
**UNIVERSIDADE FEDERAL DE CAMPINA GRANDE**
**CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA**
**CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

**LUCIANO ERICK SOUSA FIGUEIREDO FILHO**

# VULNVISOR: UM PLUGIN PARA GERAÇÃO DE DASHBOARDS DE VULNERABILIDADES PARA A FERRAMENTA TRIVY

**CAMPINA GRANDE - PB**

**2023**

**LUCIANO ERICK SOUSA FIGUEIREDO FILHO**

# VULNVISOR: UM PLUGIN PARA GERAÇÃO DE DASHBOARDS DE VULNERABILIDADES PARA A FERRAMENTA TRIVY

**Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.**

**Orientador : João Arthur Brunet Monteiro**

**CAMPINA GRANDE - PB**

**2023**

**LUCIANO ERICK SOUSA FIGUEIREDO FILHO**

# VULNVISOR: UM PLUGIN PARA GERAÇÃO DE DASHBOARDS DE VULNERABILIDADES PARA A FERRAMENTA TRIVY

**Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.**

## BANCA EXAMINADORA:

**João Arthur Brunet Monteiro**

**Orientador – UASC/CEEI/UFCG**

**Andrey Elisio Monteiro Brito**

**Examinador – UASC/CEEI/UFCG**

**Melina Mongiovi Cunha Lima Sabino**

**Professor da Disciplina TCC – UASC/CEEI/UFCG**

**Trabalho aprovado em: 17 de NOVEMBRO de 2023.**

**CAMPINA GRANDE - PB**

# RESUMO

O crescente uso de componentes de terceiros em projetos de software impulsionou a eficiência do desenvolvimento, mas também introduziu preocupações significativas relacionadas à segurança. A falta de visibilidade e rastreabilidade efetiva das dependências utilizadas pode resultar em vulnerabilidades desconhecidas, destacando a necessidade de gerenciar e avaliar esses componentes. Em resposta a estes desafios, o Trivy, uma ferramenta open source de escaneamento de vulnerabilidades, tornou-se fundamental na identificação e mitigação de ameaças em componentes de software. No entanto, um dos principais obstáculos enfrentados pelos usuários é a complexidade na interpretação dos dados gerados pela ferramenta. Desta forma, este trabalho se propõe a desenvolver um plugin para o Trivy, com o objeto de aprimorar a experiência dos usuários através da geração de um dashboard interativo. Além disso, este documento busca avaliar a ferramenta desenvolvida através de feedbacks de potenciais usuários, medindo o grau de satisfação e eficácia para mitigação de vulnerabilidades.

# VULNVISOR: A PLUGIN FOR VULNERABILITY DASHBOARD GENERATION FOR THE TRIVY TOOL

## ABSTRACT

The growing use of third-party components in software projects has driven development efficiency but has also introduced significant security concerns. The lack of visibility and effective traceability of dependencies used can result in unknown vulnerabilities, highlighting the need to manage and assess these components. In response to these challenges, Trivy, an open-source vulnerability scanning tool, has become crucial in identifying and mitigating threats in software components. However, one of the main obstacles faced by users is the complexity in interpreting the data generated by the tool. Therefore, this work aims to develop a plugin for Trivy to enhance user experience by generating an interactive dashboard. Additionally, this document seeks to evaluate the developed tool through feedback from potential users, measuring the degree of satisfaction and effectiveness in vulnerability mitigation.

# VulnVisor: Um plugin para geração de dashboards de vulnerabilidades para a ferramenta Trivy

Luciano Erick Sousa Figueiredo Filho
Universidade Federal de Campina Grande
Campina Grande, Paraíba
luciano.erick.filho@ccc.ufcg.edu.br

João Arthur Brunet Monteiro
Universidade Federal de Campina Grande
Campina Grande, Paraíba
joao.arthur@computacao.ufcg.edu.br

## RESUMO

O crescente uso de componentes de terceiros em projetos de software impulsionou a eficiência do desenvolvimento, mas também introduziu preocupações significativas relacionadas à segurança. A falta de visibilidade e rastreabilidade efetiva das dependências utilizadas pode resultar em vulnerabilidades desconhecidas, destacando a necessidade de gerenciar e avaliar esses componentes. Em resposta a estes desafios, o Trivy, uma ferramenta open source de escaneamento de vulnerabilidades, tornou-se fundamental na identificação e mitigação de ameaças em componentes de software. No entanto, um dos principais obstáculos enfrentados pelos usuários é a complexidade na interpretação dos dados gerados pela ferramenta. Desta forma, este trabalho se propõe a desenvolver um plugin para o Trivy, com o objeto de aprimorar a experiência dos usuários através da geração de um dashboard interativo. Além disso, este documento busca avaliar a ferramenta desenvolvida através de feedbacks de potenciais usuários, medindo o grau de satisfação e eficácia para mitigação de vulnerabilidades.

## REPOSITÓRIOS

https://github.com/LucianErick/VulnVisor
https://github.com/LucianErick/trivy-plugin

## 1. INTRODUÇÃO

O desenvolvimento de software moderno frequentemente se baseia na integração extensiva de componentes de terceiros, incluindo bibliotecas, frameworks e plugins, a fim de impulsionar a eficiência e a rapidez na entrega de aplicações inovadoras. Segundo um estudo recente, cerca de 96% das aplicações comerciais usam componentes de código aberto, demonstrando assim a importância crescente desses recursos no ecossistema de desenvolvimento de software [1].

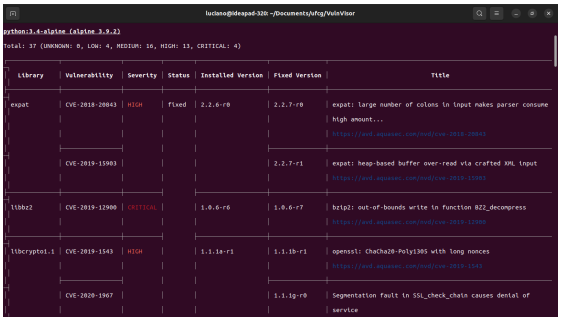
Essa crescente dependência, embora benéfica para o avanço ágil de projetos, também trouxe consigo desafios consideráveis relacionados à segurança. A vulnerabilidade dos componentes utilizados pode resultar em lacunas significativas na segurança das aplicações, tornando o gerenciamento e a rastreabilidade das dependências uma prioridade crucial.

Nesse contexto, ferramentas de escaneamento de vulnerabilidades, como o Trivy [2], têm desempenhado um papel fundamental na identificação e mitigação de ameaças em componentes de software. O Trivy é um scanner open source extremamente popular capaz de rastrear pacotes, detectar vulnerabilidades conhecidas, informações sensíveis e licenças de software para diferentes alvos de varredura, como repositórios git, imagens de containers, sistemas de arquivos, AWS, etc.

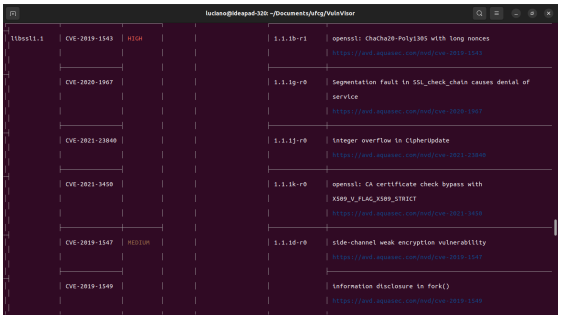
Porém, um dos desafios substanciais enfrentados pelos usuários reside na interpretação dos resultados obtidos na ferramenta, através do prompt de comando. Estes resultados, exibidos em formato de tabela, embora informativos, frequentemente carecem de uma representação visual clara e de uma experiência de usuário adequada, especialmente quando apresentam um grande volume de dados, impactando diretamente na habilidade do usuário em tomar ações assertivas e imediatas para mitigar os riscos associados.

Um exemplo disso é mostrado na figura 1. Nela, vemos que o resultado do scanner consegue trazer informações valiosas e que, certamente, são de interesse do usuário. Porém, conforme a quantidade de informações aumenta, mostrado na figura 2, é necessário que o usuário fique navegando para cima e para baixo dentro da tabela para recuperar dados importantes, afetando sua experiência.

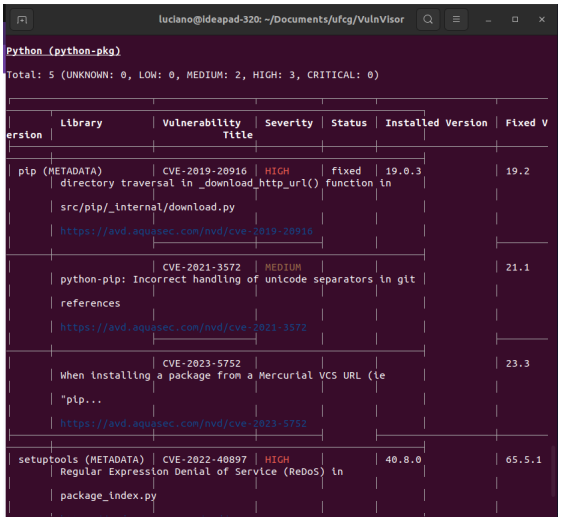
Além disso, para realizar filtragens é necessário o uso de diversas flags, tornando-se complexo para o usuário fazer isso repetidamente. Por fim, outro problema encontrado é a má responsividade quando visualizado em telas menores, mostrado na figura 3, nas quais as informações de algumas colunas são sobrepostas a outras, prejudicando a visualização.




**[Figura 1: Resultado do scanner Trivy]**




**[Figura 2: Resultado do scanner Trivy com muitos dados]**




**[Figura 3: Resultado do scanner Trivy em uma tela menor]**

Diante disso, foi desenvolvido um plugin [14] para a extração das informações do relatório do Trivy e a geração de um dashboard intuitivo e de fácil compreensão, fornecendo insights valiosos para ajudar na identificação e posterior mitigação dos riscos de segurança identificados.

Este trabalho está subdividido em seis seções distintas. A primeira seção consiste em uma introdução que contextualiza a necessidade do sistema. A segunda seção discute as funcionalidades essenciais e as decisões arquiteturais tomadas. A terceira seção aborda o processo de desenvolvimento do sistema e os desafios encontrados durante esse processo. A quarta seção trata da avaliação do sistema por meio de feedbacks de usuários. A quinta seção apresenta uma conclusão sobre o trabalho feito. E, por último, a sexta seção oferece sugestões para possíveis direções que a aplicação pode seguir.

# 2. SOLUÇÃO

A solução proposta é o plugin Vulnvisor, que tem o objetivo de criar um dashboard amigável e interativo onde qualquer usuário pode solicitar o relatório de imagens de containers ou do sistema de arquivos da sua máquina e, automaticamente, é criado um arquivo em formato html que pode ser acessado via browser.

## 2.1 Funcionalidades

Implementando algumas sugestões da comunidade [3], o plugin Vulnvisor utiliza das informações extraídas do Trivy para gerar insights valiosos.

### 2.1.1 Gráfico de vulnerabilidades por severidade

O usuário consegue visualizar a quantidade de vulnerabilidades encontradas de acordo com a severidade (Baixa, Média, Alta e Crítica) em cada uma das fontes de dependências presentes no software escaneado.

### 2.1.2 Gráfico de vulnerabilidades por pacote

O usuário consegue visualizar os seis pacotes com mais vulnerabilidades associadas e suas respectivas quantidades encontradas.

### 2.1.3 Tabela de mitigações

O usuário consegue visualizar as vulnerabilidades que já foram resolvidas na versão instalada, aquelas que permanecem sem resolução e o número daquelas para as quais existem versões de correção disponíveis.

### 2.1.4 Tabelas de vulnerabilidades e pacotes

O usuário consegue visualizar, filtrar e pesquisar por palavras-chave dentro da tabela quaisquer dados que ele queira recuperar. Além disso, ele consegue exportar os dados da tabela específica para um arquivo no formato csv.

### 2.1.5 Informações sensíveis

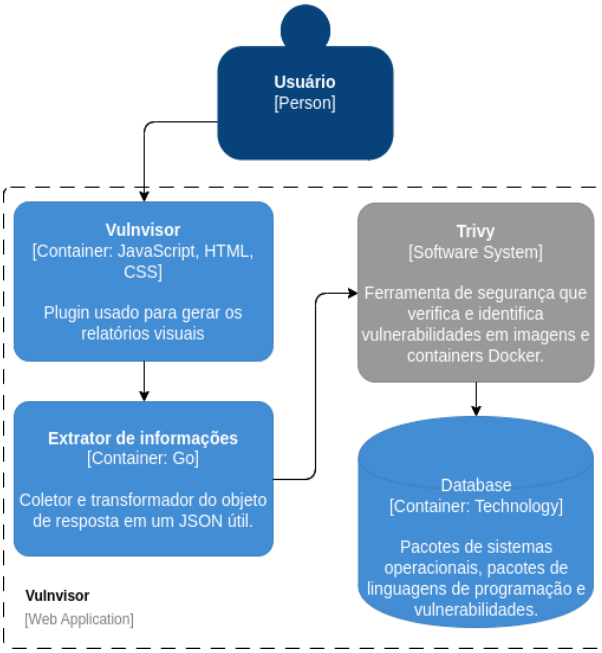
O usuário consegue visualizar informações sensíveis encontradas dentro do código escaneado (Por exemplo chaves de API, arquivos .pem, etc.)

### 2.1.6 Árvore de dependências reversa

O usuário consegue visualizar e expandir a árvore de dependências dos pacotes nos quais foram encontradas vulnerabilidades.

## 2.2 Arquitetura

Nesta seção, é fornecida uma visão geral sobre a arquitetura da solução, apresentando não apenas a estrutura de cada módulo desenvolvido, mas também as decisões críticas tomadas durante o processo de desenvolvimento.




**[Figura 4: Diagrama arquitetural do Vulnvisor]**

### 2.2.1 Extrator

Seguindo a arquitetura da aplicação, o coletor de informações (Extrator de informações na figura 4), desempenha um papel essencial para o funcionamento do plugin. Esse componente tem dois propósitos fundamentais: coletar o JSON [9] de resposta proveniente da ferramenta Trivy e configurar os workflows no Github [12]. Essa configuração é crucial para a verificação de versões e o gerenciamento de novas releases.

A implementação desse módulo foi realizada na linguagem de programação Go [4], escolha motivada pela predominância dessa mesma linguagem na ferramenta principal, o Trivy. O Go oferece diversos benefícios significativos para o projeto, principalmente uma documentação abrangente que auxilia no aprendizado e na compreensão eficaz da linguagem. Sua curva de aprendizado amigável e sintaxe clara proporcionam facilidade para os desenvolvedores compreenderem e trabalharem com eficiência na linguagem.

Sua implementação consiste em métodos que executam e interagem com a aplicação principal, seguindo argumentos específicos, métodos que processam e transformam os relatórios em objetos utilizáveis e métodos auxiliares para operações complementares. Essa implementação permite a comunicação eficaz entre o plugin e o Trivy, possibilitando o processamento e a extração de informações vitais para a funcionalidade do sistema.


**[Figura 5: Objeto JSON extraído]**

### 2.2.2 Frontend

O frontend (Vulnvisor na figura 4) foi desenvolvido usando HTML, CSS e JavaScript, sem a utilização de frameworks. Essa escolha foi impulsionada pela busca por um desempenho otimizado e uma experiência ágil para os usuários. A abordagem com essas linguagens essenciais proporciona uma execução mais leve e eficiente do frontend. Ao dispensar a dependência de frameworks, o código tende a ser mais simplificado, sem a necessidade de instalar dependências externas, resultando em menor sobrecarga. Isso se reflete em um carregamento mais rápido da página e em uma interação mais ágil para o usuário final.

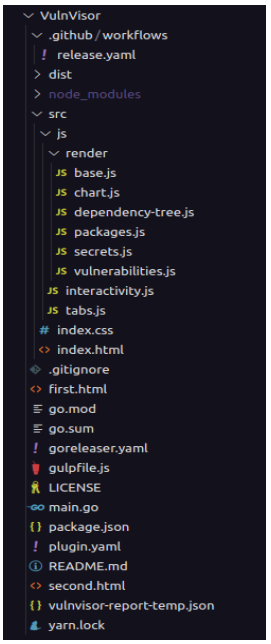
Porém, por seguir esse caminho surge um problema: o da renderização dinâmica. Essencialmente, apenas essas

tecnologias puras não conseguem lidar com atualizações da estrutura da página conforme interação com o usuário nem mudanças em tempo real, então por esse motivo optou-se pela utilização do *Gulp* [7].

O Gulp é uma ferramenta de automação de tarefas comumente utilizada no desenvolvimento web devido à sua capacidade de simplificar e automatizar processos repetitivos. Sua escolha se baseia na necessidade de executar tarefas de compilação, minificação de arquivos, otimização de imagens, entre outras atividades, de maneira automatizada e eficiente [11]. Sua utilização no contexto do Vulnvisor possibilita a segmentação do HTML em duas partes distintas: estática e dinâmica, oferecendo uma abordagem modularizada e organizada para a construção do frontend. A seção estática define a estrutura base da página, enquanto a dinâmica permite a manipulação dos dados e a interação com o usuário, facilitando o controle sobre a interface e a otimização da renderização dos dados.

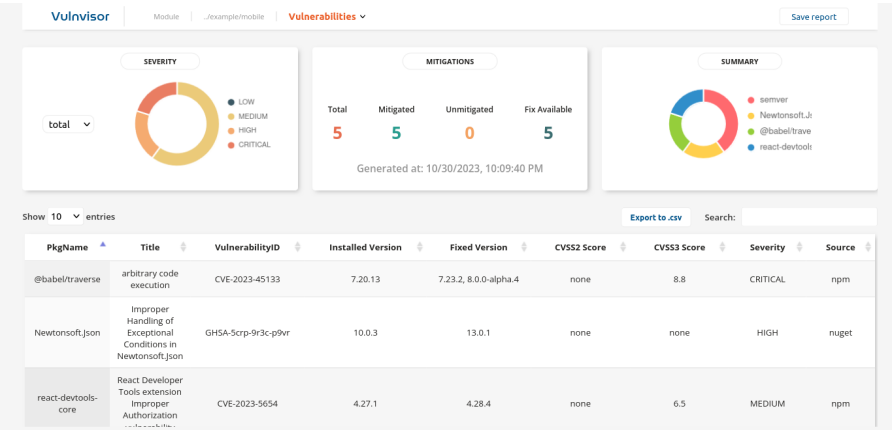
Para a construção dos gráficos renderizados dinamicamente, foi utilizada a biblioteca *Chart.js* [5]. Ela é uma biblioteca bastante popular e amplamente utilizada para criação de diversos tipos de gráficos e visualizações de dados dinâmicos e interativos em aplicações web. A facilidade de implementação, grande versatilidade, ampla gama de opções para personalização e a documentação abrangente foram fatores cruciais para a escolha dessa ferramenta.

Para construção de tabelas foi utilizada a biblioteca *DataTables* [6]. Ela é uma poderosa biblioteca JavaScript que se destaca por sua capacidade de aprimorar a renderização de tabelas HTML, oferecendo uma variedade de recursos e funcionalidades para tornar a apresentação de dados tabulares mais rica e interativa. Seu uso na aplicação proporciona diversos benefícios e funcionalidades avançadas, incluindo paginação, ordenação, filtro e pesquisa, personalização, responsividade e integração com diversos plugins.
Seu uso conseguiu enriquecer a experiência do usuário ao permitir a exploração e interação com os dados de forma mais dinâmica e funcional, tornando as tabelas uma parte vital da apresentação das informações sobre vulnerabilidades.

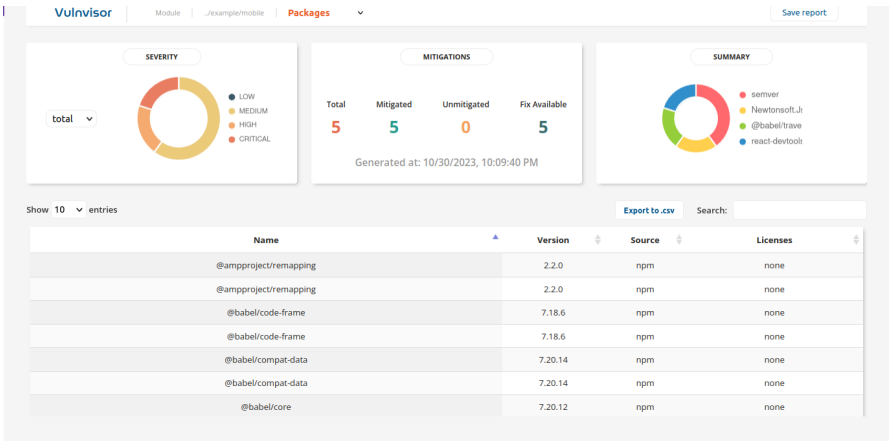



**[Figura 6: Estrutura do código do Vulnvisor]**

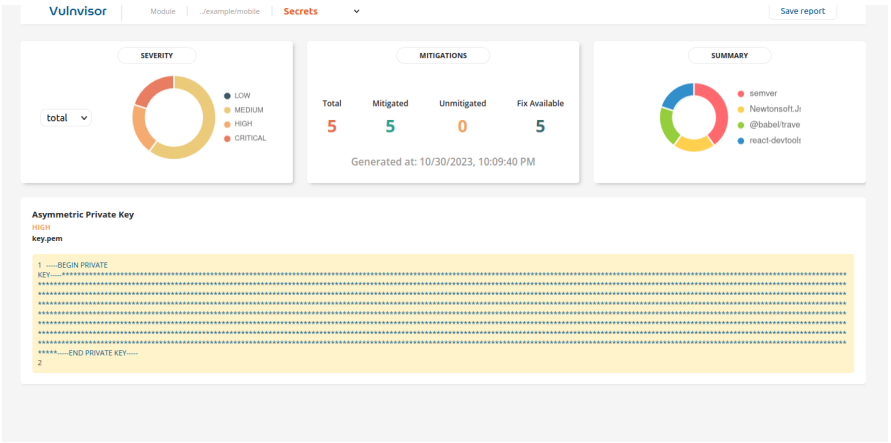
## 2.4 Sistema



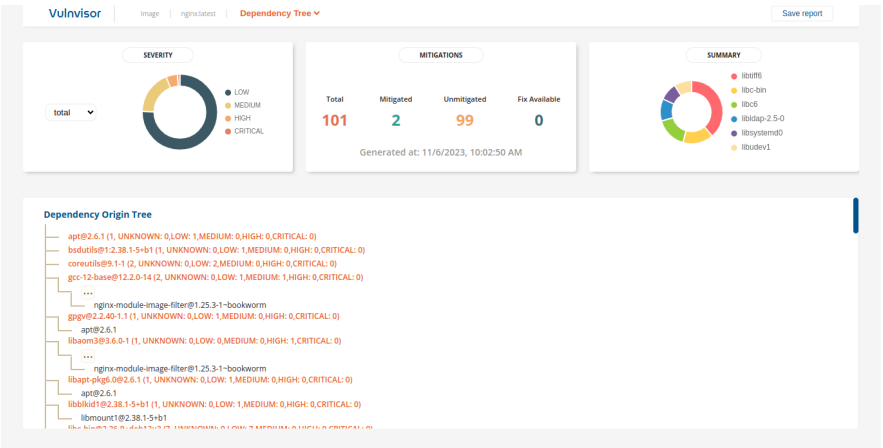

**[Figura 7: Aplicação com a aba de vulnerabilidades ativada]**




**[Figura 8: Aplicação com a aba de pacotes ativada]**




**[Figura 9: Aplicação com a aba de informações sensíveis ativada]**




**[Figura 10: Aplicação com a aba de árvore de dependências ativada]**

# 3. PROCESSO DE DESENVOLVIMENTO

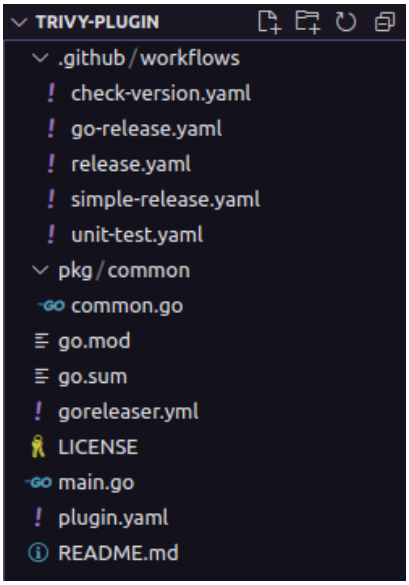
## 3.1 Investigação do código fonte

Em um primeiro momento, foi feita uma análise aprofundada no código fonte do Trivy buscando informações sobre como funciona a mecânica de execução da ferramenta, quais argumentos ela consegue processar, tipos de saída e como seus módulos se comunicam entre si. Em especial, foi analisado o pacote k8s (Kubernetes), além de seus tipos, interfaces e estruturas. Nele, é detalhado como se cria a estrutura base do relatório, um ponto crucial para a etapa seguinte.

## 3.2 Extração das informações

Após uma minuciosa análise na primeira etapa e entendendo a estrutura básica gerada nos relatórios do

Trivy, apresentados nas figuras 1 e 2, foi criado um repositório com o intuito de coletar e transformar o objeto de saída. Esse repositório é essencial para a execução da aplicação, tendo em vista que é chamado como dependência principal do plugin Vulnvisor. O código desse repositório, onde a estrutura é mostrada na figura 11, foi desenvolvido fundamentalmente na linguagem Go, tendo em vista que a maioria do código da ferramenta principal, Trivy, também é escrita nessa mesma linguagem.
A implementação se baseia na execução da ferramenta principal, seguindo parâmetros fixos já definidos no próprio código, e a posterior obtenção e transformação do objeto de resposta, dado em formato JSON (Javascript Object Notation).




**[Figura 11: Estrutura do extrator de informações]**

## 3.3 Definição de requisitos

Com base no conjunto de dados resultante da fase anterior, combinado com uma análise detalhada das sugestões provenientes da comunidade Trivy, foi elaborada uma documentação que descreve os comportamentos e funcionalidades que o dashboard deverá incorporar. Esse processo foi conduzido com o intuito de estabelecer os requisitos essenciais, os quais servirão como a base para captar e comunicar as necessidades, priorizar e alocar tarefas, gerenciar as expectativas, bem como validar o que será possível realizar com base no objeto JSON gerado pelo scanner, além de fazer o gerenciamento do escopo.
A tabela 1 detalha os requisitos identificados:

| Requisito | Descrição |
|---|---|
| R01 | Mostrar a contagem de vulnerabilidades por severidade (Baixa, Média, Alta e Crítica) em cada fonte de dependência. |
| R02 | Apresentar os seis pacotes com mais vulnerabilidades e suas respectivas quantidades. |
| R03 | Exibir vulnerabilidades resolvidas, sem resolução e as com versões de correção disponíveis. |
| R04 | Permitir listar, filtrar, pesquisar e exportar os dados das vulnerabilidades conhecidas. |
| R05 | Permitir listar, filtrar, pesquisar e exportar os dados dos pacotes instalados. |
| R06 | Listar os dados sensíveis encontrados no scanner. |
| R07 | Visualizar e expandir a árvore de dependências. |

**[Tabela 1: Requisitos do Vulnvisor]**

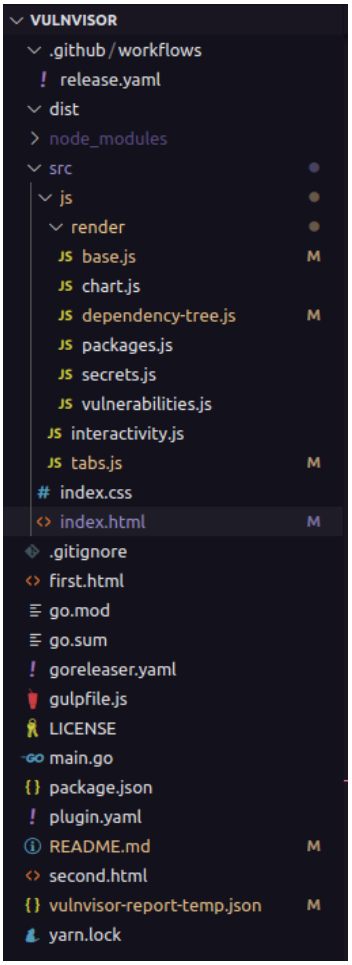
## 3.4 Prototipação da tela

A partir dos requisitos levantados na etapa anterior, foi necessário criar uma tela base utilizando o *Figma* [8], uma ferramenta online que permite o design de interfaces de usuário e a criação de protótipos interativos. Através dele, foram desenvolvidos diversos modelos de layout e design para explorar e testar diferentes formas de apresentação das informações sobre vulnerabilidades obtidas. Esses protótipos possibilitaram a análise minuciosa de representações visuais mais eficazes e de fácil compreensão, contribuindo para um resultado final mais intuitivo e funcional.

## 3.5 Implementação do frontend

Com base no design consolidado na etapa anterior, foi elaborado o frontend do dashboard, integrando as informações extraídas do Trivy, visto na segunda etapa. Este frontend, no qual sua estrutura é mostrada na figura 12, foi desenvolvido utilizando tecnologias fundamentais como o HTML, CSS e JavaScript em sua forma pura para otimizar o desempenho e garantir uma experiência mais leve para o usuário. No entanto, para habilitar a modificação dinâmica dos dados sem a dependência de nenhum framework específico, foi empregado o Gulp, uma ferramenta de automação amplamente utilizada no contexto do desenvolvimento web.
O Gulp desempenha um papel crucial na segmentação do HTML base em duas partes distintas: uma estática, onde é definida a estrutura da página, e outra dinâmica onde os dados são carregados e mostrados conforme interação com o usuário. Esta separação permite um melhor controle sobre a interface, facilitando a integração com os dados JSON previamente adquiridos e otimizando a

renderização dos dados, garantindo um sistema responsivo e ágil.




**[Figura 12 - Estrutura do frontend]**

## 3.6 Build da aplicação

Após concluir a implementação do código frontend, a etapa seguinte envolveu a configuração de um arquivo para a automatização de releases, em um processo gerenciado pelo próprio Github através de workflows. Este arquivo aciona outros de natureza similar dentro do repositório criado na segunda etapa, supervisionando o lançamento de novas versões e a geração de artefatos essenciais. Esses artefatos, como arquivos com extensão .tar.gz, são utilizados na instalação local do plugin.

A figura 13 mostra os artefatos gerados no build da aplicação. A presença do arquivo plugin.yaml é o que permite fazer a instalação localmente, enquanto os arquivos first.html e second.html são os templates para a geração do relatório.

```yaml
project_name: vulnvisor
builds:
  -
    main: main.go
    binary: vulnvisor
    ldflags:
      - -s -w
      - "-extldflags '-static'"
      - -X main.version={{.Version}}
    goos:
      - darwin
      - linux
      - freebsd
      - windows
    goarch:
      - amd64
      - 386
      - arm
      - arm64
      - s390x
      - ppc64le
    goarm:
      - 7

archives:
  -
    format: tar.gz
    name_template: "{{.Version}}_{{.Os}}-{{.Arch}}"
    files:
      - README.md
      - LICENSE
      - plugin.yaml
      - first.html
      - second.html
    format_overrides:
      - goos: windows
        format: zip

```

**[Figura 13 - arquivo goreleaser.yaml]**

# 4. AVALIAÇÃO

Nesta seção, foram analisados diversos aspectos cruciais relacionados ao desempenho e à receptividade do plugin Vulnvisor entre potenciais usuários. Avaliou-se a facilidade de uso, enfatizando a importância de uma interface intuitiva e navegável. Além disso, foi verificado o impacto do plugin na ferramenta principal, buscando compreender a utilidades dos relatórios visuais e sua influência na experiência do usuário. Também foi considerada a visualização dos dados, analisando a utilidade das informações apresentadas. Por fim, também foi examinada a taxa de retenção de usuários, indicando a probabilidade de reutilização do plugin no futuro, essencial para avaliar a qualidade do software e sua capacidade de atender às necessidades dos usuários.

## 4.1 Metodologia

Foi disponibilizado um formulário, disponível no período de 15 de outubro de 2023 a 22 de outubro de 2023, para 12 possíveis usuários com o objetivo de permitir que eles experimentassem o plugin na versão estável v0.1.1 e compartilhassem feedback sobre sua funcionalidade e usabilidade. Esses usuários foram cuidadosamente selecionados da comunidade de usuários do Trivy, sendo abordados de forma privada devido ao seu interesse no tema.

O questionário tinha um total de seis perguntas, com todas, exceto a última, tendo quatro respostas possíveis: poor, average, good e excellent. Além disso, todas as perguntas e respostas foram escritas em inglês, de modo a abranger um maior público. Para a elaboração das perguntas, elas foram divididas em quatro categorias distintas.

A primeira categoria visa classificar o grau de facilidade de uso da aplicação, impactando diretamente na experiência do usuário. Para isso, foram elaboradas duas perguntas diretas: uma sobre o quão fácil é usar o plugin e outra sobre como foi a experiência do usuário ao utilizá-lo.

A segunda categoria visa estimar o impacto do plugin na ferramenta principal, dadas suas limitações mostradas nas figuras 2 e 3. Para isso, foi necessária apenas uma única pergunta, sobre como o usuário avalia esse impacto.

A terceira categoria avalia a qualidade e utilidade das visualizações de dados escolhidas para representar as informações necessárias. Para tal objetivo, foram elaboradas duas perguntas: uma sobre como o usuário avalia os gráficos que contém as estatísticas relacionadas ao relatório e outra sobre como ele avalia as tabelas geradas.

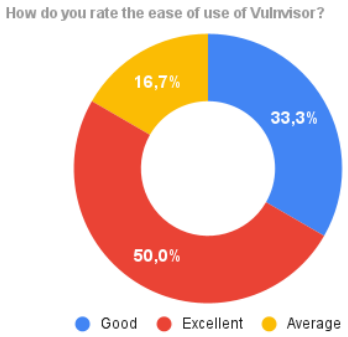
A quarta e última categoria tenta obter informações sobre a taxa de retenção de usuários, isto é, quantos deles voltariam a utilizar a ferramenta futuramente. Para isso, foi necessária apenas uma pergunta direta com três respostas possíveis: sim, não e não sei.

## 4.2 Resultados

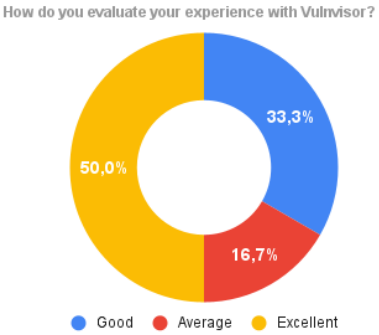
As avaliações, no geral, foram altamente positivas. Elas forneceram respostas encorajadoras destacando os pontos fortes do plugin. Os resultados abaixo seguirão as categorias descritas na subseção acima.

### 4.2.1 Facilidade de uso

Na avaliação mostrada nas figura 14 e 15, nitidamente, mostra que os usuários acham que a interface é fácil de ser navegada e encontrar as informações desejadas. Isso ressalta a importância de uma interface intuitiva, tornando a experiência mais acessível e eficaz.



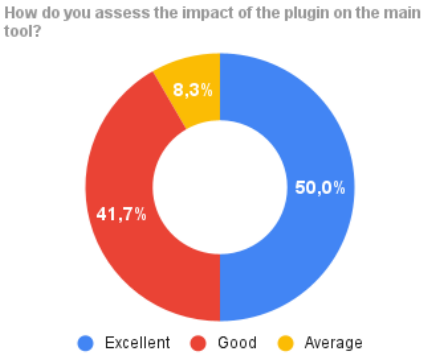

**[Figura 14: Gráfico de rosca sobre a avaliação intuitividade do Vulnvisor]**




**[Figura 15: Gráfico de rosca sobre a avaliação da experiência com o Vulnvisor]**

### 4.2.2 Impacto

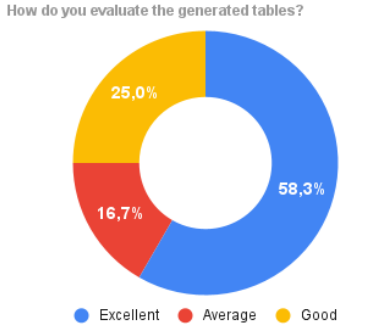
Com base na avaliação exibida na figura 16, a maioria dos usuários considerou positivo o impacto do plugin na ferramenta principal, evidenciando a utilidade dos relatórios visuais gerados pelo Vulnvisor para uma melhor experiência durante a utilização do Trivy.



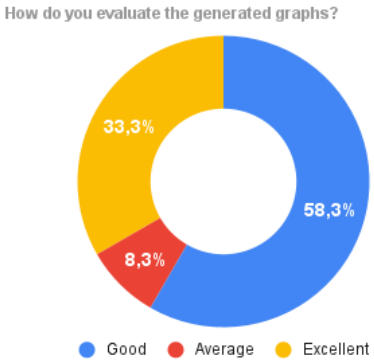

**[Figura 16: Gráfico de rosca sobre a avaliação do impacto na ferramenta principal]**

### 4.2.3 Visualização dos dados

Ainda analisando as avaliações das figuras 17 e 18, ficou evidente que as escolhas foram corretas para a visualização dos dados na aplicação. A disposição e a forma como as informações são apresentadas foram consideradas eficazes, facilitando a interpretação e compreensão dos dados pelos usuários. Isso destaca a relevância da apresentação visual para transmitir informações complexas de maneira clara e acessível, proporcionando uma experiência agradável e proveitosa.



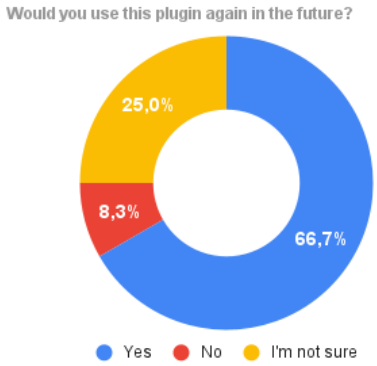
**[Figura 17: Gráfico de rosca sobre a avaliação dos gráficos gerados]**




**[Figura 18: Gráfico de rosca sobre a avaliação das tabelas geradas]**

4.2.4 Taxa de retenção de usuários

Na avaliação exibida na figura 19 mostra que a taxa de retenção provavelmente será alta. Isso refere-se à probabilidade com que os usuários planejam ou indicam o uso futuro do Vulnvisor para visualizar seus relatórios. Essa métrica é essencial para entender a satisfação dos usuários com a ferramenta e sua intenção de continuar usando, sendo um indicador significativo da qualidade do software e da capacidade de atender às necessidades dos mesmos.




**[Figura 19: Gráfico de rosca sobre a probabilidade de reuso por parte do usuário]**

# 5. CONCLUSÃO

As avaliações extremamente positivas na seção anterior ressaltaram o Vulnvisor como uma ferramenta promissora para melhorar significativamente a experiência dos usuários com a ferramenta principal. De modo geral, a diversidade de visualizações de dados foi altamente apreciada pelos usuários, destacando a utilidade e eficácia das informações apresentadas. Além disso, a facilidade de uso e a interface intuitiva do plugin foram pontos fortemente elogiados, evidenciando a relevância dos esforços para aprimorar a experiência do usuário, um dos objetivos-chave deste trabalho.

# 6. TRABALHOS FUTUROS

Embora o Vulnvisor tenha aprimorado a experiência dos usuários na visualização dos relatórios gerados pelo Trivy, ainda há várias funcionalidades que poderiam ser incorporadas no futuro para fortalecer ainda mais a solução mostrada.

## 6.1 Relatórios de misconfigurations

No contexto do Trivy, as misconfigurations [13] referem-se a configurações inadequadas ou incorretas em aplicações, sistemas ou serviços que podem levar a vulnerabilidades de segurança. Isso permite uma visão mais ampla e abrangente das possíveis brechas que podem ser exploradas por invasores. Por esse motivo, adicionar esse tipo de informação ao relatório poderia ser de grande ajuda ao usuário.

## 6.3 Mitigação de vulnerabilidades

Um recurso adicional de valor seria oferecer diretrizes, soluções ou recomendações para lidar com as vulnerabilidades identificadas. Isso envolveria apresentar abordagens para corrigir ou minimizar as falhas de segurança. A oferta dessas orientações para medidas corretivas possíveis pode ser extremamente benéfica para os usuários.

## 6.4 Integração com ferramentas de gerenciamento de projetos

Uma outra etapa valiosa para o aprimoramento do Vulnvisor seria a integração com ferramentas de gerenciamento de projetos amplamente utilizadas, como Trello, Jira, entre outras. Essa integração permitiria que os usuários sincronizassem, rastreassem e gerenciassem tarefas e correções identificadas nos relatórios de segurança, diretamente nas plataformas de gerenciamento de projetos. Isso resultaria em uma abordagem mais fluida e integrada para lidar com as questões de segurança detectadas, agilizando o processo de resolução e aprimorando a eficiência no desenvolvimento de software.

# 7. AGRADECIMENTOS

Eu gostaria primeiramente de agradecer a Deus pela oportunidade de chegar até aqui, sem ele nada disso seria possível. Também gostaria de agradecer a meus pais, Luciano e Hoanna, por trabalharem incessantemente desde sempre para me darem essa oportunidade que pouquíssimas pessoas tem. Gostaria de agradecer também a meus avós, Antônio e Gildete, por me criarem desde cedo, me darem amor e me ensinarem os valores que carrego até hoje. Agradeço também a Ana Carolina, minha companheira que conheci no início da graduação e durante todo esse tempo me deu apoio, amor e foi

essencial nesse processo, e aos inúmeros amigos que conheci durante esse tempo. Também queria agradecer ao meu orientador, João Arthur Brunet, por me dar a ideia inicial de tema e, principalmente, pela oportunidade na qual serei eternamente grato de entrar em meu primeiro projeto na UFCG. Por último, agradeço ao curso de Ciência da Computação da UFCG por propiciar um ambiente de contínuo crescimento e aprendizado.

# 8. REFERÊNCIAS BIBLIOGRÁFICAS

[1] https://www.synopsys.com/software-integrity/resources/analyst-reports/open-source-security-risk-analysis.html. Acesso em: 29 de out. de 2023.
[2] Overview - Trivy. Disponível em: <https://aquasecurity.github.io/trivy/v0.45/>. Acesso em 08 nov. 2023.
[3] aquasecurity/trivy - Discussions. Disponível em: <https://github.com/aquasecurity/trivy/discussions>. Acesso em: 8 nov. 2023.
[4] Documentation - The Go Programming Language. Disponível em: <https://go.dev/doc>. Acesso em: 8 nov. 2023.
[5] Getting Started | Chart.js. Disponível em: <https://www.chartjs.org/docs/latest/getting-started/>. Acesso em: 8 nov. 2023.
[6] Manual. Disponível em: <https://datatables.net/manual/index>. Acesso em: 8 nov. 2023.
[7] JavaScript and Gulpfiles | gulp.js. Disponível em: <https://gulpjs.com/docs/en/getting-started/javascript-and-gulpfiles>. Acesso em: 8 nov. 2023.
[8] FIGMA. Figma: the collaborative interface design tool. Disponível em: <https://www.figma.com/>. Acesso em 8 nov. 2023.
[9] JSON.ORG. JSON. Disponível em: <https://www.json.org/json-en.html>. Acesso em: 8 nov. 2023.
[10] INTERACTION DESIGN FOUNDATION. What is User Experience (UX) Design? Disponível em: <https://www.interaction-design.org/literature/topics/ux-design>. Acesso em: 8 nov. 2023.
[11] FERNANDA. Gulp direto ao ponto: fluxo de trabalho para iniciantes. Disponível em: <https://medium.com/@fnandaleite/gulp-direto-ao-ponto-fluxo-de-trabalho-para-iniciantes-2da02f5ab41e>. Acesso em: 8 nov. 2023.
[12] Triggering a workflow. Disponível em: <https://docs.github.com/en/actions/using-workflows/triggering-a-workflow>. Acesso em: 8 nov. 2023.
[13] Scanning - Trivy. Disponível em: <https://aquasecurity.github.io/trivy/v0.36/docs/misconfiguration/scanning/>. Acesso em: 8 nov. 2023.
[14] Plugins - Trivy. Disponível em: <https://aquasecurity.github.io/trivy/v0.36/docs/advanced/plugins/>. Acesso em: 8 nov. 2023.