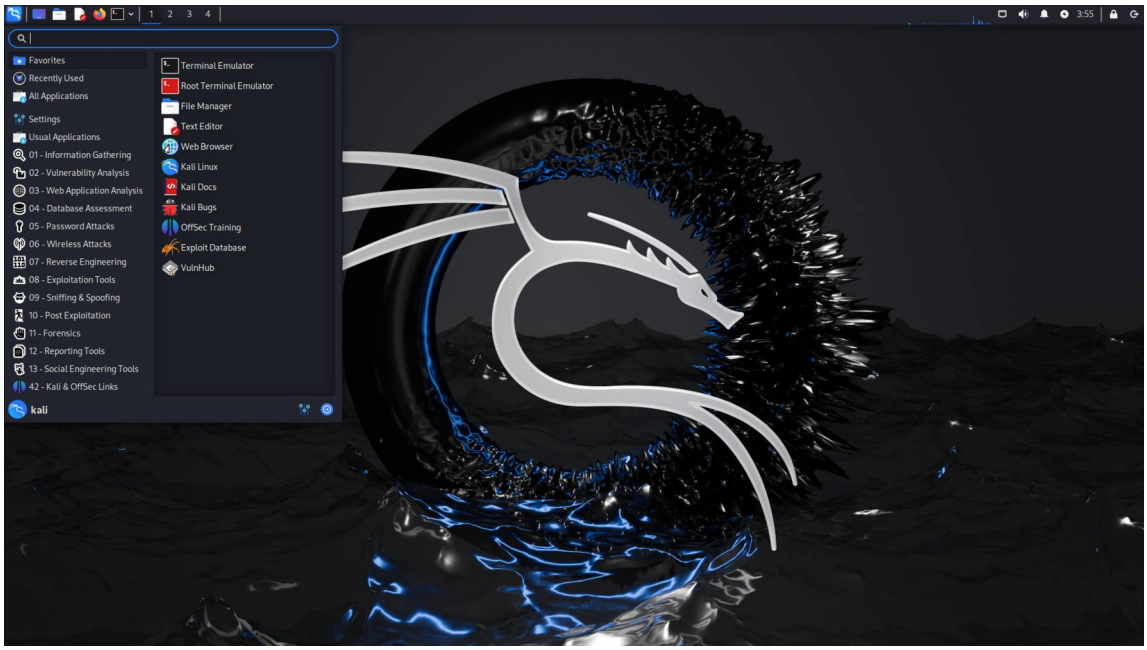

Figura 4: Kali Linux
Fonte: https://www.kali.org/

Seu uso principal é em atividades relacionadas à cibersegurança, onde profissionais realizam testes de penetração para identificar e corrigir vulnerabilidades em sistemas,

redes, e aplicações. Além disso, é utilizado para simular ataques cibernéticos, realizar análises forenses em sistemas comprometidos, e treinar profissionais em técnicas de segurança da informação.

Enquanto a utilização do Kali não está diretamente implementada na aplicação desenvolvida nesse trabalho, esse sistema operacional foi de suma importância para pesquisar e entender as ferramentas que foram de fato integradas no simulador, como o Nmap e Metasploit.

3.3. METASPLOIT

O Metasploit é uma ferramenta de segurança utilizada para realizar testes de penetração e avaliar a segurança de sistemas de computadores. Desenvolvido pela Rapid7, o Metasploit Framework é uma plataforma *open-source* que permite a criação e execução de *exploits* para encontrar e explorar vulnerabilidades em sistemas e redes.

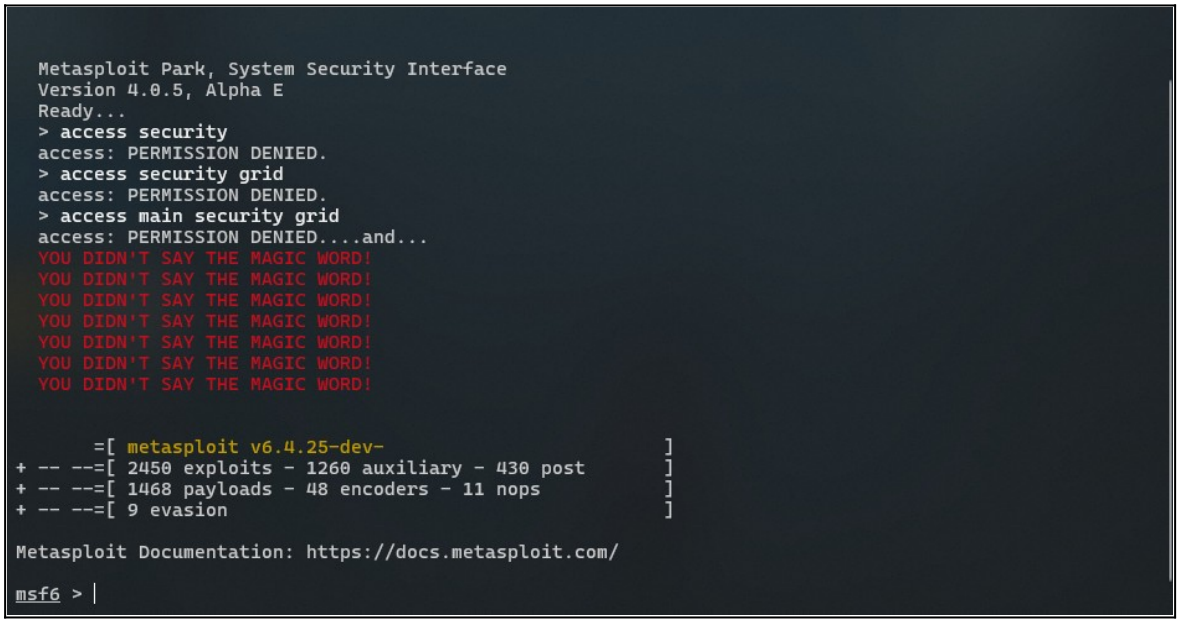

Figura 5: Metasploit CLI
Fonte: De autoria própria, 2024

No presente trabalho essa ferramenta foi crucial para ganhar acesso ao alvo do ataque informado na aplicação. Com o mapa de vulnerabilidades definido e vulnerabilidades detectadas na máquina alvo, a aplicação então chama o Metasploit passando como parâmetro a vulnerabilidade encontrada, o IP do alvo e os *scripts shell* como recurso para a exploração da vulnerabilidade ocorrer.

3.4. VIRTUAL BOX

O VirtualBox é um software de virtualização que permite a criação e execução de máquinas virtuais (VMs) em um único computador físico. Com ele, é possível instalar e executar vários sistemas operacionais diferentes simultaneamente no mesmo hardware, como Windows, Linux, macOS, entre outros, sem a necessidade de particionar o disco rígido ou configurar um sistema de *boot* múltiplo.

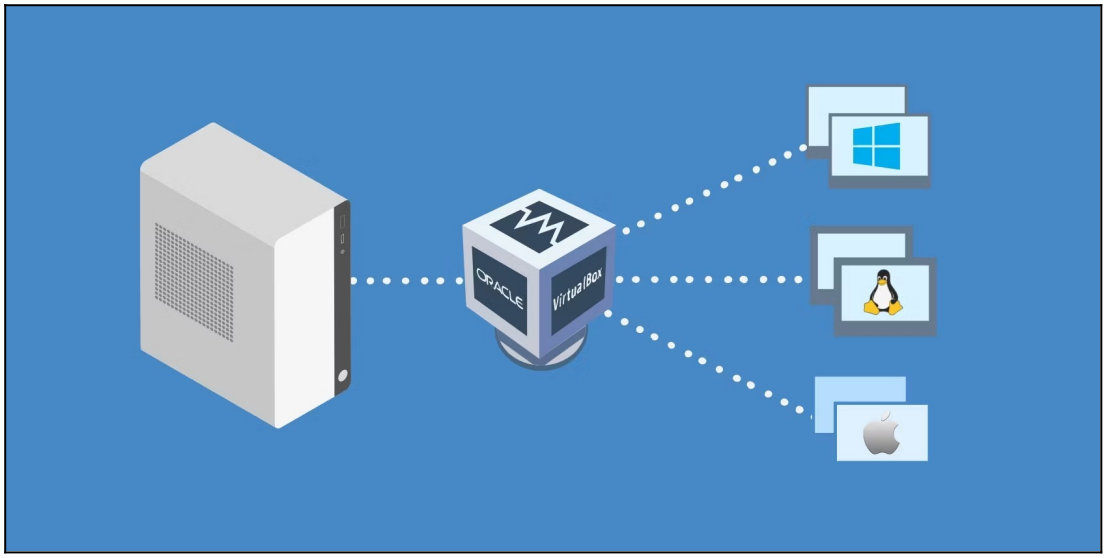

Figura 6: VirtualBox
Fonte: https://www.makeuseof.com/what-is-virtualbox-what-does-it-do/

Essa ferramenta é amplamente utilizado para testes e desenvolvimento, permitindo que desenvolvedores testem aplicativos em diferentes ambientes e versões de sistemas operacionais. Ele facilita o isolamento de ambientes, possibilitando a execução de softwares potencialmente perigosos sem comprometer o sistema principal. Além disso, é útil em ambientes de aprendizado, permitindo que estudantes e profissionais pratiquem com diferentes sistemas operacionais sem necessidade de *hardware* adicional, e pode ser usado para clonagem e *backup* de sistemas, facilitando a recuperação em caso de falhas.

Nesse TCC, o uso do VirtualBox foi necessário para criar ambientes vulneráveis para testes, onde podemos ter máquinas virtuais criadas com o único propósito de ser invadida. Dessa forma temos uma maneira segura e consistente de testar e demonstrar a aplicação atuando como atuaria em uma situação real com variáveis semelhantes.

3.5. METASPLOITABLE

O Metasploitable é uma máquina virtual especialmente projetada para simular um ambiente repleto de vulnerabilidades de segurança, sendo amplamente usada para fins educacionais e de treinamento em segurança cibernética. Desenvolvida e distribuída pela Rapid7, empresa responsável pelo Metasploit Framework, essa VM serve como um alvo seguro para profissionais de segurança, estudantes e pesquisadores testarem técnicas de invasão, exploração de vulnerabilidades e práticas de defesa.

Figura 7: Metasploitable VM
Fonte: https://elhackeretico.com/resolucion-de-metasploitable-2/

O Metasploitable contém diversas vulnerabilidades intencionais, em serviços e aplicativos comuns, como servidores web, bancos de dados, e sistemas operacionais. Isso permite que os usuários pratiquem o uso de ferramentas de exploração como o Metasploit, estudem a dinâmica das falhas de segurança, e desenvolvam habilidades práticas de *pentest*. Além disso, ela é usada em laboratórios e desafios de CTF (*Capture The Flag*), onde o objetivo é encontrar e explorar falhas de segurança para ganhar pontos ou concluir tarefas.

No contexto desse trabalho, o metasploitable será o nosso sistema alvo, onde vamos realizar scans, mapear quais serviços são vulneráveis e cobertos pelo simulador criado, e então invadir esse sistema de forma automática, retornando métricas para o usuário final.

3.6. C++

Segundo Meyers (2011) C++ é uma linguagem de programação de múltiplos paradigmas, que suporta uma combinação de recursos procedurais orientados a objetos, funcionais, genéricos e de metaprogramação. Ela é amplamente adotada em áreas que exigem alto desempenho e controle preciso sobre os recursos do sistema, como sistemas operacionais, jogos, sistemas embarcados, *drivers* de hardware, e aplicações críticas que demandam processamento intensivo.

A escolha de C++ para este trabalho foi motivada por sua capacidade de oferecer alta performance e escalabilidade, o que é essencial para desenvolver aplicações que possam crescer em complexidade e eficiência ao longo do tempo. Com C++, é possível trabalhar diretamente com a alocação e gerenciamento de memória, manipular *buffers*, e otimizar o uso de recursos do sistema, características fundamentais para aplicações que requerem um nível mais baixo de programação.

Além disso, C++ permite a integração de código otimizado com outras linguagens e bibliotecas, facilitando o desenvolvimento de soluções híbridas que podem maximizar o desempenho da aplicação. A linguagem também oferece suporte para programação paralela e concorrente, o que pode ser crucial para futuros requisitos de escalabilidade, especialmente em sistemas multicores e distribuídos.

3.7. CMAKE

CMake é uma ferramenta de automação de compilação que é usada para controlar o processo de compilação de software usando arquivos de configuração independentes de plataforma. Ela gera *scripts* de construção específicos para diferentes sistemas, como Makefiles para Unix ou projetos para Visual Studio no Windows.

É uma ferramenta de construção multiplataforma que permite aos desenvolvedores escrever um único arquivo de configuração (CMakeLists.txt) para gerar *scripts* de construção em várias plataformas e compiladores. Ele suporta a criação de projetos modulares, facilita a organização e reutilização de código, e é adequado para projetos de grande escala com dependências complexas. CMake oferece suporte a várias linguagens de programação, como C, C++, Fortran, CUDA e Python. Além disso, detecta automaticamente dependências entre arquivos-fonte, integra-se com *frameworks* de teste como CTest e permite a criação de regras de instalação para distribuição de software.

Junto a linguagem de programação C++, CMake foi crucial para a compilação de forma coesa e consistente da aplicação Inside, também facilitando a importação de bibliotecas externas necessárias para a execução do projeto, como “nlohmann_json” e “libmariadbcpp”.

3.8. ANGULAR

Angular é um *framework* JavaScript de código aberto desenvolvido pelo Google, lançado em 2010. Ele é usado para criar aplicações web de página única (SPAs) ao permitir que os desenvolvedores usem HTML (*Hypertext Markup Language*) como linguagem de *template* e estendam a sintaxe do HTML para expressar os componentes da aplicação de forma clara e concisa. (ANGULAR, 2024)

Esse *framework* facilita a ligação de dados bidirecional *(two-way data binding*), permitindo que mudanças no modelo de dados sejam refletidas automaticamente na interface do usuário e vice-versa. Além disso, ele inclui recursos como injeção de dependência, diretivas, controle de rotas, e muito mais, tornando a criação de aplicações web mais organizada e eficiente. (ANGULAR, 2024)

O uso desse framework foi crucial para desenvolver um painel de controle (*dashboard)* com uma interface amigável para o usuário final, facilitando a interação com componentes de segurança da infraestrutura da empresa até mesmo para pessoas sem conhecimento prévio na área.

3.9. MARIADB

O MariaDB é um sistema de gerenciamento de banco de dados relacional, desenvolvido como um *fork* do MySQL, após preocupações com a aquisição do MySQL pela Oracle. Ele mantém compatibilidade com o MySQL em grande parte, mas com o tempo incorporou melhorias próprias, como novos mecanismos de armazenamento, otimizações de desempenho e segurança. O MariaDB é amplamente utilizado em sistemas que precisam gerenciar grandes volumes de dados, oferecendo suporte robusto a transações, replicação e alta disponibilidade, além de ser uma solução de código aberto com uma comunidade ativa. (MARIADB FUNDATION, 2024)

O MariaDB é comumente utilizado em aplicativos web, sendo uma escolha popular em pilhas de software como LAMP (Linux, Apache, MySQL/MariaDB, PHP/Perl/Python). Suas capacidades de escalabilidade o tornam adequado para desde pequenos projetos até grandes ambientes corporativos. (MARIADB FUNDATION, 2024)

No presente trabalho esse banco de dados é o principal componente para realizar tarefas que demandam persistência de dados. Acompanhado de seu *driver* de conexão com o C++

foi possível dar autonomia para a aplicação salvar movimentações e gerar relatórios posteriormente.

3.10. API REST

Uma API REST (*Representational State Transfer*) é um estilo arquitetural para a construção de interfaces que permite a comunicação entre diferentes sistemas, principalmente entre servidores e clientes. Baseada nos princípios da web, uma API REST utiliza os métodos HTTP (Hypertext Transfer Protocol) como GET, POST, PUT e DELETE, para realizar operações sobre recursos, que são representados por dados, como um usuário ou um produto. Cada recurso é acessado por meio de URLs (endpoints), e as respostas geralmente estão no formato JSON (JavaScript Object Notation) ou XML (Extensible Markup Language), tornando-as simples e eficientes para o intercâmbio de dados. (CASA DO DESENVOLVEDOR, 2024)

APIs REST são amplamente usadas em aplicativos web e móveis, fornecendo uma forma padronizada e escalável de integrar diferentes serviços e componentes. Elas permitem que os sistemas possam se comunicar de forma independente da linguagem de programação, o que facilita o desenvolvimento de arquiteturas distribuídas, como microsserviços. (CASA DO DESENVOLVEDOR, 2024)

Nesse TCC a escolha dessa arquitetura se deu pela necessidade da elaboração de um *dashboard web* que é a interface do usuário. Com a biblioteca httplib foi possível integrar o *back-end* em C++ com o framework AngularJS, gerando uma API REST com os *end-points* relevantes para o *front-end* consumir.

Nos dias atuais páginas web se tornaram muito populares, e utilizar uma interface web tende a facilitar o entendimento do usuário final, se tratando de um ambiente familiar. Por isso a proposta é construir uma aplicação *desktop*, com uma linguagem no *back-end* que fornece um alto desempenho e integração com o sistema operacional, e disponibilizar uma *Dashboard* para a navegação e utilização do *software*.

4. CONSTRUÇÃO DA APLICAÇÃO

O propósito deste TCC é arquitetar um software capaz de realizar testes de penetração em alvos específicos determinados pelo usuário da aplicação, e com isso obter resultados sobre o sucesso ou falha da obtenção de um acesso remoto não autorizado.

Para chegar à tal resultado com as tecnologias propostas, foi criado um *Dashboard* para a aplicação, que será acessado através do navegador de internet, e através dele serão realizados todas as movimentações.

Com a utilização de uma arquitetura REST, a aplicação em C++ vai se comunicar com a aplicação em Angular através de JSONs, que vão ser responsáveis por carregar os dados enviados e recebidos.

4.1. ARQUITETANDO A APLICAÇÃO

Para o início do desenvolvimento da aplicação, foi primeiro definido qual seria seu escopo de atuação e seus processos internos para alcançar o resultado desejado.

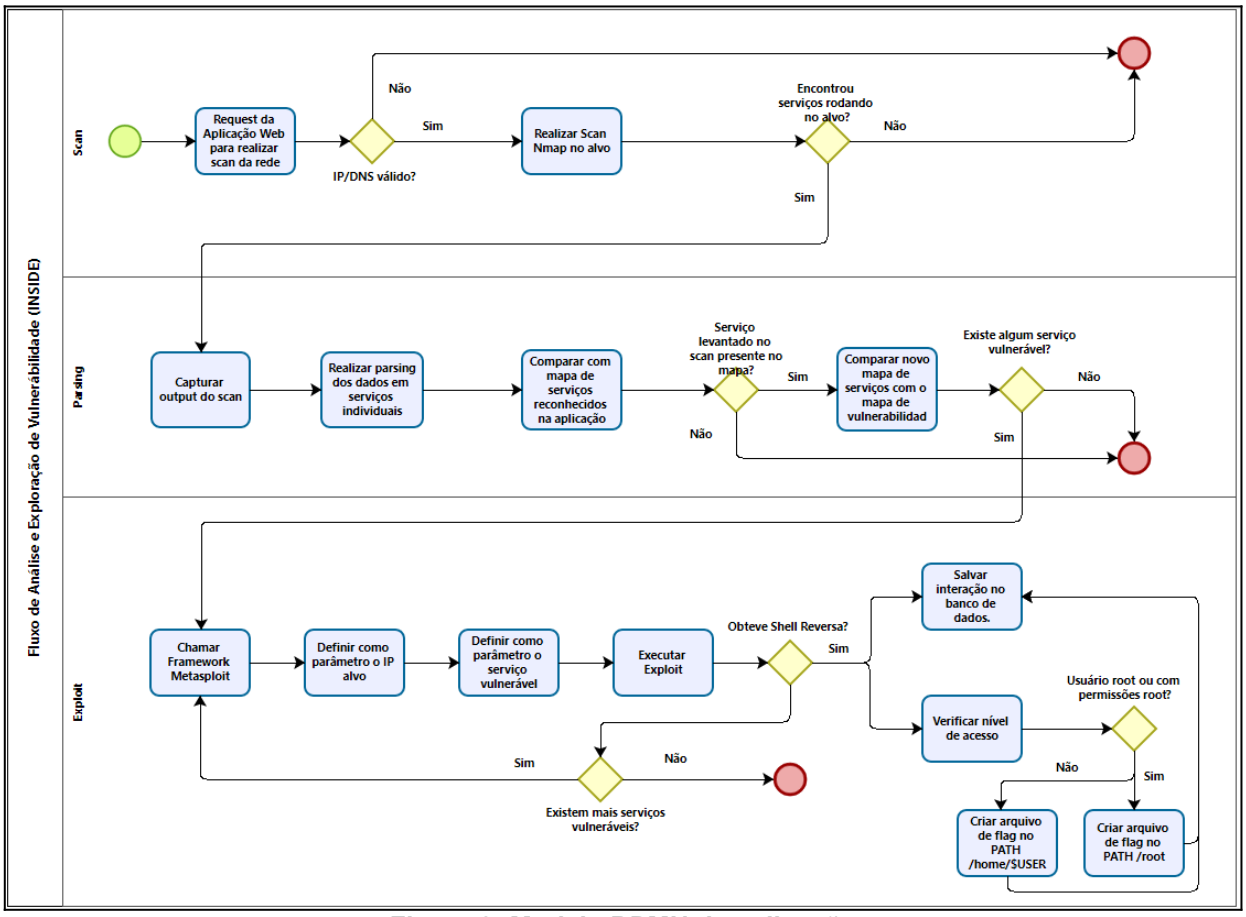

Figura 8: Modelo BPMN da aplicação
Fonte: De autoria própria, 2024

O processo começa com uma solicitação da aplicação web para executar uma varredura na rede. Se o IP ou DNS informado não for válido o processo é interrompido (indicado por um círculo vermelho). Se for válido, a varredura Nmap é executada no alvo.

Se não forem encontrados serviços, o processo é interrompido, mas caso hajam serviços rodando na máquina alvo, o processo continua para uma melhor análise dos dados coletados.

A saída do processo anterior é uma *string* com todos os dados do *scan*, portanto precisamos realizar uma separação desses dados em dados individuais. Após essa separação, cada serviço será comparado com um mapa de serviços reconhecidos pela aplicação como vulneráveis. Caso o serviço esteja no mapeamento de serviços vulneráveis, a aplicação vai avançar para a etapa de exploração da vulnerabilidade no serviço em questão.

Na etapa de exploração de vulnerabilidade, a aplicação vai chamar o Framework Metasploit via linha de comando, passar como parâmetro o IP alvo, a vulnerabilidade à ser explorada e um arquivo de recurso para automatizar ações após a invasão acontecer com sucesso.

Após essa etapa, havendo ou não sucesso, a aplicação vai salvar no banco de dados o momento da execução da simulação e o resultado dessa tentativa de invasão. Se houverem mais serviços vulneráveis identificados e a tentativa de ataque da vulnerabilidade atual falhar, a próxima será utilizada em uma nova tentativa.

Por fim, a aplicação valida o nível de acesso do usuário no qual conseguiu se conectar com uma *shell* reversa e cria um arquivo para sinalizar o sucesso da invasão na máquina. A figura 8 demonstra esse fluxo de processos.

4.2. CONSTRUÇÃO DO SOFTWARE

O propósito deste TCC é arquitetar um software capaz de realizar testes de penetração em alvos específicos determinados pelo usuário da aplicação, e com isso obter resultados sobre o sucesso ou falha da obtenção de um acesso remoto não autorizado.

4.2.1. SERVIDOR RESTFUL

Para a construção da fundação da aplicação, foi necessário criar um núcleo de pontos finais de comunicação para interação entre a aplicação *front-end* com o *back-end*, chamados de *end-points*. Esses *end-points* então serão consumidos pela aplicação Angular no *front-end*, possibilitando a comunicação das duas aplicações.

Na figura 9 é demonstrado como o início da aplicação REST se dá, utilizando a biblioteca httplib para abrir a conexão via HTTP na porta padrão da aplicação (8080). Além disso, esse método é responsável por definir as opções de CORS (Cross-Origin Resourse Sharing), que controla como e os recursos da API podem ser acessados por aplicações de uma

origem diferente. Assim possibilitando que a aplicação *front-end* (de origem externa se comunique com os recursos do *back-end*.

```
void RestfulServer::start() const {
    httplib::Server svr;

    // Handle OPTIONS preflight request
    svr.Options(pattern:R"(/.*)", handler:[](const httplib::Request &req, httplib::Response &res) {
        res.set_header(key:"Access-Control-Allow-Origin", val:"*");
        res.set_header(key:"Access-Control-Allow-Methods", val:"GET, POST, OPTIONS");
        res.set_header(key:"Access-Control-Allow-Headers", val:"Content-Type");
        res.status = 204; // No content
    });

    Routes::setupRoutes(&:svr);

    std::cout << "Server Running on Port: " << port << std::endl;
    svr.listen(host:"0.0.0.0", port);
}
```

Figura 9: Método *start*, da classe *RestfulServer*
Fonte: De autoria própria, 2024

O código demonstrado na figura 10 define rotas HTTP utilizando a biblioteca httplib para um servidor, que é passado como referência ao método *setupRoutes*. As rotas são configuradas para lidar com diferentes *endpoints*, e cada rota está associada a uma função de *callback* específica que lida com a requisição e envia uma resposta.

```
void Routes::setupRoutes(httplib::Server &svr) {
    svr.Post(pattern:"/nmap-scan", handler:[](const httplib::Request &req, httplib::Response &res) {
        handleNmapScan(req, &:res);
    });

    svr.Post(pattern:"/internal-net-analysis", handler:[](const httplib::Request &req, httplib::Response &res) {
        handleInternalNetworkAnalysis(req, &:res);
    });

    svr.Get(pattern:"/home-details", handler:[](const httplib::Request &req, httplib::Response &res) {
        handleHomeDetails(req, &:res);
    });

    svr.Post(pattern:"/network-report", handler:[](const httplib::Request &req, httplib::Response &res) {
        handleNetworkReport(req, &:res);
    });
}
```

Figura 10: Método *setupRoutes*, da classe *Routes*
Fonte: De autoria própria, 2024

Cada rota é definida utilizando uma função lambda, que é uma função anônima e simplificada, como manipulador. No contexto de rotas, essa função lambda é chamada

automaticamente quando a rota correspondente é acessada, e seu objetivo é invocar a função associada àquela rota, passando os objetos de requisição e resposta como referência.

Com esses *end-points* configurados, é possível realizar a integração do *back-end* em C++ com a aplicação *front-end*, escrita em TypeScript, utilizando o Angular como framework.

4.2.2. REALIZANDO SCAN COM NMAP

A realização do scan com a ferramenta Nmap se inicia com o método *prepareNmapScan* da classe Scan, que recebe um IP e os argumentos do scan como parâmetro. Esse método constrói a *string* do comando concatenando o IP e argumentos recebidos, e então envia essa *string* para o método de execução de comando (*executeCommand*) na classe *CommandLineExecute*.

```
std::string Scan::prepareNmapScan(const std::string &ip, const std::string &args) {
    std::string scan = "nmap ";
    std::string cmd = scan.append(str:args).append(s:" ").append(str:ip);
    std::string result = CommandLineExecute::executeCommand(cmd:cmd.c_str());

    auto conn = DBConnector::generateDatabaseConnectionInstance();

    // Result will be stored in the database after exploit running, successfully or not.
    if (conn) {
        std::string activityType = "Scan";
        bool scanSuccess = true; //Always true since nmap will give some output on the target nonetheless.
        ExploitScanRepository::createExploitScanRegister(connection:conn,
                                                         &:const_cast<std::string &>(ip),
                                                         &:activityType, &:scanSuccess);
    }
    return result;
}
```

Figura 11: Método *prepareNmapScan*, da classe *Scan*
Fonte: De autoria própria, 2024

Ao executar o comando, é armazenado no banco de dados o momento e tipo de atividade sendo realizada, e enviado para a classe da camada de banco dados responsável por armazenar os dados de scans.

Esse fluxo, apesar de simples, é o coração das demais funcionalidades da aplicação, pois através dele é gerada a *string* de resultado que será propagada através da aplicação, gerando uma lista de serviços. Essa lista, por sua vez, será comparada com o mapa de

vulnerabilidades conhecidas, e implicará a chamada ou não da execução do *exploit* na máquina alvo.

4.2.3. EXECUÇÃO DE COMANDOS

O método *executeCommand* demonstrado na figura 12 tem como objetivo executar um comando de linha de comando no sistema e capturar sua saída como uma *string*. Ele recebe um comando (cmd) do tipo *const char** e retorna uma *std::string* contendo o resultado da execução do comando.

A função utiliza um *array* fixo de 128 caracteres chamado *buffer* para armazenar os dados temporariamente enquanto o comando é processado. Um ponteiro inteligente (*std::unique_ptr*) é empregado para gerenciar automaticamente a alocação de um arquivo, aberto com a função *popen*, que executa o comando passado e direciona sua saída para leitura. O ponteiro é configurado para fechar o arquivo automaticamente com *pclose* após a execução, garantindo que os recursos sejam liberados corretamente.

```
std::string CommandLineExecute::executeCommand(const char* cmd) {
    std::array<char, 128> buffer{};
    std::string result;

    std::unique_ptr<FILE, decltype(&pclose)> pipe (p:popen(command:cmd, modes:"r"), d:pclose);
    if (!pipe) {
        throw std::runtime_error("Execution failed!");
    }
    while (fgets(s:buffer.data(), n:buffer.size(), stream:pipe.get()) != nullptr) {
        result.append(s:buffer.data());
    }

    return result;
}
```

Figura 12: Metodo *executeCommand*, da classe *CommandLineExecute*
Fonte: De autoria própria, 2024

Dentro de um *loop while*, o método lê a saída do comando linha por linha usando *fgets* e, a cada iteração, adiciona o conteúdo lido ao objeto *result*. Caso o ponteiro do pipe não seja corretamente inicializado (indicando uma falha ao tentar executar o comando), uma exceção *std::runtime_error* é lançada. Após o término da leitura de toda a saída, o método retorna a *string* resultante, que contém a saída completa do comando executado.

Esse método foi criado para majoritariamente lidar com as execuções de programas como o Nmap, mas também foi utilizado para executar o Metasploit em etapas mais avançadas do fluxo da aplicação.

4.2.4. PROCESSAMENTO DOS RESULTADOS DO NMAP

O processamento dos resultados provenientes do scan Nmap é feita de maneira a separar cada serviço encontrado sendo executado na máquina alvo em registros únicos. Esses registros vão conter as informações de porta, protocolo, status de atividade, nome do serviço e a sua versão.

```
std::vector<ScanResult> ScanParser::parseScanResult(std::string &scanResult) {
    std::regex mappingRegex(p: (R"((\d+)(?:\/(tcp|udp|icmp))(\s+open\s+)(\S+)(\s+.*))"));
    std::smatch match;
    std::vector<ScanResult> resultsVector;

    for (auto it = std::sregex_iterator(a: scanResult.begin(),
                                        b: scanResult.end(),
                                        re: mappingRegex);
                                        it != std::sregex_iterator();
                                        ++it) {
        resultsVector.emplace_back(
            ScanResult{ .port: std::stoi(str: TrimUtils::trim(s: (*it)[1].str())),
                .protocol: TrimUtils::trim(s: (*it)[2].str()),
                .status: TrimUtils::trim(s: (*it)[3].str()),
                .service: TrimUtils::trim(s: (*it)[4].str()),
                .version: TrimUtils::trim(s: (*it)[5].str())
            }
        );
    }

    return resultsVector;
}
```

Figura 13: Método *parseScanResult*, da classe *ScanParser*
Fonte: De autoria própria, 2024

Esses dados processados permitem que a aplicação lide mais facilmente e de forma programática com o resultado do scan. Atribuindo esses registros a um vetor do tipo *ScanResult*, podemos passar por todos os resultados e realizar a manipulação desses dados com agilidade.

4.2.5. MAPEAMENTO DE VULNERABILIDADES

Essa etapa vai se dar início após o scan do Nmap finalizar, proveniente do *end-point* "*internal-net-analysis*". Sua principal funcionalidade é receber os resultados já analisados pela classe *ScanParser*, e validar se algum dos resultados está presente nos serviços já conhecidos como vulneráveis.

```
std::vector<std::string> VulnerabilityMapping::findVulnerableServices(
        std::vector<ScanResult>& parsedScanVector,
        std::string &ip) {
    std::vector<std::string> vulnerableServices;

    for (auto &result : parsedScanVector) {
        if (knownVulnerableServices.find(x:result.version) != knownVulnerableServices.end()) {
            vulnerableServices.push_back(result.version);

            auto conn = DBConnector::generateDatabaseConnectionInstance();

            // If it's a match, result will be stored in the database
            if (conn) {
                VulnerabilityRepository::createVulnerableServiceRegister(connection:conn,
                                                                          &:ip,
                                                                          result.port,
                                                                          &:result.version,
                                                                          &:result.service);
            }
        }
    }
    return vulnerableServices;
}
```

Figura 14: Método *findVulnerableServices*, da classe *VulnerabilityMapping*
Fonte: De autoria própria, 2024

4.2.6. EXPLORANDO VULNERABILIDADES ENCONTRADAS

Após encontrar serviços vulneráveis rodando no sistema alvo, a aplicação vai dar início à fase de exploração dessas vulnerabilidades. Para realizar esse processo, será utilizado o framework Metasploit, que possui uma gama de *exploits* pré configurados, bastando apenas informar o alvo, qual serviço deseja explorar e opcionalmente algum recurso extra, como um *script bash* para ser executado após a exploração concluir, dentro do sistema alvo.

A figura 15 demonstra como a aplicação está lidando com a exploração de um serviço após identificá-lo sendo executado no sistema alvo. Primeiro validamos qual é o serviço para tratar de acordo com cada necessidade, e posteriormente o comando é concatenado para ser enviado para a classe de execução da aplicação.

```
for (const auto& service : vulnerableServices) {
    if (service == "vsftpd 2.3.4") {
        std::string exploitScriptPath = "../resources/scripts/ftp.sh";

        command.append(str:exploitScriptPath).append(s:" ").append(str:target);
        std::string resultOutput;
        bool exploitResult;

        resultOutput = CommandLineExecute::executeCommand(cmd:command.c_str());
        exploitResult = checkSuccess(input:resultOutput);

        auto conn = DBConnector::generateDatabaseConnectionInstance();

        // Result will be stored in the database after exploit running, successfully or not.
        if (conn) {
            ExploitScanRepository::createExploitScanRegister(connection:conn, &:target, &:activityType, &:exploitResult);
        }
    }
}
```

Figura 15: Exploração da vulnerabilidade no serviço VSFTPD 2.3.4
Fonte: De autoria própria, 2024

Na finalização da execução, uma instância do banco de dados é aberta e o resultado é validado no método *checkSuccess*, demonstrado na figura 16.

```
bool Exploit::checkSuccess(const std::string& input) {
    std::regex pattern(p:"Exploitation successful!");
    return std::regex_search(s:input, e:pattern);
}
```

Figura 16: Método checkSuccess, da classe Exploit
Fonte: De autoria própria, 2024

4.2.7. ARMAZENANDO RESULTADOS

Em etapas específicas da aplicação, dados do processo sendo executado serão armazenados no banco de dados, para gerar métricas e relatórios. Para esse processo, foi utilizado o banco de dados MariaDB, com seu conector para C++.

```
std::unique_ptr<sql::Connection> DBConnector::getConnection(const std::string& host,
                                                            const std::string& user,
                                                            const std::string& password,
                                                            const std::string& database) {
    try {
        sql::Driver* driver = sql::mariadb::get_driver_instance();
        std::unique_ptr<sql::Connection>
                conn(p:driver->connect(host:"jdbc:mariadb://" + host + "/" + database, user, pwd:password));
        return conn;
    } catch (sql::SQLException& e) {
        std::cerr << "Error connecting to the database: " << e.what() << std::endl;
        return nullptr;
    }
}
```

Figura 17: Conexão com o *driver* do banco de dados MariaDB
Fonte: De autoria própria, 2024

```
std::unique_ptr<sql::Connection> DBConnector::generateDatabaseConnectionInstance() {
    std::string ip = "<DB IP ADDRESS>";
    std::string user = "<DB USER>";
    std::string password = "<DB PASSWORD>";
    std::string database = "<DB NAME>";

    return getConnection(host:ip, user, password, database);
}
```

Figura 18: Dados da conexão com o banco de dados
Fonte: De autoria própria, 2024

As informações para compor a *connectionstring* são geradas no método *generateDatabaseConnectionInstance*, que manda como parâmetro os dados para a conexão no banco de dados para o método *getConnection*. Esse método então chama o *driver* de conexão com o JDBC, concatenando as informações na *connectionstring* para, por fim, se conectar ao banco de dados.

Com essa conexão obtida, a aplicação chama o repositório que deseja utilizar no momento, onde está a lógica de banco de dados, e passa como referência o ponteiro que está armazenando a conexão, permitindo a execução das interações com o banco.

5. RESULTADO DA APLICAÇÃO

5.1. PÁGINA INICIAL

A figura 19 está representando a página inicial da aplicação, onde é informado para o usuário uma gama de análises realizadas previamente na aplicação, como *scans* e vulnerabilidades encontradas recentemente.

Novas *features* poderão ser implementadas nessa página, permitindo que o usuário tenha cada vez mais controle sobre seus sistemas na organização e quais são as métricas gerais de segurança na sua empresa.

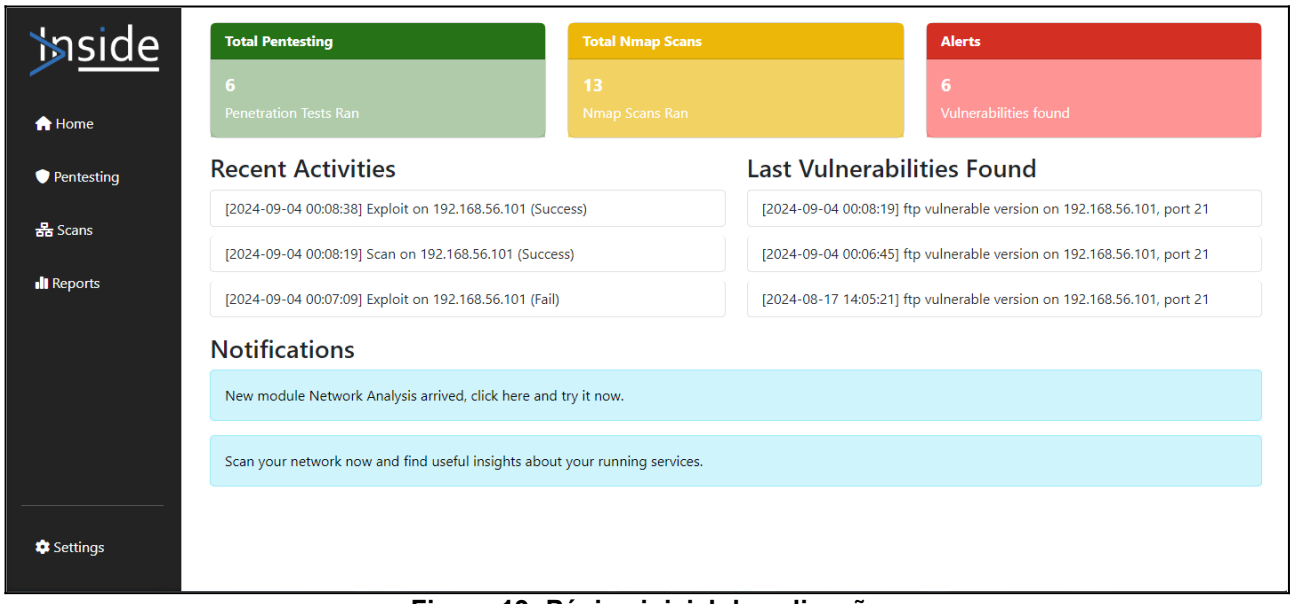

Figura 19: Página inicial da aplicação
Fonte: De autoria própria, 2024

A figura 20 demonstra a tela onde é informado o alvo do simulador. No campo visível, é necessário informar qual o endereço IP da máquina alvo, então selecionar o botão com o símbolo de iniciar para dar início à simulação.

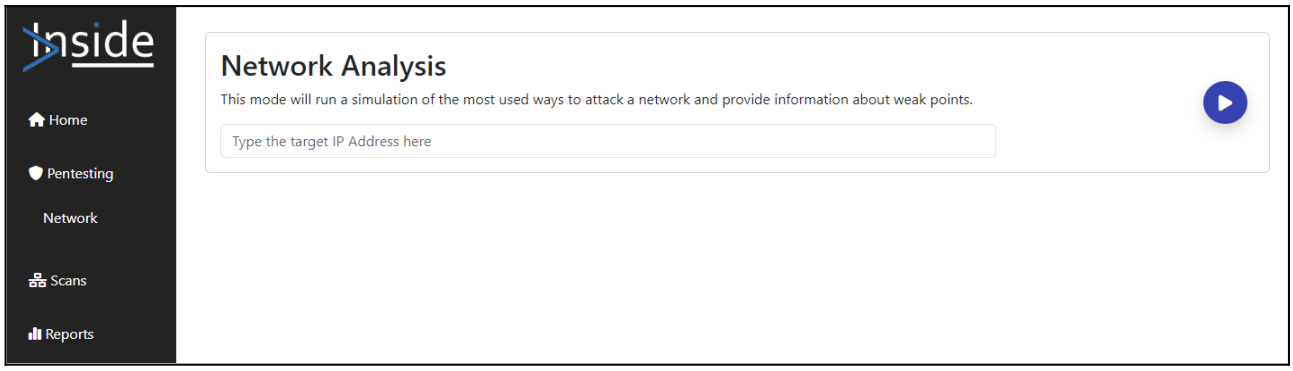

Figura 20: Página da simulação de *pentesting*
Fonte: De autoria própria, 2024

Ao clicar no botão para iniciar a simulação, o *back-end* em C++ vai acionar o Nmap no IP informado, realizará a separação dos serviços, vai comparar os serviços com os serviços vulneráveis conhecidos pela aplicação, e tentar realizar uma invasão utilizando o Metasploit.

Nessa tela o usuário receberá um retorno visual sobre qual foi o resultado da simulação, informando as vulnerabilidades encontradas caso haja alguma, se houve sucesso na exploração de alguma vulnerabilidade e qual foi a vulnerabilidade que permitiu a obtenção da conexão reversa com o sistema alvo.

5.2. PÁGINA DE SCAN

Na figura 21 podemos visualizar a tela de escaneamento Nmap manual, onde o usuário tem a liberdade de adicionar alguns filtros para melhor atender seus propósitos.

Essa tela não tem correlação direta com a simulação demonstrada na figura 20, mas sim serve como um escaneamento de rede *standalone*. Seu objetivo é permitir que o usuário tenha autonomia de utilizar a ferramenta com uma interface gráfica amigável. Em versões futuras, poderão ser implementadas mais ferramentas com uma interface gráfica para facilitar análises personalizadas mais profundas como o Metasploit ou Feroxbuster.

Figura 21: Página de *scan* Nmap
Fonte: De autoria própria, 2024

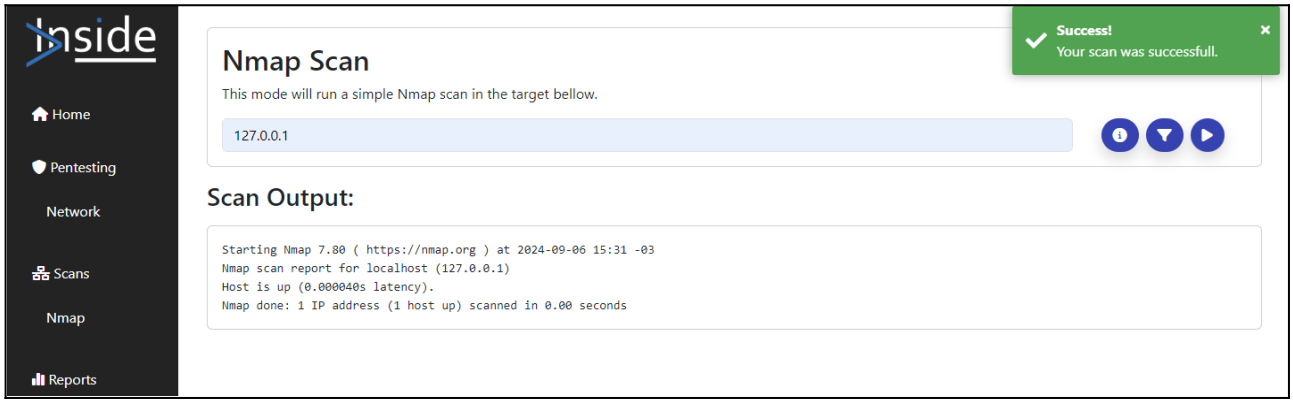

Figura 22: Resultado do *scan*
Fonte: De autoria própria, 2024

5.3. RELATÓRIO DE EXECUÇÕES

A figura 23 demonstra a tela onde podem ser filtrados os *scans* e simulações de testes de penetração realizados, em quais redes foram realizados e seus resultados.

Esse relatório vai permitir que o usuário tenha mais controle sobre como a aplicação está sendo utilizada e a frequência de análises que estão sendo realizadas. Assim o usuário pode ter um panorama de como sua rede está se defendendo das investidas das simulações de ataque.

Figura 23: Página de relatório de rede
Fonte: De autoria própria, 2024

5.4. RELATÓRIO DE VULNERABILIDADES

Esse relatório tem como propósito trazer para o usuário um levantamento geral de todos os serviços vulneráveis que foram encontrados nos sistemas que foram testados pela aplicação.

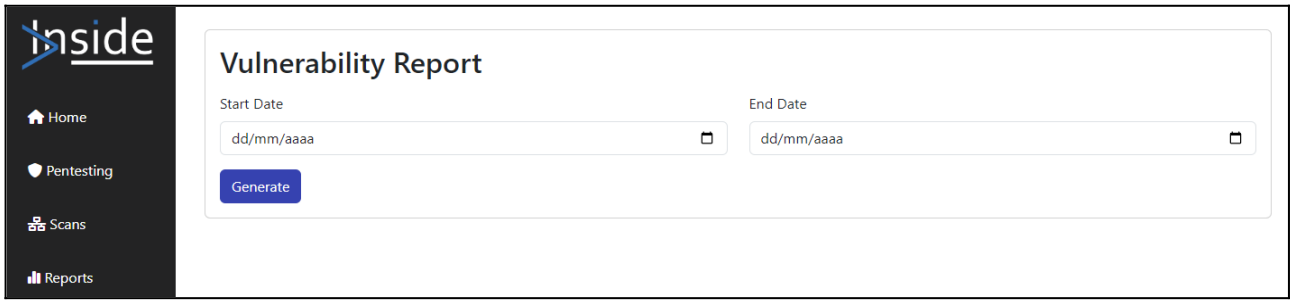

Figura 24: Página de relatório de vulnerabilidades
Fonte: De autoria própria, 2024

Com essa informação armazenada, o usuário pode se preocupar em testar todos os sistemas que deseja e depois voltar validando quais serviços precisam ser atualizados ou desativados, e em quais sistemas específicos foram encontrados

Em versões futuras, esse relatório pode trazer informações adicionais, como dicas específicas para cada vulnerabilidade encontrada, a fim de fazer com que a correção seja mais ágil.

6. CONCLUSÃO

Neste estudo, foi possível realizar uma análise aprofundada sobre o estado atual da segurança cibernética em ambientes corporativos, enfatizando a importância de identificar vulnerabilidades e ameaças comuns que possam comprometer a integridade de sistemas e dados empresariais. O uso da aplicação “Inside” se mostrou eficaz na condução de testes de penetração, permitindo explorar diferentes vetores de ataque e medir o impacto que essas vulnerabilidades poderiam causar.

Os resultados obtidos evidenciaram a relevância de uma postura proativa em relação à cibersegurança. Empresas que adotam uma estratégia reativa, apenas corrigindo falhas após incidentes, ficam mais expostas a danos financeiros, à perda de dados sensíveis e à erosão da confiança dos clientes. Em contrapartida, as empresas que investem em ferramentas preventivas e realizam testes de penetração regulares conseguem mitigar riscos com mais eficácia e responder de forma rápida a novas ameaças.

O desenvolvimento da aplicação abre oportunidade para que as empresas tomem essa postura pró-ativa, identificando ameaças antes de realmente se tornarem um problema real. E por sua facilidade de utilização, não é necessário que o usuário final tenha conhecimentos muito aprofundados em segurança cibernética, facilitando esse contato entre segurança da informação e o meio corporativo.

Um outro fator à se considerar é a realidade e a infraestrutura das pequenas e médias empresas, que muitas vezes possuem recursos limitados para investir em soluções avançadas de segurança. Dessa forma o uso de ferramentas como a aplicação "Inside" mostrou que é possível realizar uma análise eficaz de vulnerabilidades sem a necessidade de altos custos, o que pode beneficiar organizações de diferentes portes.

A utilização da linguagem C++ nesse projeto permitiu que a aplicação fosse desenvolvida para ter um alto desempenho, e, em atualizações futuras, consiga agregar novas funcionalidades sem demasiada dificuldade. No entanto, ela gera algumas dificuldades em nível de complexidade para se implementar algumas funcionalidades que possuem bibliotecas prontas em outras linguagens.

Em contrapartida, no desenvolvimento do front-end da aplicação em angular não houve muitas dificuldades, por se tratar de um framework intuitivo e poderoso. Através dele foi

possível criar uma interface com uma boa experiência de usuário e um design de páginas atraente e intuitivo, dando a aplicação mais ênfase em sua facilidade de uso para um usuário leigo em cibersegurança.

Portanto, este estudo contribui de maneira significativa para o entendimento do panorama da cibersegurança nas empresas. Ele não apenas reforça a necessidade de uma postura preventiva, mas também oferece um modelo prático para a implementação de testes de penetração e o mapeamento de riscos nas empresas. No entanto, cabe ressaltar que este trabalho não esgota o assunto. Futuras pesquisas podem explorar a integração de novas tecnologias, como inteligência artificial e *machine learning*, no aprimoramento dos sistemas de defesa cibernética, bem como o desenvolvimento de metodologias de resposta a incidentes automatizadas para que não apenas seja feita uma análise, mas também uma correção onde se mostrar necessário.

Por fim, espera-se que este trabalho sirva como um ponto de partida para a implementação de práticas de segurança cibernética mais eficazes nas empresas, garantindo a proteção de seus ativos digitais em um cenário cada vez mais ameaçador e complexo. A conscientização sobre a importância de medidas de segurança preventiva e o investimento contínuo em tecnologias e treinamento serão fundamentais para garantir a resiliência das organizações diante dos desafios cibernéticos do futuro.

REFERÊNCIAS

ANGULAR. Disponível em: <https://angular.dev/overview>. Acesso em: 18 ago. 2024.

CASA DO DESENVOLVEDOR. Disponível em: <https://blog.casadodesenvolvedor.com.br/api-rest/>. Acesso em: 18 ago. 2024.

BERTOGLIO, ZORZO. **Um Mapeamento Sistemático sobre Testes de Penetração.** Pontifícia Universidade Católica do Rio Grande do Sul, 2015

BASTA, Alfred; BASTA, Nadine; BROWN, Mary. **Segurança de Computadores e teste de invasão – Tradução da 2ª edição norte-americana**. São Paulo: Cengage Learning Brasil, 2014. *E-book.* ISBN 9788522121366. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788522121366/. Acesso em: 16 ago. 2024.

CPOMAGAZINE. **Rockstar GTA6 Leak Came From Cyber Attack That Breached Internal Slack Channel.** Disponível em: <https://www.cpomagazine.com/cyber-security/rockstar-gta6-leak-came-from-cyber-attack-that-breached-internal-slack-channel/>. Acesso em: 12 ago. 2023.

DECRYPT. **Insomniac Games Breaks Silence on 'Extremely Distressing' Bitcoin Ransom Hack.** Disponível em: <https://decrypt.co/210735/insomniac-games-breaks-silence-extremely-distressing-bitcoin-ransom-hack>. Acesso em: 22 dez. 2023.

GOMES, Marcelo Rodrigues. **A formação profissional de TI no âmbito da segurança da informação: estudo de caso em instituições de ensino superior de Santa Catarina.** Florianópolis – SC, 2017.

MARIADB FUNDATION. Disponível em: <https://mariadb.org/en/>. Acesso em: 18 abr. 2024.

MEYERS, Scott. **C++ eficaz**. Porto Alegre: Grupo A, 2011. ***E-book.*** ISBN 9788577808205. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788577808205/. Acesso em: 16 ago. 2024.

NAKAMURA, E. T.; GEUS, P. L. **Segurança de Redes em Ambientes Cooperativos.** Rio de Janeiro – Novatec, 2010

SOPHOS; **The State of Ransomware 2024.** Disponível em: https://assets.sophos.com/X24WTUEQ/at/9brgj5n44hqvgsp5f5bqcps/sophos-state-of-ransomware-2024-wp.pdf. Acesso em: 22 ago. 2024.

SOUZA, Raniery; **SEQUESTRO DE ARQUIVOS DIGITAIS: ANÁLISE SOBRE AS VULNERABILIDADES DO AMBIENTE E PROPOSTAS DE SOLUÇÕES EM SEGURANÇA.** Disponível em: https://www.unibalsas.edu.br/wp-content/uploads/2017/01/ARTIGO-RANIERY.pdf. Acesso em: 22 ago, 2024.

VIEIRA, Yago Dyogenes Bezerra. **Utilização de Pentest na Prevenção de Ataques Cibernéticos às Organizações.** Universidade Federal Rural de Pernambuco, Unidade Acadêmica de Serra Talhada, 2023.