Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnologia
Curso de Ciência da Computação

Segurança em Aplicações Web: Análise de Vulnerabilidades e Testes de Penetração

Marcos Vinicius Correia Leite

São Luís
2024

Marcos Vinicius Correia Leite

Segurança em Aplicações Web: Análise de Vulnerabilidades e Testes de Penetração

Trabalho de Conclusão de Curso apresentado ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos para obtenção do diploma de bacharel em Ciência da Computação.

Orientador: Prof. Dr. Mário Antonio Meireles Teixeira

São Luís - MA

2024

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Diretoria Integrada de Bibliotecas/UFMA

Leite, Marcos Vinicius Correia.
Segurança em Aplicações Web: Análise de Vulnerabilidades e Testes de Penetração / Marcos Vinicius Correia Leite. - 2024.
48 f.

Orientador(a): Mário Antonio Meireles Teixeira.
Curso de Ciência da Computação, Universidade Federal do Maranhão, São Luís - MA, 2024.

1. Ameaças cibernéticas. 2. Segurança da informação. 3. Segurança em aplicações web. I. Teixeira, Mário Antonio Meireles. II. Título.

Segurança em Aplicações Web:
Análise de Vulnerabilidades e Testes de Penetração

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em: 15/01/2024.

Orientador: Prof. Dr. Mário Antonio Meireles Teixeira
Universidade Federal do Maranhão

BANCA EXAMINADORA

Prof. Ms. Carlos Eduardo Portela Serra de Castro
Universidade Federal do Maranhão

Prof. Dr. Tiago Bonini Borchartt
Universidade Federal do Maranhão

Agradecimentos

Agradeço, em primeiro lugar, à vida e à saúde que Deus me deu, que dão ao homem fonte inesgotável de oportunidades e aprendizado. À minha família e aos amigos, agradeço o alicerce sólido que me sustentou nos momentos mais desafiadoresda vida, faróis que iluminaram meu caminho com risos, compreensão e apoio incondicional.

Ao meu pai Adelmo e minha mãe Rejane, vislumbrei suas forças excepcionais para me incentivar a um caminho de clareza, que me dão forças para acreditar em mim, me deram seu amor em muitas formas possíveis, no tempo, no carinho, aprendizado, subsistência, ceva, ensinamentos e as grandes lutas que ambos tiveram de sofrer, para poder me entregar tudo isso.

À minha companheira de vida Ana Clara, que me deu a razão para me tornar a melhor versão de mim mesmo, esteve sempre a me incentivar e a cuidar de mim, suas palavras sempre foram um grande combustível, agradeço, que por quando eu errava, suas palavras eram além de conforto, duras e verdadeiras, me lembrava que o mundo não se importa se você está cansado e o quanto desgastado está, só vence aquele que cai e volta a ficar de pé!

À orientação e paciência do meu orientador, que guiou meus passos e tive o prazer de conhecer em minhas estradas da UFMA. Cada conselho, cada correção e cada palavra de encorajamento contribuíram para a construção deste trabalho.

Agradeço aos desafios que me foram apresentados, pois se eles que me mostraram minha resiliência e capacidade de superação. Cada obstáculo transformou-se em uma oportunidade de crescimento.

Às vós que cito, me ajudaram a aguentar uma jornada de noites insones, às lágrimas de frustração e aos momentos de incerteza, agradeço por me ensinarem a perseverar. Cada dificuldade foi um degrau a mais na escalada em direção aos meus objetivos.

"Não importa o quão devagar você vá, desde que não pare."

Confúcio

LISTA DE FIGURAS

RESUMO

A segurança em aplicações web é uma preocupação essencial devido ao crescente número de ameaças cibernéticas. A análise de vulnerabilidades e os testes de penetração são práticas fundamentais para identificar e corrigir potenciais brechas de segurança. A avaliação de vulnerabilidades envolve a busca proativa por falhas de segurança, como injeção de SQL, Cross-Site Scripting (XSS) e Cross-Site Request Forgery (CSRF). E neste cenário os testes de penetração vão além, simulando ataques reais para avaliar a eficácia das defesas implementadas. Ferramentas automatizadas e abordagens manuais são empregadas para identificar e explorar falhas. Com estas ferramentas utilizadas numa aplicação WEB com modelo MVC, em ambiente local, foram testadas as abordagens em questão somado com o uso das ferramentas mencionadas. Tudo isso trouxe um entendimento de como se comporta uma aplicação mediante um ataque, e como um atacante pode comprometer o sistema. Mas após esses testes, é necessário que se tome conhecimento de como mitigar esses riscos, a fim de que não deixe mais estas falhas podendo serem exploradas novamente.

Palavras chave: Segurança da informação, segurança de aplicações web, ameaças cibernéticas.

ABSTRACT

The security in web applications is an essential concern due to the increasing number of cyber threats. Vulnerability analysis and penetration testing are fundamental practices for identifying and fixing potential security breaches. Vulnerability assessment involves proactively searching for security flaws, such as SQL injection, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF). In this scenario, penetration testing goes further by simulating real attacks to assess the effectiveness of implemented defenses. Automated tools and manual approaches are employed to identify and exploit vulnerabilities. With these tools used in a web application with an MVC model, in a local environment, the mentioned approaches were tested along with the use of the mentioned tools. All of this provided an understanding of how an application behaves under attack and how an attacker can compromise the system. But after these tests, it is necessary to be aware of how to mitigate these risks so that these vulnerabilities cannot be exploited again.

Keywords: Information security, web application security, cyber threats.

LISTA DE ABREVIATURAS E SIGLAS

CRUD - Create, Read, Update, Delete (Criar, Ler, Atualizar, Deletar)

CSP - Content Security Policy

GDPR - General Data Protection Regulation (Regulamento Geral de Proteção de Dados)

HTTP - Hypertext Transfer Protocol (Protocolo de Transferência de Hipertexto)

HTTPS - Hypertext Transfer Protocol Secure (Protocolo Seguro de Transferência de Hipertexto)

IoT - Internet of Things (Internet das Coisas)

IRP - Incident Response Plan (Plano de Resposta a Incidentes)

MIME - Multipurpose Internet Mail Extensions

MVC - Model-View-Controller (Modelo, Visão, Controlador)

NMAP - Network Mapper (Mapeador de Redes)

NSE - Nmap Script Engine

OWASP ZAP - Open Web Application Security Project Zed Attack Proxy

PTES - Penetration Testing Execution Standard (Padrão de Execução para Testes de Penetração)

RESTful - Representational State Transfer (Transferência de Estado Representacional)

CSRF - Cross-Site Request Forgery (Falsificação de Solicitações entre Sites)

XSS - Cross-Site Scripting (Script entre Sites)

Web - World Wide Web (Rede Mundial de Computadores).

SUMÁRIO

1. INTRODUÇÃO

O cenário digital teve um crescimento exponencial de aplicações web, possibilitando o grande emprego de novas ferramentas e serviços no âmbito digital, as inovações das aplicações web trouxe consigo grande positividade para o mercado de trabalho, criando novas empresas com serviço web, como por exemplo: aplicações para cadastro de alunos, cadastro de funcionários, emissão de notas fiscais, entre outras; Até mesmo o crescimento de empresas que já vinham atuando com serviços web e começaram a expandir suas entregas com novas utilidades, assim como foi com a empresa google, que antes era focada como uma ferramenta de busca digital muito poderosa, hoje atua no mercado como uma empresa que oferece um dos melhores serviços de email, streaming privado, aplicativos e muito mais.

De acordo com Mitnick (2005), a ascensão vertiginosa dessas novas empresas e tecnologias trouxe consigo um aumento significativo nas ameaças cibernéticas, destacando a necessidade premente de abordagens robustas em segurança da informação.

Nos últimos anos, marcada pela crescente interconexão digital e dependência de avançadas tecnologias, a segurança cibernética emerge como uma disciplina de extrema importância, incumbida da preservação da integridade, confidencialidade e disponibilidade das informações digitais. Este domínio abrangente concentra-se na blindagem de sistemas, redes e dados contra uma ampla gama de ameaças cibernéticas, que abarcam desde ataques de malware até sofisticadas tentativas de intrusão. Em um mundo denso onde a massividade da quantidade de pessoas se torna inviável ter organização com tantos indivíduos, a digitalização se torna necessária para organizar este contingente populacional, e com esta tendo que arcar com o recebimento de tantos dados, a segurança cibernética desempenha um papel vital na proteção de informações sensíveis, na preservação da privacidade e no asseguramento do funcionamento seguro de infraestruturas críticas.

Neste contexto, a abordagem de "Testes de Penetração" emerge como uma metodologia bem importante no cenário atual para identificar e corrigir falhas de segurança. Um profissional da área de pentest, este utiliza de um olhar pelo ponto de vista de um atacante cibernético, utilizando-se de metodologias diferenciadas

para a obtenção de ferramentas e modos de identificar vulnerabilidades, atuando de maneira estratégica e metodológica para identificar e corrigir vulnerabilidades em sistemas, redes ou aplicações, simulando as ações de um potencial atacante. A abordagem é realizada de forma ética e autorizada, com o objetivo de fortalecer a postura de segurança de uma organização e/ou sistemas, a fim de expor a tal falha, para que esta possa ser sanada.

1.1 Objetivo Geral

Utilizar uma aplicação web para realizar testes de penetração e propor melhorias de segurança digital.

1.2 Objetivos Específicos

O Penetration Test Execution Standard (PTES), é uma estrutura formalizada que estabelece as diretrizes e os padrões para a execução de testes de penetração. Desenvolvido para fornecer uma abordagem sistemática e abrangente, o PTES é projetado para orientar profissionais de segurança cibernética na condução de testes de penetração de forma consistente e eficaz. Vejamos os tópicos em que será focado neste trabalho.

- Identificar vulnerabilidades comuns e realizar uma revisão detalhada da infra-estrutura da aplicação em questão.
- Desenvolver metodologia de teste de penetração usada abrangentemente para a realização de testes de penetração em aplicações web.
- Realizar testes práticos em ambiente controlado e implementar alguns dos conceitos da metodologia desenvolvida.
- Propor recomendações para mitigação de vulnerabilidades em aplicações web com base nos resultados obtidos

2. REFERENCIAL TEÓRICO

Neste capítulo serão revisados os conceitos principais relacionados a este trabalho.

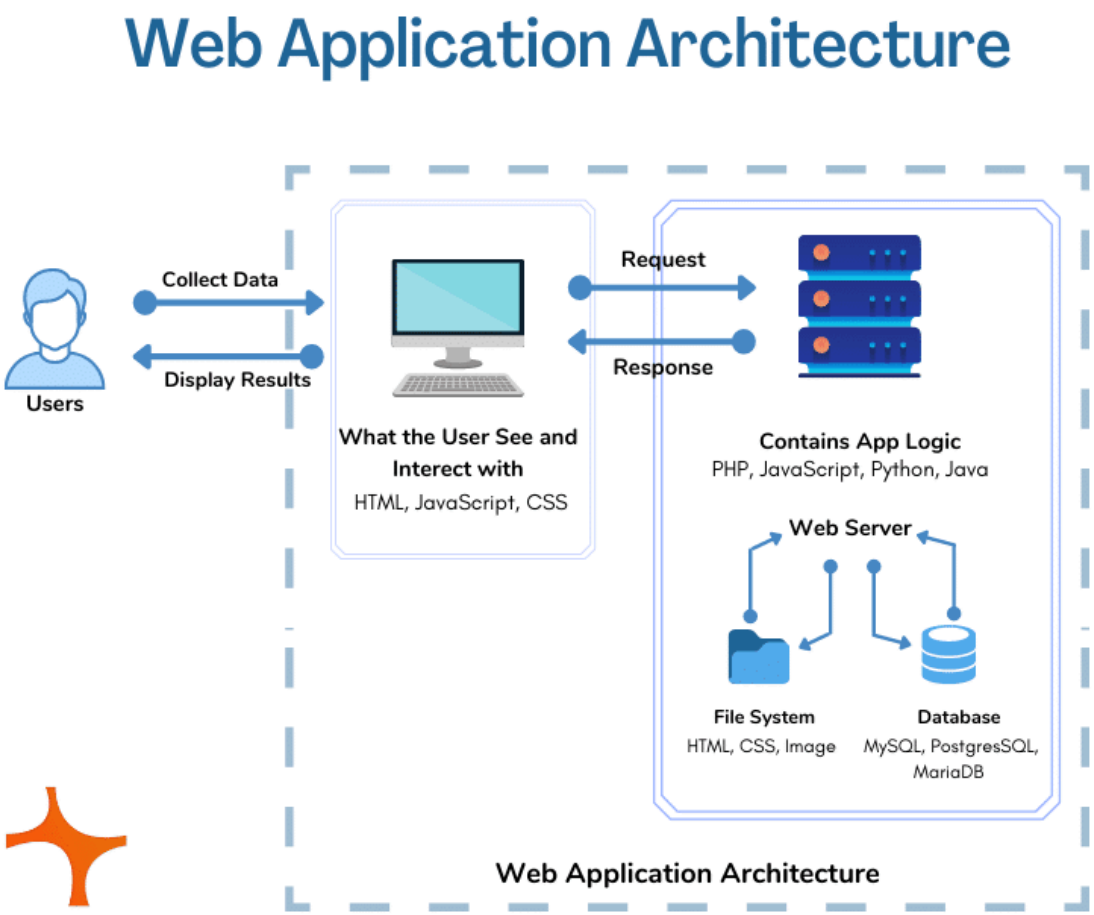
2.1 Aplicações Web

Uma aplicação web é um tipo de software projetado para operar mediante o acesso e a interação por meio de um navegador web. Ao contrário das aplicações tradicionais que são instaladas localmente em dispositivos individuais, as aplicações web são hospedadas em servidores remotos e oferecem funcionalidades aos usuários através de interfaces de usuário acessíveis por navegadores da web convencionais.

As características fundamentais das aplicações web englobam elementos intrínsecos que contribuem para a sua natureza e funcionalidade específicas. Entre esses atributos, destaca-se, em primeiro plano, a imperativa necessidade de conectividade à Internet. Este requisito é inerente à própria essência das aplicações web, as quais dependem da rede global para viabilizar a transmissão de dados e interações em tempo real.

Outro aspecto crucial reside na configuração de uma interface de usuário interativa que, por sua vez, busca simular experiências análogas às proporcionadas por aplicações desktop. Esta abordagem visa criar uma experiência de usuário fluida e intuitiva, promovendo a eficácia na interação com a aplicação web, e, por conseguinte, melhorando a usabilidade geral. Adicionalmente, um elemento distintivo é a capacidade de realizar atualizações de maneira simplificada nos servidores remotos, podemos ver isso na figura 1 o funcionamento genérico de uma aplicação web. Esse mecanismo confere agilidade e eficiência ao processo de implementação de novas funcionalidades, correções de bugs e aprimoramentos, sem a necessidade de intervenções diretas nos dispositivos dos usuários.

Figura 1 - Modelo de funcionamento de uma aplicação web



Fonte: Cynoteck. 2021

A acessibilidade multiplataforma é outra característica inerente às aplicações web, permitindo sua utilização em diversos sistemas operacionais e espaços geográficos. Essa flexibilidade proporciona aos usuários a liberdade de escolha em relação aos dispositivos e ambientes nos quais desejam acessar a aplicação, fomentando a abrangência e a adaptabilidade. As características fundamentais das aplicações web transcendem a mera prestação de serviços online, abraçando a conectividade, interatividade, facilidade de atualização e a compatibilidade com diferentes plataformas. Estes atributos coletivamente contribuem para a definição e eficácia dessas aplicações em um panorama tecnológico cada vez mais dinâmico e interconectado, quanto mais bem estruturado forem estes aspectos, mais reconhecido o software se torna para uma gama extensa de usuários.

Aplicações web, de maneira recorrente, incorporam a capacidade de viabilizar colaboração em tempo real, facilitando a interação simultânea de múltiplos usuários

com as mesmas funcionalidades do sistema para a segurança emerge como uma consideração essencial, respaldada por medidas meticulosamente implementadas com o propósito de salvaguardar os dados dos usuários e garantir transações seguras no contexto de sua operação em um ambiente online.

A colaboração em tempo real, enquanto atributo distintivo, amplia significativamente a eficácia e a versatilidade das aplicações web, permitindo que diferentes usuários interajam de forma sincronizada e participem ativamente na manipulação ou análise de informações compartilhadas. Esta funcionalidade não apenas otimiza a eficiência operacional, mas também proporciona um ambiente de trabalho colaborativo, com implicações particularmente relevantes em contextos empresariais e educacionais.

No que concerne à segurança, esta é abordada de maneira abrangente e proativa, dada a natureza sensível dos dados transitados e armazenados pelas aplicações web. Estratégias de criptografia, autenticação robusta e autorização granular são implementadas para proteger a confidencialidade, integridade e disponibilidade dos dados do usuário. Além disso, medidas de prevenção contra ameaças cibernéticas, como firewalls e sistemas de detecção de intrusões, são incorporadas para mitigar potenciais riscos à segurança.

A necessidade de assegurar transações seguras é especialmente crucial, considerando que as aplicações web operam em um ambiente online suscetível a uma diversidade de ameaças digitais. Mecanismos de criptografia de dados durante a transmissão, autenticação de usuários e práticas de gerenciamento de sessões são implementados para garantir a integridade das interações e a legitimidade dos usuários envolvidos. A integração de funcionalidades de colaboração em tempo real e a implementação rigorosa de medidas de segurança refletem o compromisso das aplicações web em fornecer uma experiência robusta, eficiente e, acima de tudo, segura para os usuários que operam em ambientes online dinâmicos e interativos. Exemplos de aplicações web englobam serviços de e-mail, redes sociais, plataformas de comércio eletrônico, sistemas de gerenciamento de conteúdo (CMS) e diversas ferramentas especializadas que oferecem funcionalidades específicas acessíveis por meio de navegadores web.

2.1.2 Persistência da Conexão

Para melhorar o desempenho, algumas conexões podem ser mantidas abertas (persistência de conexão), podendo ser opcional, para evitar a necessidade de estabelecer uma nova conexão para cada solicitação subsequente, um bom exemplo são páginas que realizam provas temporizadas, a desvantagem desse método é o uso de banda e processamento.

2.2 Modelos arquiteturais

Os paradigmas arquiteturais para aplicações web definem abordagens destinadas a organizar tanto o código quanto os componentes inerentes a uma dada aplicação. O escopo desses modelos visa proporcionar uma estrutura que favoreça de maneira significativa o desenvolvimento, a manutenção e a escalabilidade eficiente dos sistemas, visto que essas características tomam grande quantidade de tempo e recursos no decorrer da vida de um sistema. E dentre os modelos arquiteturais frequentemente empregados em aplicações web, destacam-se alguns como flux, RESTful, microservices, graphQL, serverless, e um dos mais usados e que iremos abordar neste estudo, o MVC (Model View Controller). A escolha criteriosa do modelo arquitetural a ser adotado repousa nos requisitos específicos do projeto, da equipe de desenvolvimento e nas singularidades intrínsecas à aplicação em questão. Com frequência, adota-se uma combinação judiciosa de modelos, de modo a satisfazer distintas exigências em diferentes segmentos do sistema.

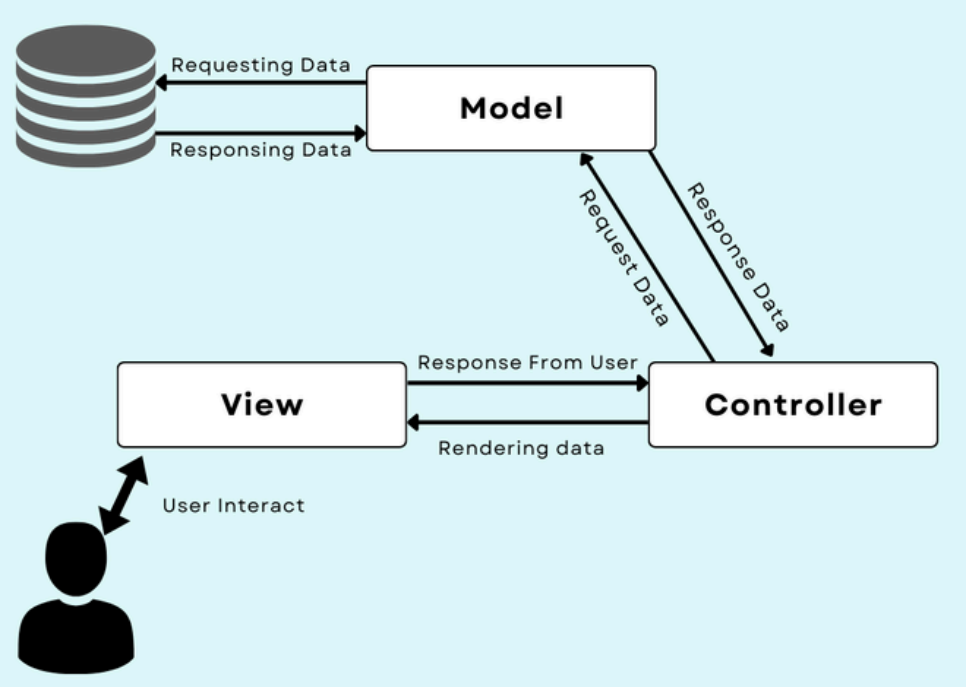
2.2.1 Arquitetura MVC

A arquitetura muito usada nas aplicações web chamada de MVC é um padrão arquitetural de significativa relevância para o desenvolvimento de aplicações web e de software em geral. Este modelo desmembra a aplicação em três componentes distintos: Modelo, Visão e Controlador, cada qual com atribuições específicas, colaborando de maneira coordenada para estabelecer uma estrutura organizada e de fácil manutenção.

O Modelo, primariamente, encarrega-se da representação da lógica de negócios e dos dados da aplicação, gerenciando tanto o acesso quanto o processamento dos dados pertinentes. A Visão assume a responsabilidade pela

apresentação dos dados ao usuário, exibindo as informações que o Modelo tratou e devolveu de acordo com as diretrizes do Controlador. Por fim, o Controlador desempenha o papel de intermediário, recebendo as interações do usuário provenientes da Visão, processando essas entradas e efetuando atualizações no Modelo. Adicionalmente, notifica a Visão para atualizar a interface do usuário quando ocorrem alterações no Modelo, na figura 2 temos a ilustração para entender como funciona esta arquitetura.

Figura 2 - Arquitetura MVC



Fonte: GeeksforGeeks. (2022).

A interação do usuário concentra-se predominantemente na Visão, que direciona essas interações ao Controlador. Este, por sua vez, efetua as atualizações no Modelo conforme necessário, preservando a coerência entre a lógica de negócios e a interface do usuário. A clara distinção entre esses três componentes facilita a manutenção e o desenvolvimento independente de cada um, promovendo a reutilização eficiente do código.

O padrão MVC é extensivamente empregado em frameworks web, a exemplo do Ruby on Rails, Django (Python), Spring (Java), e em tecnologias de interface do usuário, tais como Cocoa (para aplicativos macOS) e Angular

(JavaScript/TypeScript). Em síntese, o MVC é um modelo arquitetural eficaz que fomenta a modularidade, escalabilidade e manutenção em projetos de desenvolvimento de software.

2.3 Segurança Cibernética

Veja o que diz Bruce Schneier (2001), renomado especialista em segurança cibernética e autor de "Segredos e Mentiras: A Segurança na Era da Informação":

> *"Os três pilares fundamentais da segurança da informação - confidencialidade, integridade e disponibilidade - são a espinha dorsal que sustenta a confiança na era digital. Sem esses princípios, a segurança cibernética seria uma estrutura frágil e vulnerável, sujeita a abalos constantes."*

Quando Schneier fala sobre esses três pilares ele quer dizer que estes constituem os alicerces essenciais para a preservação da integridade e confiança nos ambientes digitais contemporâneos.

A confidencialidade, primariamente, versa sobre a proteção de informações sensíveis, assegurando que apenas entidades autorizadas tenham acesso a dados restritos. Este princípio visa prevenir vazamentos ou divulgações inadvertidas que possam comprometer a privacidade e a segurança dos dados. Também se estende à gestão adequada de privilégios de acesso. Garantir que apenas usuários autorizados possam visualizar e manipular dados confidenciais, sendo essencial para a preservação da privacidade e a conformidade com as regulamentações de proteção de dados.

A integridade, por sua vez, concentra-se na precisão e consistência dos dados. Isso envolve não apenas evitar modificações não autorizadas, mas também implementar mecanismos que assegurem a consistência e precisão dos dados. A integridade é crucial em setores onde a exatidão das informações é crítica, como em registros médicos, transações financeiras, processos judiciais, que usam por exemplo, exatidão de dados informados. Este pilar é crucial para garantir a confiabilidade das informações. Busca-se evitar qualquer forma de alteração não autorizada, manipulação indevida ou corrupção das informações armazenadas.

A disponibilidade emerge como um terceiro pilar essencial, enfocando a prontidão e acessibilidade contínua dos sistemas e dados quando necessários. Isso não apenas envolve prevenir ataques que possam resultar em interrupções, mas também planejar para contingências, como desastres naturais ou falhas de hardware, que possam afetar a continuidade dos serviços. Este princípio visa mitigar interrupções indesejadas ou ataques que possam comprometer a acessibilidade e funcionalidade dos recursos digitais.

A tríade formada por esses pilares é intrinsecamente interdependente, promovendo uma abordagem equilibrada para as estratégias de segurança. Em conjunto, eles visam estabelecer um ambiente digital robusto e confiável. Esses princípios não apenas delineiam a postura defensiva contra ameaças cibernéticas, mas também fundamentam a confiança necessária para operações digitais seguras em uma sociedade interconectada. A segurança cibernética é um domínio preciso no ambiente digital contemporâneo, compreendendo um conjunto de práticas, tecnologias e processos concebidos para blindar sistemas, redes e dados contra crises cibernéticas. Este campo dinâmico e em constante evolução desempenha um papel crucial devido à crescente complexidade e sofisticação tanto das tecnologias quanto das ameaças digitais, já que essas duas, estão sempre em constante evolução e surgindo com novos paradigmas.

Nesta ascensão da tecnologia, a inteligência artificial (IA) e a aprendizagem de máquina (ML) estão sendo cada vez mais incorporadas para aprimorar a capacidade de detecção de ameaças. Estas permitem análises mais rápidas e precisas de padrões de comportamento, identificando atividades suspeitas em tempo real.

Atualmente a segurança cibernética não se restringe apenas ao ambiente empresarial. Com o advento da Internet das Coisas (IoT - internet of things), a expansão de dispositivos interconectados cria novos desafios de segurança, exigindo medidas para proteger não apenas os sistemas tradicionais, mas também objetos do cotidiano conectados à internet, como celulares, relógios, geladeiras e até veículos.

A colaboração global e a partilha de informações são aspectos fundamentais na defesa. Comunidades de segurança, organizações e governos compartilham inteligência sobre ameaças, padrões de ataque e melhores práticas para fortalecer a resiliência cibernética em escala global.

Por isso há o surgimento de profissionais de segurança cibernética tendo muitas vagas no mercado, tais profissionais desempenham um papel crucial na concepção e implementação de estratégias de segurança, contra ataques, para isso realizam monitoramento contínuo, análise forense, resposta a incidentes e a implementação de políticas de segurança robustas e assim então deixar o ambiente preparado para quando o cenário for real, os danos serem mínimos ou até nulos.

2.3.1 Usuários

A promoção da conscientização do usuário emerge como um elemento crucial no âmbito da segurança cibernética. A implementação de programas de treinamento destina-se a cultivar práticas seguras entre os usuários, capacitando-os a reconhecer e lidar com ameaças potenciais. Essa abordagem visa não apenas fortalecer a postura defensiva da organização, mas também criar uma cultura de segurança proativa.

Os treinamentos proporcionam aos usuários o entendimento necessário sobre as práticas de segurança recomendadas, incluindo a identificação de comportamentos suspeitos, a manutenção de senhas robustas e a compreensão dos riscos associados a práticas negligentes. Ao elevar o nível de literacia em segurança cibernética, a organização estabelece uma linha de defesa adicional, mitigando a probabilidade de incidentes originados por falhas humanas.

Além disso, a conscientização do usuário não se limita apenas ao treinamento inicial; ela deve ser um processo contínuo, adaptando-se às evoluções das ameaças cibernéticas. Estratégias de sensibilização regular, atualização de conteúdo e simulações de ataques contribuem para manter os usuários informados e vigilantes diante de ameaças em constante mutação.

2.3.2 Planos de resposta a incidentes

A presença de planos de resposta a incidentes (IRP- Incident Response Plan) é de suma importância para a gestão eficaz de situações relacionadas à segurança. Estes planos constituem elementos essenciais na estrutura organizacional,

fornecendo diretrizes claras e protocolos específicos para lidar com incidentes de segurança de maneira ágil e eficiente.

A implementação de IRPs não apenas aprimora a capacidade de resposta diante de incidentes, mas também contribui para a mitigação de potenciais danos e a minimização de impactos negativos. Tais planos devem ser elaborados levando em consideração a natureza específica das ameaças enfrentadas pela organização, abrangendo aspectos como detecção precoce, análise de incidentes, comunicação eficiente, contenção de danos e restauração de operações normais.

Além disso, é recomendável que os IRPs sejam regularmente revisados e atualizados para refletir as mudanças no cenário de ameaças, bem como para incorporar lições aprendidas a partir de incidentes anteriores. Essa abordagem proativa assegura que a organização esteja preparada para enfrentar novos desafios e que seus processos de resposta evoluam em conformidade com a dinâmica do panorama de segurança cibernética.

Um exemplo da falta de um plano IRP, foi no dia 04 de novembro de 2021, de acordo com Narciso, F. A., & Marin, I. (2021), serviços oferecidos pela empresa Meta, como whatsapp, facebook e instagram tiveram seus serviços interrompidos por uma falha de manutenção de rotina, que acabou por inviabilizar todos os serviços virtuais tanto externos como internos da empresa, a partir do momento que os programadores foram ter acesso físico aos servidores para corrigir a falha, houve um grande problema ao adentrar a área, meios de segurança como catracas e portas estavam totalmente bloqueadas, pois elas funcionavam com o mesmo sistema que havia sido afetado pela falha, isso também contribuiu para aumentar ainda mais o tempo de contra-resposta ao problema. Depois que a falha foi sanada, a meta além de muitas novas tomadas de decisões para que acidentes assim não acontecessem futuramente, relatou sobre novos métodos para que as pessoas autorizadas possam ter acesso aos datacenters sem que necessitem de sistemas eletrônicos. Esse caso seria bem diferente se tal IRP em questão fosse tomada antes de uma falha como essa acontecesse.

2.3.3 Normas governamentais

A conformidade com leis e regulamentos, como o GDPR na União Europeia, é uma preocupação relevante, garantindo a proteção da privacidade e segurança dos dados. Em suma, a segurança cibernética demanda uma abordagem proativa e em camadas, uma vez que as organizações enfrentam ameaças em constante evolução no cenário digital e prover a privacidade dos dados dos usuários. O não cumprimento dessas regulamentações deve resultar em penalidades, servindo de exemplo e intimidando àqueles que queiram realizar ações criminosas nos ambientes digitais.

2.3.4 Firewalls

Firewalls, em termos de segurança cibernética, são elementos essenciais na defesa de redes de computadores contra ameaças maliciosas. Estes dispositivos, que podem ser tanto hardware quanto software, atuam como barreiras de proteção, monitorando e controlando o tráfego de dados entre redes distintas. A principal função de um firewall é estabelecer políticas de segurança que determinam quais comunicações são permitidas ou bloqueadas, com base em critérios predefinidos.

Ao inspecionar o tráfego de entrada e saída, os firewalls desempenham um papel crítico na prevenção de acessos não autorizados, ataques maliciosos e na identificação de padrões suspeitos. Esses dispositivos são configurados para analisar pacotes de dados, determinando se eles atendem aos critérios estabelecidos nas políticas de segurança. Os firewalls podem oferecer funcionalidades avançadas, como a criação de redes privadas virtuais (VPNs - Virtual Private Network), filtragem de conteúdo e a detecção/prevenção de intrusões. E em um contexto empresarial, os firewalls desempenham um papel crucial na proteção de informações sensíveis, garantindo a conformidade com regulamentações criadas pela empresa e assim preservando a integridade das operações online.

2.3.5 Ferramentas de escaneamento

Ferramentas de escaneamento de aplicações para segurança cibernética representam elementos fundamentais na proteção de sistemas digitais contra vulnerabilidades potenciais, principalmente pelo fato de sua automação e rapidez. Estas ferramentas são projetadas para examinar minuciosamente o código e a infraestrutura de aplicações, identificando possíveis pontos de fragilidade que poderiam ser explorados por ameaças cibernéticas. O escopo dessas ferramentas abrange a detecção de falhas de segurança, brechas de autenticação, injeção de código, entre outras vulnerabilidades conhecidas. Muitas ferramentas de escaneamento oferecem funcionalidades avançadas, incluindo a análise estática e dinâmica do código-fonte, a fim de aprimorar a precisão na identificação de possíveis ameaças. Existem várias ferramentas de escaneamento de aplicações amplamente utilizadas na segurança cibernética, alguns exemplos são: Nessus, acunetix, burp Suite, e as duas ferramenta que usaremos neste trabalho é a OWASP ZAP (Zed Attack Proxy) e o NMAP. O OWASP ZAP é uma ferramenta de código aberto focada em encontrar vulnerabilidades em aplicações da web. O ZAP pode ajudar na identificação de problemas de segurança, como injeção de SQL, cross-site scripting (XSS) e cross-site request forgery (CSRF). E o Nmap (Network Mapper): embora seja conhecida por varreduras de rede, o Nmap também pode ser usado para avaliar a segurança de aplicações, identificando portas abertas e serviços em execução.

2.3.6 Criptografia

A criptografia é uma disciplina fundamental na segurança da informação, envolvendo a técnica de converter dados em uma forma ilegível, chamada de texto cifrado, a fim de proteger a confidencialidade, integridade e autenticidade das informações. Ela desempenha um papel crucial em uma variedade de contextos, desde comunicações online até armazenamento seguro de dados.

Um exemplo notável de criptografia é a criptografia de chave simétrica, onde a mesma chave é utilizada tanto para cifrar quanto para decifrar as informações. Isso

é eficaz para comunicações ponto a ponto, mas levanta desafios em termos de distribuição segura de chaves.

Outro exemplo é a criptografia de chave assimétrica, que envolve um par de chaves: uma chave pública, usada para cifrar informações, e uma chave privada, usada para decifrar. Essa abordagem é valiosa para garantir a autenticidade e a integridade em comunicações mais amplas, como transações online.

A criptografia de ponta a ponta em mensagens instantâneas e e-mails é um terceiro exemplo, garantindo que apenas os destinatários autorizados possam acessar o conteúdo das mensagens, mesmo se os dados forem interceptados durante a transmissão.

2.3.7 CSP

O CSP - Content Security Policy, Política de Segurança de Conteúdo, é uma medida de segurança que adiciona uma camada adicional de proteção aos sites. Ela ajuda a detectar e mitigar ataques que podem ser usados para roubar dados, desfigurar sites ou distribuir malware. O CSP funciona declarando quais fontes de conteúdo são aprovadas para serem carregadas em uma página. Os navegadores devem seguir essas declarações e bloquear o carregamento de conteúdo de fontes não aprovadas. Os tipos de conteúdo que podem ser protegidos pelo CSP incluem: JavaScript, ActiveX, arquivos de áudio e vídeo, CSS, imagens e objetos incorporados, quadros HTML, fontes, como applets Java.

2.4 Ameaças cibernéticas

As ameaças cibernéticas englobam uma série de ataques e atividades maliciosas direcionadas à exploração de vulnerabilidades em sistemas computacionais, redes, dispositivos e dados. O propósito subjacente dessas ações é obter acesso não autorizado, provocar danos ou perpetrar o roubo de informações sensíveis. À medida que a sociedade contemporânea intensifica sua dependência da tecnologia da informação, as ameaças cibernéticas emergem como uma inquietação de magnitude expressiva, demandando atenção e medidas preventivas substanciais. Vejamos alguns exemplos de ameaças

2.4.1 Malware

Malwares, uma contração do inglês “malicious softwore”, ou seja software malicioso, são programas maliciosos projetados para infectar dispositivos e redes. De acordo com Firmino (2019), os principais tipos de ataques são estes códigos maliciosos. A principal característica dos malwares é sua capacidade de se infiltrar em sistemas ou dispositivos sem o conhecimento ou consentimento do usuário, muitas vezes explorando falhas de segurança. Uma vez dentro, esses programas maliciosos podem executar uma variedade de ações prejudiciais, como roubo de informações, danificação de arquivos, interrupção de operações normais do sistema, espionagem e muito mais. Os malwares são frequentemente distribuídos por meio de técnicas como anexos de e-mail, links maliciosos, downloads infectados e exploração de vulnerabilidades em software. A eficaz contenção de malwares demanda a implementação de medidas de segurança proativas, incorporando a utilização de antivírus, a configuração de firewalls, a manutenção de atualizações regulares de software e a adoção de práticas seguras de navegação na internet. Vejamos os malwares mais conhecidos.

Vírus: Um vírus é um programa maligno que se anexa a outros programas ou arquivos legítimos e se replica quando estes programas são executados. Os vírus são conhecidos por sua capacidade de se espalhar e infectar outros arquivos, programas ou até mesmo sistemas inteiros, seu principal objetivo é usar recursos computacionais ao máximo, como memória e processamento. Podem ser transmitidos por meio de dispositivos de armazenamento removíveis, downloads da internet, arquivos e programas infectados ou anexos de e-mail.

Trojan (Cavalo de Troia): Trojan ou conhecido também como cavalo de troia é um tipo de malware que se disfarça como um software legítimo para enganar os usuários e infiltrar-se em sistemas sem o conhecimento do usuário. O objetivo mais utilizado pelos trojans é abrir portas para outros malwares entrarem. Eles podem roubar informações, criar backdoors para acesso remoto ou causar danos variados, dependendo de qual objetivo o atacante queira com esse cavalo de troia digital.

Ransomware: é um acrônimo de “ranson software” ou seja ,programa sequestrador em portugês, esse é um tipo de malware que criptografa os dados de uma vítima, tornando-os inacessíveis, e exige um resgate em troca da chave de descriptografia.

Worms: Os worms, vermes em portugês, são agentes autônomos de propagação que exploram vulnerabilidades em sistemas e redes para disseminar-se rapidamente. Sua capacidade de auto multiplicação, ausente da necessidade de um programa hospedeiro, torna-os particularmente eficientes na transmissão e comprometimento de sistemas em larga escala.

Spyware: Acrônimo para spy software, programa espião, caracteriza-se pela sua finalidade de monitorar as atividades dos usuários, clandestinamente coletando informações pessoais sem consentimento. Seu meio de agir não é como o do vírus, que é totalmente perceptível por estar acessando recursos e provocando lentidão da máquina hospedeira, no caso do spyware, sua finalidade é ser o mais indetectável possível. Esta categoria de malware pode comprometer dados bancários, senhas e outras informações confidenciais, representando uma ameaça à privacidade e à segurança digital.

Adware: O adware, embora não seja diretamente prejudicial, exibe anúncios indesejados em dispositivos infectados, podendo afetar o desempenho do sistema. Sua presença intrusiva, no entanto, é extremamente incômoda e compromete a experiência do usuário.

Rootkits: Os rootkits são formas avançadas de malware que visam ocultar a presença de outros agentes maliciosos em um sistema operacional. Esses programas são projetados para operar de maneira furtiva, camuflando-se profundamente no núcleo do sistema e dificultando sua detecção por ferramentas de antivírus convencionais. Uma vez instalados, os rootkits podem manipular até mesmo o kernel do sistema, fornecendo acesso privilegiado ao atacante e permitindo a execução de atividades não autorizadas. Sua habilidade de integrar-se às camadas mais profundas do sistema operacional torna-os particularmente desafiadores de serem identificados e removidos, conferindo-lhes uma natureza insidiosa. A presença de rootkits representa uma séria ameaça à integridade e segurança do sistema, exigindo medidas proativas e especializadas para sua detecção e mitigação eficazes.

Backdoors: Os backdoors são elementos de software que buscam por portas de acesso intencionalmente deixadas abertas por desenvolvedores em um sistema, comumente para fins de manutenção. No entanto, quando exploradas por atacantes, essas portas tornam-se pontos de entrada não autorizados, e isso compromete a segurança do sistema. A presença de backdoors pode resultar em acessos não permitido e atividades maliciosas, conferindo aos invasores a capacidade de contornar medidas convencionais de segurança e expor sistemas para outros tipos de ameaça possam adentrar no ambiente atacado utilizando estas portas abertas.

2.4.2 cross-site request forgery

O Cross-Site Request Forgery (CSRF) é uma vulnerabilidade de segurança em aplicações web que ocorre quando um atacante engana um usuário autenticado para executar ações não intencionais em uma aplicação na qual o usuário está autenticado. Em um ataque CSRF, o atacante induz o usuário a realizar uma ação, como clicar em um link ou carregar uma imagem, sem o conhecimento do usuário.

O cenário típico envolve a exploração da confiança que uma aplicação deposita no navegador do usuário. Se o usuário estiver autenticado em um site e, ao mesmo tempo, visitar um site malicioso que contenha código CSRF, a aplicação maliciosa pode enviar solicitações não autorizadas em nome do usuário autenticado para a aplicação legítima. Para mitigar o CSRF, as aplicações web geralmente implementam medidas de segurança, como tokens anti-CSRF (também conhecidos como tokens synchronizer) que são incorporados aos formulários da web. Esses tokens são únicos para cada sessão e precisam ser incluídos em todas as solicitações que modificam o estado do servidor, garantindo que a solicitação seja legítima.

O OWASP (Open Web Application Security Project) considera o CSRF como uma das 10 principais ameaças de segurança em aplicações web e destaca a importância de implementar práticas adequadas para prevenir esse tipo de ataque.

2.4.3 Header Injection

Neste tipo de ataque, os invasores manipulam cabeçalhos de solicitação HTTP para inserir conteúdo malicioso, como scripts ou redirecionamentos, que podem impactar adversamente na segurança.

2.4.4 Cross-Site Scripting (XSS)

Neste tipo de ataque, o invasor insere scripts maliciosos (geralmente em JavaScript) em páginas web visualizadas por outros usuários, possibilitando a aquisição de informações sensíveis, como cookies de autenticação, ou a execução de ações não autorizadas em nome do usuário.

2.4.5 Injeção de Comandos

Paralelamente ao SQL injection, existem outras formas de injeção, como Command Injection. Este ataque ocorre quando um aplicativo aceita entradas não confiáveis e as executa como comandos, permitindo que um invasor execute comandos arbitrários no sistema.

2.4.6 Clickjacking

Clickjacking, que vem da junção de click(clique) hijacking(sequestro), também conhecido como "UI redirecionamento" ou "User Interface redirecionamento",é uma técnica maliciosa empregada em ataques cibernéticos para enganar os usuários, levando-os a clicar em elementos da interface do usuário sem seu conhecimento. Essa abordagem explora a sobreposição de elementos na página web, utilizando a transparência de certos elementos para manipular as ações do usuário.

No clickjacking, um elemento web legítimo, como um botão ou link, é ocultado por trás de um elemento aparentemente inofensivo, como um banner publicitário ou

uma imagem. O usuário, ao clicar aparentemente no elemento inofensivo, acaba clicando no elemento oculto, realizando ações indesejadas.

Os objetivos desse tipo de ataque variam e podem incluir o redirecionamento de cliques do usuário para realizar ações não autorizadas, como interações em redes sociais, transações financeiras ou engajamento com elementos sensíveis da página. Além disso, o clickjacking pode ser utilizado para induzir o usuário a baixar ou instalar malware no dispositivo.

2.4.7 SlowLoris

O ataque envolve o envio lento e prolongado de requisições HTTP para um servidor, mantendo cada conexão aberta por um longo período de tempo sem completar totalmente a solicitação, a principal ideia por trás do Slowloris é manter várias conexões abertas simultaneamente, esgotando assim os recursos disponíveis do servidor, como os slots de conexão. Isso pode resultar em uma negação de serviço, onde o servidor fica incapaz de atender a novas solicitações legítimas devido à ocupação de seus recursos por conexões maliciosas persistentes.

2.4.8 MIME sniffing

Os ataques do tipo MIME sniffing, também conhecidos como "Content-Type sniffing", constituem uma abordagem na qual navegadores da web tentam inferir o tipo de conteúdo de um arquivo, independentemente das informações fornecidas pelos cabeçalhos Content-Type dos servidores. Este comportamento é adotado quando o servidor não fornece ou fornece informações inconsistentes sobre o tipo MIME do conteúdo. Em tal contexto, navegadores podem realizar uma análise heurística para determinar a natureza do arquivo, o que pode ser explorado por invasores para manipular a execução de código malicioso. Por exemplo, um arquivo JavaScript camuflado como uma imagem pode ser interpretado erroneamente pelo navegador, resultando em execução indesejada de scripts potencialmente prejudiciais.

2.5 penetration test

O teste de penetração, conhecido como penetration test, é uma prática essencial em segurança cibernética que busca avaliar proativamente a resistência dos sistemas e redes de uma organização a possíveis ataques. Este procedimento envolve a simulação controlada de ataques a fim de identificar e analisar vulnerabilidades que poderiam ser exploradas por agentes maliciosos. Durante o penetration test, diversas técnicas avançadas e ferramentas específicas são aplicadas para avaliar a segurança de redes, aplicativos e infraestrutura. Essas práticas incluem análise de código, testes de phishing, exploração de vulnerabilidades e avaliação das configurações de segurança. O resultado é a produção de um relatório abrangente que destaca as deficiências encontradas, acompanhadas de recomendações detalhadas para mitigação ou correção.

Conduzido por profissionais especializados, os "ethical hackers" ou "pentesters", o profissional pentester atua como um arquiteto de defesa cibernética, empregando uma abordagem técnica e ética para identificar, explorar e documentar as vulnerabilidades presentes nos sistemas. Seu trabalho não se limita apenas à aplicação de testes automáticos; ele também realiza análises aprofundadas, utilizando sua expertise para simular cenários de ataque real e explorar possíveis pontos de fragilidade.

Os métodos do pentester englobam uma ampla gama de técnicas, desde a análise estática e dinâmica de código-fonte até a execução de testes de intrusão e a avaliação de configurações de segurança. O profissional deve manter-se atualizado com as mais recentes tendências em cibersegurança, compreendendo as novas ameaças e vulnerabilidades emergentes. Os profissionais pentesters podem realizar testes de penetração em diferentes níveis de conhecimento sobre os sistemas a serem avaliados. Esses níveis de conhecimento são geralmente classificados como White Box, Black Box e Grey Box.

- Pentesting White Box: Neste cenário, o pentester possui total conhecimento sobre a infraestrutura e os sistemas a serem avaliados. Esse conhecimento inclui detalhes internos, como diagramas de rede, códigos-fonte e informações privilegiadas. Esse tipo de teste permite uma análise mais profunda e específica das vulnerabilidades e é ideal para se entender o mais

profundo possível quando se busca uma avaliação detalhada e completa da segurança.

- Pentesting Black Box: No teste Black Box, o pentester não possui conhecimento prévio sobre a infraestrutura alvo. Isso simula um cenário mais próximo do que um atacante externo enfrentaria. Essa abordagem testa a capacidade do profissional de identificar e explorar vulnerabilidades sem informações privilegiadas. O pentester parte do zero, realizando uma avaliação externa do sistema e usando até mesmo técnicas de engenharia social, tentando escalar o máximo possível de seus privilégios e realizar o máximo de operações para entender o que um usuário externo tem como meios para um cenário deste ataque.
- Pentesting Grey Box: O teste Grey Box combina elementos dos métodos White Box e Black Box. O pentester tem um conhecimento parcial do ambiente, proporcionando uma visão equilibrada entre a perspectiva interna e externa. Essa abordagem pode ser útil quando se deseja simular ameaças de alguém com algum nível de acesso interno, como um usuário autenticado ou colaboradores. O Grey Box Testing busca encontrar um equilíbrio entre a profundidade de um teste White Box e ao mesmo tempo a perspectiva de um Black Box.

3. METODOLOGIA

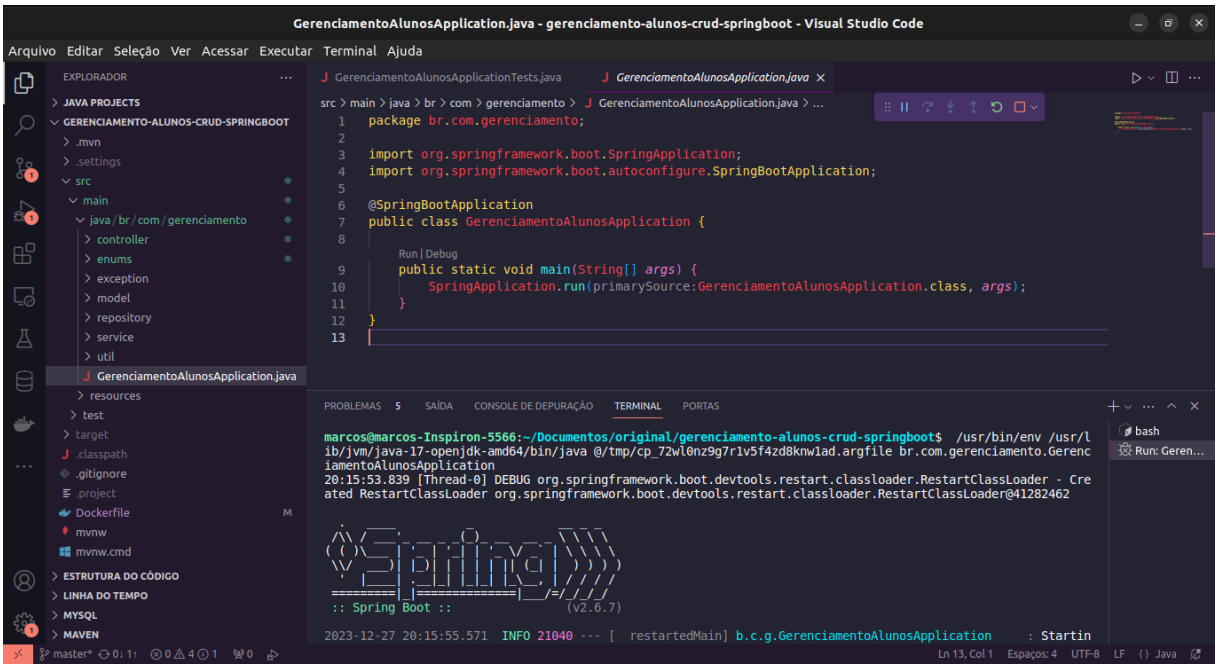
3.1 justificativa

Como foi escolhido um cenário mais próximo do real, a aplicação em questão é um CRUD, ou seja, um sistema com as quatro operações fundamentais realizadas em sistemas que interagem com bancos de dados, criar, ler, atualizar e deletar respectivamente. O sistema em questão utiliza o framework spring boot e a sua linguagem Java, e a metodologia do penetration test foi o white box, onde há conhecimento total da aplicação.

3.2 seleção de amostras

O sistema selecionado foi um já de conhecimento do autor, presente no site de repositórios remotos github (https://github.com/juliuscavalcante/gerepelo framework nciamento-alunos-crud-springboot), e podemos ver sua inicialização na figura 3.

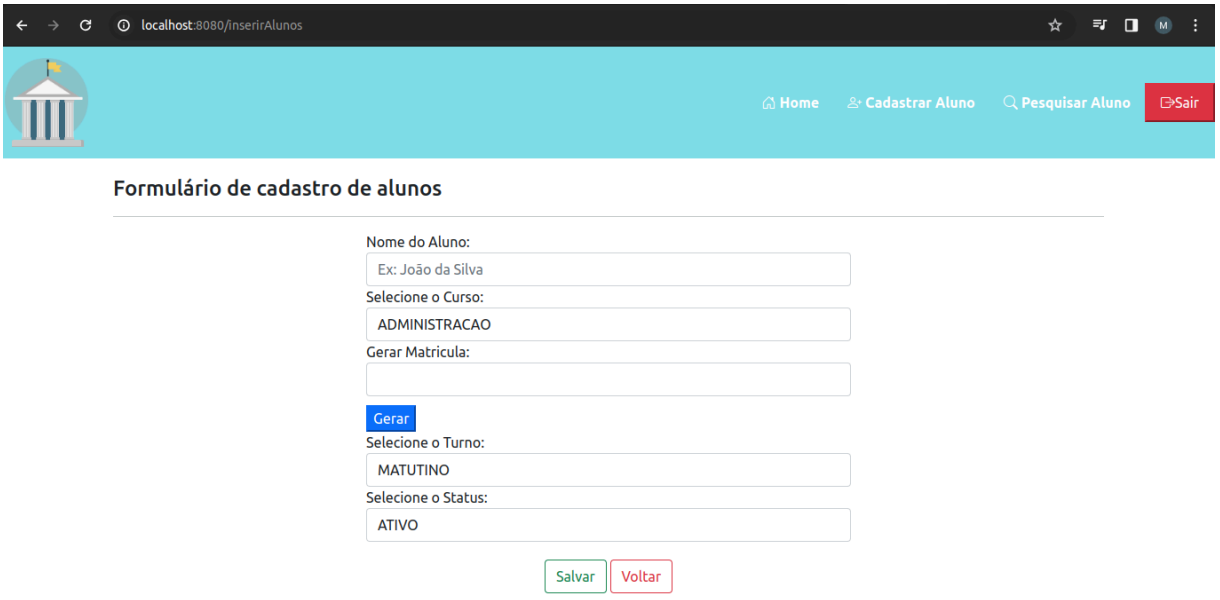
Figura 3 - Inicialização do projeto spring boot



Fonte: Elaborado pelo autor

Após algumas correções e atualizações de versão e banco de dados, ela ficou pronta para ser executada. Esta contava com funcionalidades de criação de usuários, cadastro de alunos, assim como a listagem e atualização destes, vejamos na figura 4.

Figura 4 - Tela de cadastro de alunos



Fonte: Elaborado pelo autor

3.3 Ferramentas utilizadas

Foi utilizado neste estudo as ferramentas OWASP ZAP e NMAP, ambas as ferramentas, são amplamente empregadas em atividades de segurança cibernética e teste de penetração. Enquanto o ZAP foca em aplicações web, o NMAP concentra-se na identificação de sistemas e serviços em uma rede. A integração dessas ferramentas em práticas de segurança ajuda a identificar e corrigir vulnerabilidades.

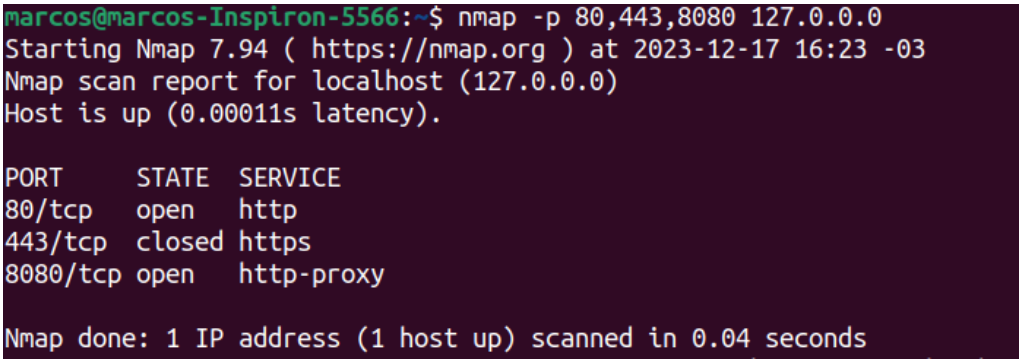
4. RESULTADOS

Será analisado agora o resultado das ferramentas utilizadas e propor a mitigação dos riscos que estas retornam.

4.2.1 NMAP

O primeiro comando a ser usado no terminal do sistema operacional Linux Ubuntu versão 22.04 foi o “*nmap -p 80,443,8080 127.0.0.0” ,* esse comando foi utilizado para verificar se as portas padrões de conexão com aplicações estão sendo utilizadas.

Figura 5 - Escaneando com *nmap -p 80,443,8080 127.0.0.0*



Fonte: Elaborado pelo autor

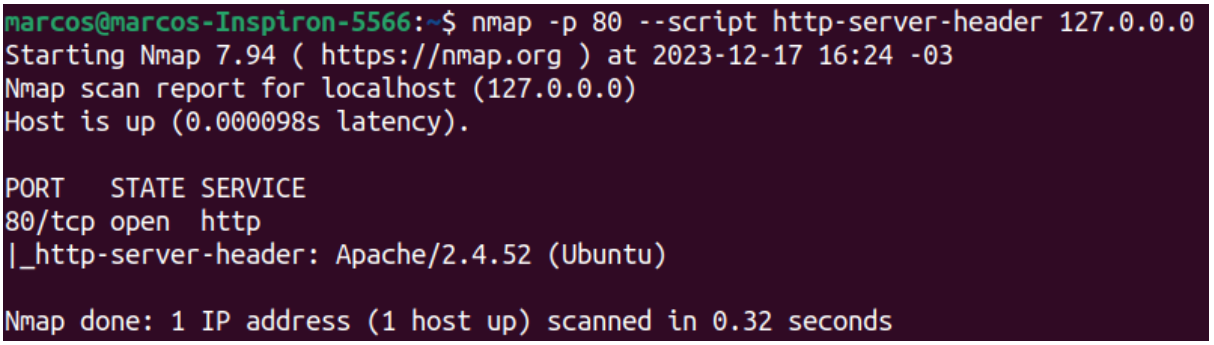
O que podemos ver a partir dos resultados presentes na imagem 6, o é que as portas 80, 443 e 8080, estão com status “open” “closed” e “open”, estas estão comumente associada a serviços web usando o protocolo *HTTP, HTTPS* e *HTTP-proxy,* estão com o estado "open" “closed” e “open” respectivamente, ou seja, significa que as portas padrões 80 e 8080 estão aceitando conexões HTTP e é onde geralmente onde são hospedados sites ou aplicativos web, como o serviço não usa protocolo HTTPS.

A varredura ajudará a identificar se há servidores web em execução nas portas comuns (80 para HTTP, 443 para HTTPS e 8080 para servidores web alternativos). Vale ressaltar que qualquer conexão que o endereço IP comece com 127, refere-se ao "loopback address", ou endereço de retorno, que é usado para testar a conectividade de rede em um dispositivo. O endereço específico varia de

127.0.0.0 a 127.255.255.255 que é reservada para o loopback. Quando um dispositivo se comunica com o endereço IP 127.0.0.1, ele está se comunicando consigo mesmo, não passando pela rede, para verificar se a pilha TCP/IP está operacional.

O segundo comando foi o "*nmap -p 80 --script HTTP-server-header 127.0.0.0*"

Figura 6 - Escaneando com *nmap -p 80 --script HTTP-server-header 127.0.0.0*



Fonte: Elaborado pelo autor

Esse comando utiliza um script NSE - Nmap Script Engine, para tentar identificar a versão do servidor web em execução na porta 80. O NMAP encontrou a versão apache/2.4.52 (ubuntu), conteúdo da imagem 7, que de acordo com o site "https://httpd.apache.org/security/vulnerabilities_24.html" possui uma vulnerabilidade de nível baixo, on um intruso, ao estabelecer uma conexão HTTP/2 com uma janela de tamanho inicial de 0, tem a capacidade obstruir de maneira indefinida o processamento dessa conexão no servidor Apache HTTP. Tal exploit poderia ser empregado para esgotar os recursos do servidor, assemelhando-se ao padrão de ataque amplamente reconhecido denominado "slow loris".

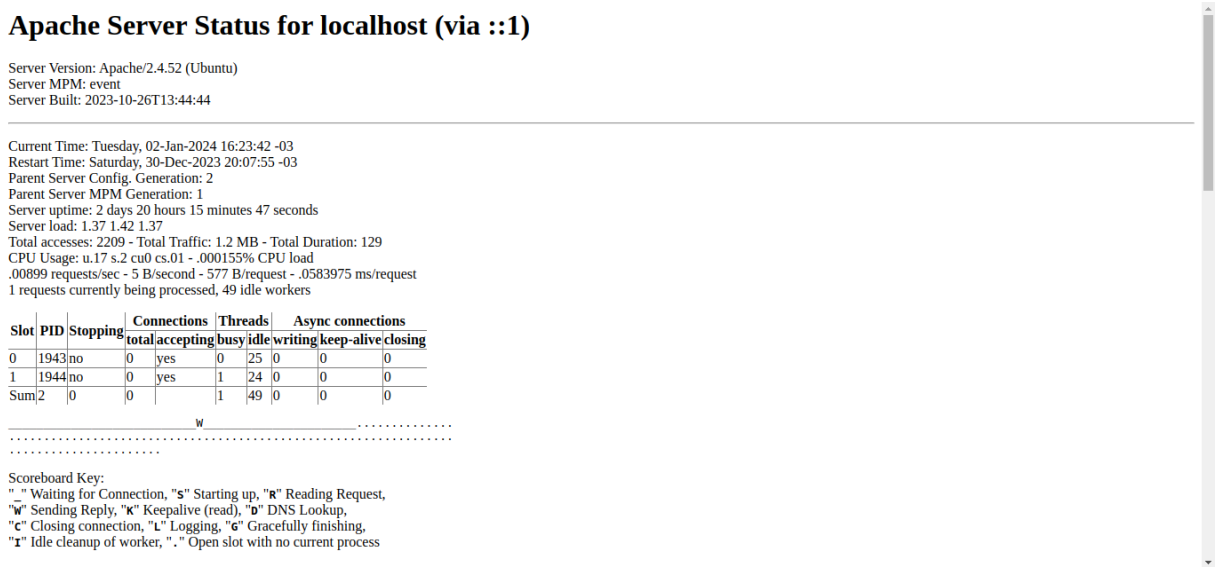
Figura 7 - Escaneando com *nmap -p 80 --script http-enum 127.0.0.0*

```
marcos@marcos-Inspiron-5566:~$ nmap -p 80 --script http-enum 127.0.0.0
Starting Nmap 7.94 ( https://nmap.org ) at 2023-12-17 16:27 -03
Nmap scan report for localhost (127.0.0.0)
Host is up (0.000085s latency).

PORT   STATE SERVICE
80/tcp open  http
| http-enum:
|_  /server-status/: Potentially interesting folder

Nmap done: 1 IP address (1 host up) scanned in 0.63 seconds
```

Fonte: Elaborado pelo autor

Já o terceiro comando foi o “nmap -p 80 --script http-enum 127.0.0.0” esse comando utiliza um script NSE específico para HTTP para realizar enumeração de diretórios, identificar tecnologias utilizadas e obter informações adicionais sobre o servidor web, imagem 8. No resultado em questão, o script http-enum revelou a presença de um diretório chamado “/server-status/”, que pode conter informações relevantes sobre o servidor web. Então acessando a URL com o diretório achado temos a seguinte página HTML:

Figura 8 - Tela do diretório /server-status (parte 1)



Apache Server Status for localhost (via ::1)

Server Version: Apache/2.4.52 (Ubuntu)
Server MPM: event
Server Built: 2023-10-26T13:44:44

Current Time: Tuesday, 02-Jan-2024 16:23:42 -03
Restart Time: Saturday, 30-Dec-2023 20:07:55 -03
Parent Server Config. Generation: 2
Parent Server MPM Generation: 1
Server uptime: 2 days 20 hours 15 minutes 47 seconds
Server load: 1.37 1.42 1.37
Total accesses: 2209 - Total Traffic: 1.2 MB - Total Duration: 129
CPU Usage: u.17 s.2 cu0 cs.01 - .000155% CPU load
.00899 requests/sec - 5 B/second - 577 B/request - .0583975 ms/request
1 requests currently being processed, 49 idle workers

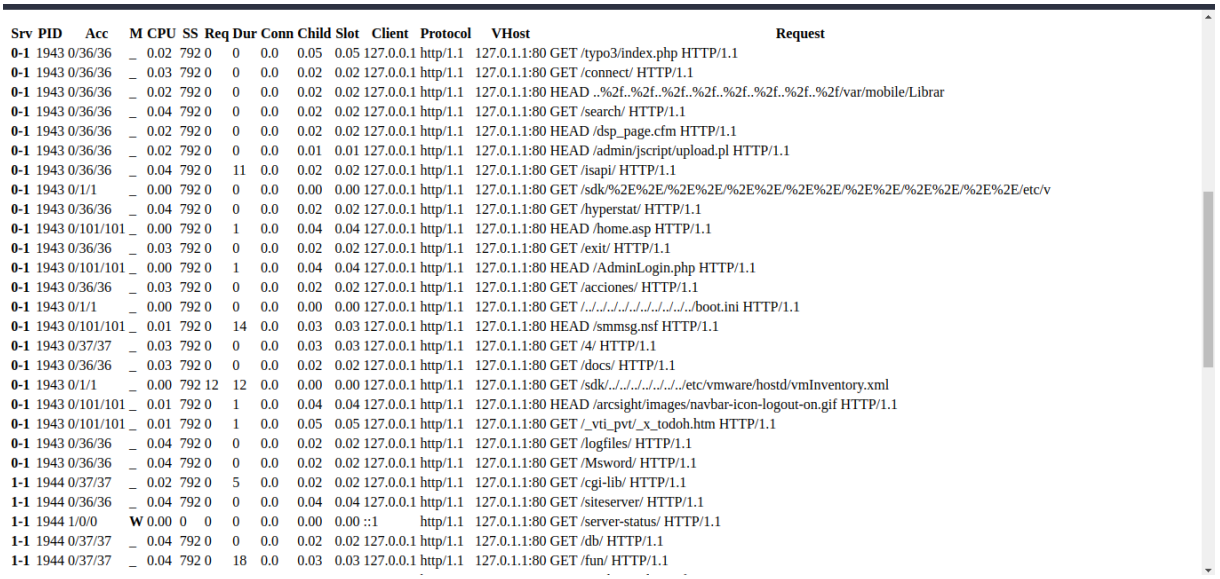
Slot	PID	Stopping	Connections		Threads		Async connections		
			total	accepting	busy	idle	writing	keep-alive	closing
0	1943	no	0	yes	0	25	0	0	0
1	1944	no	0	yes	1	24	0	0	0
Sum	2	0	0		1	49	0	0	0

_____W_____.....
.....
.....

Scoreboard Key:
"_" Waiting for Connection, "S" Starting up, "R" Reading Request,
"W" Sending Reply, "K" Keepalive (read), "D" DNS Lookup,
"C" Closing connection, "L" Logging, "G" Gracefully finishing,
"I" Idle cleanup of worker, "." Open slot with no current process

Fonte: Elaborado pelo autor

Figura 9 - Tela do diretório /server-status (parte 2)



Srv	PID	Acc	M	CPU	SS	Req	Dur	Conn	Child	Slot	Client	Protocol	VHost	Request
0-1	1943	0/36/36	_	0.02	792	0	0	0.0	0.05	0.05	127.0.0.1	http/1.1	127.0.1.1:80	GET /typo3/index.php HTTP/1.1
0-1	1943	0/36/36	_	0.03	792	0	0	0.0	0.02	0.02	127.0.0.1	http/1.1	127.0.1.1:80	GET /connect/ HTTP/1.1
0-1	1943	0/36/36	_	0.02	792	0	0	0.0	0.02	0.02	127.0.0.1	http/1.1	127.0.1.1:80	HEAD ..%2f..%2f..%2f..%2f..%2f..%2f..%2f..%2f/var/mobile/Librar
0-1	1943	0/36/36	_	0.04	792	0	0	0.0	0.02	0.02	127.0.0.1	http/1.1	127.0.1.1:80	GET /search/ HTTP/1.1
0-1	1943	0/36/36	_	0.02	792	0	0	0.0	0.02	0.02	127.0.0.1	http/1.1	127.0.1.1:80	HEAD /dsp_page.cfm HTTP/1.1
0-1	1943	0/36/36	_	0.02	792	0	0	0.0	0.01	0.01	127.0.0.1	http/1.1	127.0.1.1:80	HEAD /admin/jscript/upload.pl HTTP/1.1
0-1	1943	0/36/36	_	0.04	792	0	11	0.0	0.02	0.02	127.0.0.1	http/1.1	127.0.1.1:80	GET /isapi/ HTTP/1.1
0-1	1943	0/1/1	_	0.00	792	0	0	0.0	0.00	0.00	127.0.0.1	http/1.1	127.0.1.1:80	GET /sdk/%2E%2E/%2E%2E/%2E%2E/%2E%2E/%2E%2E/%2E%2E/%2E%2E/%2E%2E/%2E%2E/etc/v
0-1	1943	0/36/36	_	0.04	792	0	0	0.0	0.02	0.02	127.0.0.1	http/1.1	127.0.1.1:80	GET /hyperstat/ HTTP/1.1
0-1	1943	0/101/101	_	0.00	792	0	1	0.0	0.04	0.04	127.0.0.1	http/1.1	127.0.1.1:80	HEAD /home.asp HTTP/1.1
0-1	1943	0/36/36	_	0.03	792	0	0	0.0	0.02	0.02	127.0.0.1	http/1.1	127.0.1.1:80	GET /exit/ HTTP/1.1
0-1	1943	0/101/101	_	0.00	792	0	1	0.0	0.04	0.04	127.0.0.1	http/1.1	127.0.1.1:80	HEAD /AdminLogin.php HTTP/1.1
0-1	1943	0/36/36	_	0.03	792	0	0	0.0	0.02	0.02	127.0.0.1	http/1.1	127.0.1.1:80	GET /acciones/ HTTP/1.1
0-1	1943	0/1/1	_	0.00	792	0	0	0.0	0.00	0.00	127.0.0.1	http/1.1	127.0.1.1:80	GET /../../../../../../../../../boot.ini HTTP/1.1
0-1	1943	0/101/101	_	0.01	792	0	14	0.0	0.03	0.03	127.0.0.1	http/1.1	127.0.1.1:80	HEAD /smmsg.nsf HTTP/1.1
0-1	1943	0/37/37	_	0.03	792	0	0	0.0	0.03	0.03	127.0.0.1	http/1.1	127.0.1.1:80	GET /4/ HTTP/1.1
0-1	1943	0/36/36	_	0.03	792	0	0	0.0	0.02	0.02	127.0.0.1	http/1.1	127.0.1.1:80	GET /docs/ HTTP/1.1
0-1	1943	0/1/1	_	0.00	792	12	12	0.0	0.00	0.00	127.0.0.1	http/1.1	127.0.1.1:80	GET /sdk/../../../../../../../etc/vmware/hostd/vmInventory.xml
0-1	1943	0/101/101	_	0.01	792	0	1	0.0	0.04	0.04	127.0.0.1	http/1.1	127.0.1.1:80	HEAD /arcsight/images/navbar-icon-logout-on.gif HTTP/1.1
0-1	1943	0/101/101	_	0.01	792	0	1	0.0	0.05	0.05	127.0.0.1	http/1.1	127.0.1.1:80	GET /_vti_pvt/_x_todoh.htm HTTP/1.1
0-1	1943	0/36/36	_	0.04	792	0	0	0.0	0.02	0.02	127.0.0.1	http/1.1	127.0.1.1:80	GET /logfiles/ HTTP/1.1
0-1	1943	0/36/36	_	0.04	792	0	0	0.0	0.02	0.02	127.0.0.1	http/1.1	127.0.1.1:80	GET /Msword/ HTTP/1.1
1-1	1944	0/37/37	_	0.02	792	0	5	0.0	0.02	0.02	127.0.0.1	http/1.1	127.0.1.1:80	GET /cgi-lib/ HTTP/1.1
1-1	1944	0/36/36	_	0.04	792	0	0	0.0	0.04	0.04	127.0.0.1	http/1.1	127.0.1.1:80	GET /siteserver/ HTTP/1.1
1-1	1944	1/0/0	W	0.00	0	0	0	0.0	0.00	0.00	::1	http/1.1	127.0.1.1:80	GET /server-status/ HTTP/1.1
1-1	1944	0/37/37	_	0.04	792	0	0	0.0	0.02	0.02	127.0.0.1	http/1.1	127.0.1.1:80	GET /db/ HTTP/1.1
1-1	1944	0/37/37	_	0.04	792	0	18	0.0	0.03	0.03	127.0.0.1	http/1.1	127.0.1.1:80	GET /fun/ HTTP/1.1

Fonte: Elaborado pelo autor

Expor o diretório /server-status/, como mostrado nas figuras 9 e 10, de uma aplicação acarreta riscos significativos à segurança, pois fornece informações detalhadas sobre o estado interno do servidor web, como o Apache. Entre os problemas associados a tal exposição, destaca-se ,por exemplo, o vazamento de informações sensíveis, revelando detalhes sobre o servidor, como sua versão, módulos carregados e estatísticas de requisições, e a exposição desse diretório pode facilitar a obtenção de dados sensíveis, como endereços IP de usuários e URLs acessadas, comprometendo a privacidade dos mesmos.

O acesso público ao /server-status/ também pode ser explorado para a realização de ataques de força bruta ou amplificação, envolvendo tentativas repetidas de acesso à página ou a utilização de suas informações para ataques direcionados.

Ademais, a visibilidade do /server-status/ pode ser explorada para conduzir ataques de enumeração, permitindo a coleta de informações valiosas sobre a infraestrutura e serviços em operação.

A divulgação de detalhes sobre a versão do servidor e módulos carregados pode expor o sistema a um risco maior de exploração de vulnerabilidades conhecidas associadas a essas versões específicas.

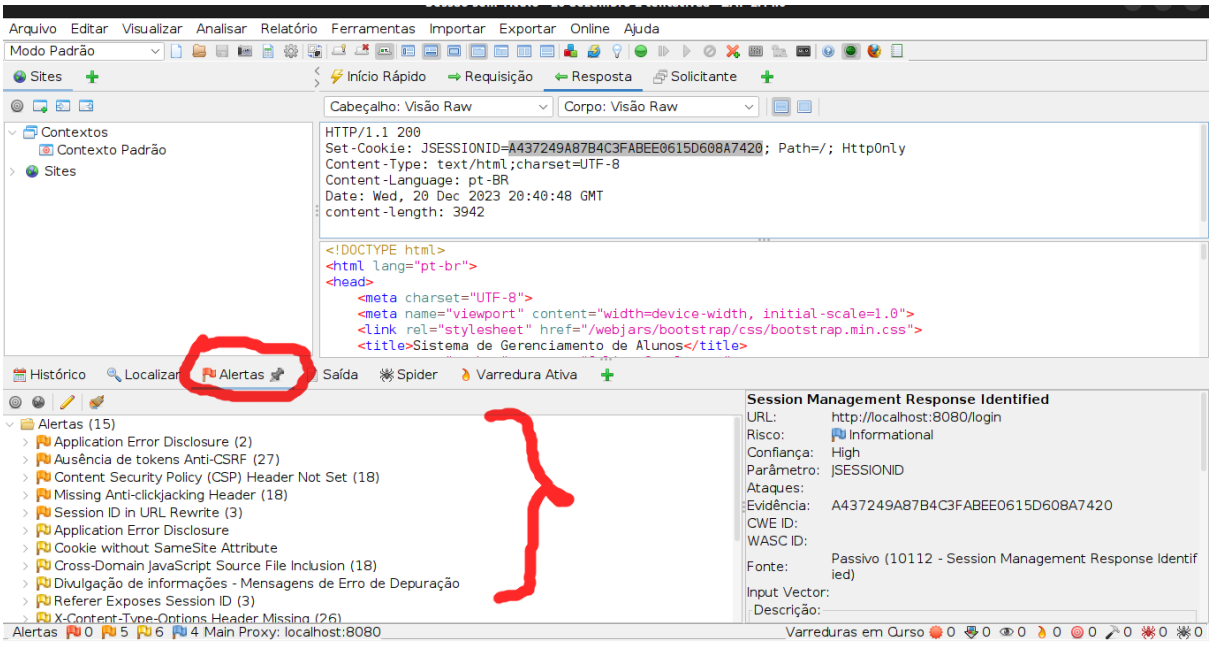
O acesso ao /server-status/ deve ser exclusivamente a administradores do sistema que sejam autorizados, podendo ser por meio de configurações apropriadas

no servidor web, como autenticação. A exposição pública desse diretório deve ser mitigada em ambientes de produção.

4.2.2 OWASP ZAP

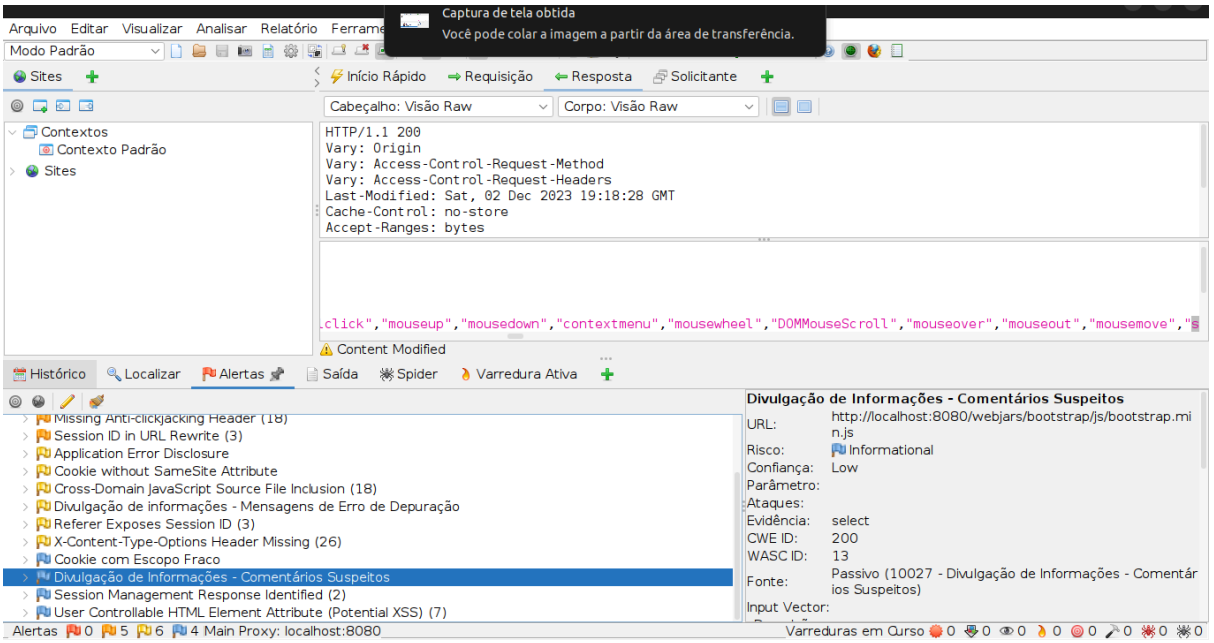
Ao executar a ferramenta, foi escaneado o endereço IP da aplicação, e um de seus métodos é achar campos em que permite a injeção de códigos, e ao detectar esses campos, ele tenta utilizar comando de injeção de código como, /etc/passwd, <!--#EXEC cmd="ls /"-->, </td><scrIpt>alert(1);</scRipt><td>, ZAP AND 1=1 –, ZAP;cat /etc/passwd; zj{#3562*6700}zj, #{%x(sleep 2)}, {system("sleep 2")}, etcetera.
Embora os comandos de injeção de código tenham falhado, outras varreduras foram feitas, e ocasionou nos seguintes resultados a respeito dos alertas da aplicação

Figura 10 - Tela da ferramenta OWASP ZAP (parte 1)



Fonte: Elaborada pelo autor

Figura 11 Tela da ferramenta OWASP ZAP (parte 2)



Fonte: Elaborada pelo autor

No momento que a ferramenta de escaneamento de vulnerabilidades OWASP ZAP terminou seu escaneamento padrão, imagem 11 e 12, ela entregou os seguintes resultados.

I. Application error disclosure
   A. Resumo: Este alerta contém uma mensagem que pode ser de erro ou de aviso que pode divulgar informações sensíveis, como por exemplo a localização do arquivo que produziu a exceção que não está tratada.
   B. Solução: Revisar o código fonte desta página e implementar páginas de erro personalizadas. É de boa prática utilizar identificadores de erro.

II. Ausência de Tokens Anti-CSRF
   A. Resumo: O ID da sessão pode ser armazenado no histórico do navegador ou nos logs do servidor. A reescrita de URL é usada para rastrear o ID da sessão do usuário.
   B. Solução: Colocar o ID da sessão em cookie.

III. Content security policy (CSP) header not set
   A. Resumo: Não foi configurado a CSP

   B. Solução: Certificar de que o servidor web está configurado para definir o cabeçalho Content-Security-Policy.

IV. Missing anti-clicking header
   A. Resumo: Não inclui o código de segurança com a diretiva X-Frame-Options para proteção contra ataques ‘ClickJacking’.
   B. Solução: Para a página não ser enquadrada, deve-se usar a opção DENY na configuração do X-Frame, ou SAMEORIGIN.

V. Session ID in URL rewrite
   A. Resumo: O Session ID (ID de Sessão) está sendo transmitido na URL, o que é vulnerável a ataques de captura de URL, resultando em potencial comprometimento da sessão do usuário.
   B. Solução: Configurar a aplicação para usar cookies seguros e de maneira apropriada para armazenar o Session ID.

VI. Application error disclosure
   A. Resumo: A aplicação está revelando detalhes excessivos sobre erros, como mensagens de erro completas, stack traces, ou outras informações sensíveis, o que pode ser explorado por atacantes para obter insights sobre a infraestrutura e possíveis pontos de vulnerabilidade.
   B. Solução: Personalize as mensagens de erro para fornecer informações mínimas e genéricas. Configure a aplicação para registrar os detalhes completos dos erros no lado do servidor, mas evite exibi-los diretamente aos usuários. Mantenha registros de erro em um local seguro e acesse-os apenas quando necessário para diagnóstico.

VII. Cookie without SameSite Attribute
   A. Resumo: O cookie não possui o atributo SameSite configurado, o que pode resultar em riscos de segurança, especialmente em cenários de terceiros, como ataques de CSRF (Cross-Site Request Forgery).
   B. Solução: Adicionar o atributo SameSite aos cookies, especificando *SameSite=Strict* ou *SameSite=Lax* conforme necessário para controlar o comportamento do cookie em

relação às solicitações de terceiros. Isso ajuda a mitigar riscos de CSRF.

VIII. Cross-domain JavaScript source file inclusion
   A. Problema: Permite a inclusão de arquivos JavaScript de fontes externas não confiáveis, o que pode resultar em ataques Cross-Site Scripting (XSS).
   B. Solução: Utilizar Content Security Policy (CSP) para controlar quais fontes são permitidas para execução de scripts na página.

IX. Divulgação de informações
   A. Problema: A aplicação revela informações sensíveis, como detalhes do servidor, mensagens de erro detalhadas ou outros dados confidenciais.
   B. Solução: Configurar a aplicação para fornecer mensagens de erro genéricas e eviter expor informações sensíveis. Outra solução é utilizar códigos que somente administradores tenham o conhecimento do erro que cada um significa.

X. Referer exposes session ID
   A. O cabeçalho "Referer" está expondo o ID de sessão, o que pode representar um risco de segurança, especialmente se o tráfego estiver sendo observado por terceiros.
   B. Solução: Retirar informações sensíveis no cabeçalho

XI. X-content-type-options-header-missing
   A. Problema: O cabeçalho HTTP X-Content-Type-Options não está presente na resposta, o que pode permitir ataques de tipo MIME sniffing.
   B. Solução: Adicionar o cabeçalho X-Content-Type-Options: nosniff à resposta HTTP. Isso instrui os navegadores a não realizar o "sniffing" do tipo MIME, restringindo o conteúdo àquele definido pelo cabeçalho Content-Type.

XII. Cookies com escopo fraco
   A. Problema: Os cookies estão configurados com escopo mais amplo do que o necessário, o que pode aumentar o risco de ataques de Cross-Site Scripting (XSS) e Cross-Site Request Forgery (CSRF).

B. Solução: Configurar os cookies para terem um escopo restrito, especificando domínios, caminhos e atributos Secure e HttpOnly conforme necessário.

XIII. Divulgação de informações

A. Problema: A aplicação revela informações sensíveis, como detalhes do servidor, mensagens de erro detalhadas.

B. Solução: Configurar a aplicação para fornecer mensagens de erro genéricas.

XIV. Session management response identified

A. Problema: A resposta HTTP contém evidências de manipulação inadequada da sessão do usuário, o que pode levar a vulnerabilidades de gestão de sessão.

B. Solução: Certificar de que a gestão de sessão seja realizada de maneira segura. Utilizando mecanismos seguros para autenticação e gestão de sessão.

XV. User controllable HTML element attribute (potential XSS).

A. Problema: Elementos HTML permitem a entrada do usuário sem a devida validação ou escaping, podendo resultar em vulnerabilidades de Cross-Site Scripting (XSS).

B. Solução: Implementa validação e escaping adequados para entrada do usuário em elementos HTML. Utilize técnicas como context-aware encoding para garantir que os dados de entrada não sejam interpretados como código executável no contexto do navegador.

5. CONCLUSÃO

Nesse contexto, a presente monografia visa explorar a temática crítica da "Segurança em Aplicações Web", focando especificamente na análise de vulnerabilidades e testes de penetração utilizando o framework spring boot, este focado na linguagem de programação Java, e com a análise das falhas identificadas nesta pesquisa, o que foi revelado é uma perspectiva sobre a segurança de um sistema que poderia em cenários diversos, ter em sua posse, dados sensíveis de vários usuários, e evidenciaram-se falhas de natureza moderada e leve, como a ferramenta OWASP ZAP classificou, mas até mesmo podendo serem críticas, indicando áreas que requerem atenção para fortalecer a robustez do sistema. No entanto, é crucial ressaltar que uma parcela significativa de possíveis vulnerabilidades foi mitigada eficientemente devido às configurações implementadas pelas bibliotecas Java e Spring Boot. Essas escolhas de configuração mostraram-se fundamentais para a preservação da integridade e segurança da aplicação, atuando como um escudo eficaz contra potenciais ameaças.

Ao reconhecer as falhas identificadas como oportunidades para melhorias contínuas, os resultados desta pesquisa oferecem um direcionamento claro para aprimoramentos futuros na segurança do sistema. A implementação eficaz de medidas corretivas, aliada à manutenção das práticas seguras já incorporadas, proporcionará um ambiente mais resiliente e confiável. Este estudo ressalta a importância da constante vigilância e atualização das práticas de segurança, reforçando a necessidade de uma abordagem proativa na mitigação de potenciais riscos, garantindo assim a robustez e a confiabilidade contínuas do sistema em questão. O reconhecimento das configurações eficazes da biblioteca Java e do framework Spring Boot é crucial, evidenciando a relevância das escolhas tecnológicas na promoção de um ambiente mais seguro. Visto que os scanners de vulnerabilidades usados testaram as mais amplas possibilidades de ataques, e apenas 12 foram encontradas.

Isso não significa que deve-se despreocupar. A constante evolução das ameaças cibernéticas demanda uma postura proativa na implementação de medidas de segurança, garantindo que as práticas de desenvolvimento estejam alinhadas com as mais recentes diretrizes de segurança.

Ao analisar a aplicação em questão, percebemos que seus elementos são cruciais para o desempenho, demonstrando uma estrutura simples, porém fundamental. À medida que as aplicações evoluem, adquirindo novas funcionalidades e integrações, torna-se imperativo reconhecer que os riscos associados também se ampliam significativamente. Diante desse cenário, ressalta-se a importância crucial de conduzir estudos e testes frequentes no âmbito da segurança cibernética. Portanto, é fundamental estabelecer uma cultura organizacional que valorize a segurança cibernética como uma prioridade estratégica, garantindo a resiliência do sistema diante de desafios crescentes no panorama digital.

REFERÊNCIAS

Anderson, R. (2015). Security Engineering: A Guide to Building Dependable Distributed Systems (2nd ed.). Wiley

K. M. Henry. 2012. **Penetration Testing: Protecting Networks and Systems**. IT Governance Publishing.

SCHNEIER, Bruce. (2001) Segurança.Com - **Segredos e Mentiras Sobre a Proteção na Vida Digital. São Paulo**: Campus

Apache Software Foundation. (2024). **Apache HTTP Server 2.4 Vulnerabilities.** Disponível em: https://httpd.apache.org/security/vulnera bilities_24.html

Cavalcante, J. (2024). **Gerenciamento de Alunos** - CRUD Spring Boot. Disponível em: https://github.com/juliuscavalcante/gerenciamento-alu nos-crud-springboot

Computerworld. (2024). **Five free pen-testing tools.** Computerworld. Disponível em: https://www.computerworld.com/article/2536045/five-free-pen-testing-tools.html

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). **Introduction to Algorithms** (3rd ed.). MIT Press.

FIRMINO, M. (2019). **Segurança de redes: Introdução à Segurança da Informação**. IFRN. Disponível em: https://docente.ifrn.edu.br/josemacedo/disciplinas/2019/2019.1/introducao-aseguranca-da-informacao. Acesso em: 13 ago. 2023.

Deitel, P. J., & Deitel, H. M. (2018). **Java: How to Program** (11th ed.). Pearson

GeeksforGeeks. (2022). **Introduction to Laravel and MVC Framework.** GeeksforGeeks. Disponível em: https://www.geeksforgeeks.org/introduction-to-laravel-and-mvc-framework/

Mitnick, Kevin D. A. (2005). **Arte de Invadir: Como Proteger-se no Mundo Digital.** Rio de Janeiro: Alta Books.

Nystrom, M. (2018). **Go Programming Blueprints: Build real-world, production-ready solutions in Go using cutting-edge technology and techniques.** Packt Publishing.

Schneier, B. (2015). Data and Goliath: **The Hidden Battles to Collect Your Data and Control Your World**. W. W. Norton & Company.

Sedgewick, R., & Wayne, K. (2011). **Algorithms** (4th ed.) Addison-Wesley.

Stallings, W. (2017). **Cryptography and Network Security: Principles and Practice** (7th ed.). Pearson.

Unixwiz. (2024). **SQL Injection Attacks by Example.** Disponível em: http://www.unixwiz.net/techtips/sql-injection.html

Jornal do Campus. (2021, novembro). Apagão das redes: a queda do WhatsApp, Facebook e Instagram. Jornal do Campus. Disponível em: https://www.jornaldocampus.usp.br/index.php/2021/11/apagao-das-redes-a-queda-do-whatsapp-facebook-e-instagram/