INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
CAMPUS CAMPINA GRANDE
CURSO DE BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

CADEIA DE SUPRIMENTOS EM INFRAESTRUTURA DE AMBIENTES VIRTUALIZADOS COM KUBERNETES

CLARISSA DE LUCENA NÓBREGA
MARCELO RIBEIRO DA SILVA

ORIENTADOR: PROF. DR. ANDERSON FABIANO BATISTA F. DA COSTA
COORIENTADOR: PROF. DR. REINALDO CÉZAR DE MORAIS GOMES

CAMPINA GRANDE – PB
AGOSTO DE 2025

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA

CAMPUS CAMPINA GRANDE

CURSO DE BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

Cadeia de Suprimentos em Infraestrutura de Ambientes Virtualizados com Kubernetes

Clarissa De Lucena Nóbrega
Marcelo Ribeiro Da Silva

Orientador: Prof. Dr. Anderson Fabiano Batista F. Da Costa
Coorientador: Prof. Dr. Reinaldo Cézar De Morais Gomes

Artigo apresentado à Coordenação do Curso de Bacharelado em Engenharia de Computação do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, como requisito parcial para a obtenção do título de Engenheiro de Computação.

Campina Grande – PB
Agosto de 2025

Catalogação na fonte:
Ficha catalográfica elaborada por Gustavo César Nogueira da Costa - CRB 15/479

N737c Nóbrega, Clarissa de Lucena.

Cadeia de suprimentos em infraestrutura de ambientes virtualizados com Kubernetes / Clarissa de Lucena Nóbrega, Marcelo Ribeiro da Silva. - Campina Grande, 2025.
21 f.: il.

Trabalho de Conclusão de Curso (Graduação em Bacharelado em Engenharia de Computação) - Instituto Federal da Paraíba, 2025.
Orientador: Prof. Dr. Anderson Fabiano Batista F. da Costa.
Coorientador: Prof. Dr. Reinaldo Cézar de Morais Gomes.

1. Kubernetes. 2. Segurança da informação. 3. Infraestrutura virtualizada. 4. Cadeia de suprimentos digital. I. Silva, Marcelo Ribeiro da. II. Costa, Anderson Fabiano Batista F. da. III. Gomes, Reinaldo Cézar de Morais. IV. Título.

.

CDU 004.42

À memória de Angélica de Lucena Nóbrega, que sonhou conosco e segue presente em cada uma de nossas conquistas.

Agradecimentos

Agradecemos primeiramente a Deus, por nos conceder força, coragem e perseverança, permitindo que pudéssemos concluir este curso, mesmo diante das dificuldades.

À nossa família, pelo amor incondicional, confiança, apoio e compreensão ao longo de toda a trajetória acadêmica, que foram fundamentais para que superássemos os desafios e alcançássemos esta conquista.

Aos nossos orientadores, Prof. Dr. Anderson Fabiano Batista F. da Costa e Prof. Dr. Reinaldo Cézar de Morais Gomes, pela constante motivação, valiosas contribuições durante o desenvolvimento deste trabalho, e por todo o suporte, incentivo e pelas oportunidades que nos proporcionaram no decorrer do curso.

Aos demais professores do curso de Bacharelado em Engenharia de Computação e aos funcionários do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, em especial Dona Ângela, que direta ou indiretamente contribuíram para nossa formação, proporcionando conhecimento, apoio e ambiente propício ao aprendizado.

Aos nossos amigos Antonio Roberto, Aryelson Gonçalves, Felipe Braz e Ismael Marinho, pois, sem eles, esta caminhada não teria sido tão prazerosa.

Enfim, àqueles que, embora não citados, contribuíram de forma significativa nesta trajetória.

Cadeia de Suprimentos em Infraestrutura de Ambientes Virtualizados com Kubernetes

Clarissa de L. Nóbrega[1], Marcelo R. da Silva[1], Anderson Fabiano B. F. da Costa[1], Reinaldo Cézar de M. Gomes[2]

[1]Instituto Federal da Paraíba (IFPB) – Campina Grande, PB – Brazil

[2]Universidade Federal de Campina Grande (UFCG) – Campina Grande, PB – Brazil

{clarissa.nobrega,ribeiro.silva}@academico.ifpb.edu.br,
anderson@ifpb.edu.br, reinaldo@lsd.ufcg.edu.br

Abstract. *This work proposes the IBOM (Infrastructure Bill of Materials), a unified document that consolidates the concepts of SBOM (Software Bill of Materials) and KBOM (Kubernetes Bill of Materials) to facilitate vulnerability validation and analysis in Kubernetes environments. The approach centralizes the generation, unification, signing, and verification of infrastructure components using the tools Trivy, sbomasm, and Cosign. Experimental results demonstrate that the IBOM simplifies audits, reduces redundancy, and improves visibility across multiple systems. The consolidation of SBOMs and KBOMs into a single standardized document significantly enhances governance and security in modern infrastructure environments, eliminating the need for repetitive processes for each individual component.*

Keywords: *Software Bill of Materials, SBOM, Kubernetes Bill of Materials, KBOM, Kubernetes.*

Resumo. *Este trabalho propõe o IBOM (Lista de Materiais de Infraestrutura), um documento unificado que consolida os conceitos de SBOM (Lista de Materiais de Software) e KBOM (Lista de Materiais do Kubernetes) para facilitar a validação e análise de vulnerabilidades em ambientes Kubernetes. A abordagem centraliza a geração, unificação, assinatura e verificação de componentes da infraestrutura com o uso das ferramentas Trivy, sbomasm (SBOM Assembler) e Cosign. Os resultados experimentais demonstram que o IBOM simplifica auditorias, reduz redundâncias e melhora a visibilidade sobre múltiplos sistemas. A consolidação de SBOMs e KBOMs em um único documento padronizado amplia significativamente a governança e a segurança em ambientes de infraestrutura modernos, evitando a repetição de processos individuais para cada componente analisado.*

Palavras-chave: *Lista de Materiais de Software, SBOM, Lista de Materiais do Kubernetes, KBOM, Kubernetes.*

1. Introdução

A crescente complexidade dos ambientes de infraestrutura em nuvem e a necessidade de garantir a segurança em todas as camadas da cadeia de suprimento de software impulsionaram o desenvolvimento de mecanismos mais eficazes para inventariar e analisar componentes críticos. Nesse contexto, surgem dois conceitos fundamentais: o SBOM (*Software Bill of Materials,* Lista de Materiais de Software), que descreve detalhadamente todos os componentes, bibliotecas e dependências de um software; e o KBOM (*Kubernetes Bill of Materials,* Lista de Materiais do Kubernetes), que mapeia os elementos estruturais de um *cluster* Kubernetes, como componentes do

plano de controle, componentes de nó e addons, incluindo suas versões e imagens [1] - [3].

Embora SBOMs e KBOMs ofereçam visibilidade segmentada em nível dos componentes internos da aplicação e da infraestrutura Kubernetes, respectivamente, a gestão separada desses documentos pode tornar o processo de validação e análise de vulnerabilidades mais fragmentado e ineficiente. Assim, o objetivo deste trabalho é propor um único documento, chamado de IBOM (*Infrastructure Bill of Materials,* Lista de Materiais de Infraestrutura), que consolide todos os componentes da infraestrutura, ou seja, que integre os conceitos de SBOM e KBOM para facilitar a análise de vulnerabilidades em ambientes Kubernetes. A solução centraliza a geração, unificação, assinatura e verificação da infraestrutura, buscando simplificar auditorias, aumentar a segurança e evitar processos redundantes na análise de múltiplos componentes.

O trabalho está organizado em seções que descrevem a fundamentação teórica com os principais conceitos sobre o tema; as ferramentas utilizadas, como Trivy, *sbomasm* (SBOM *Assembler*) e Cosign, responsáveis respectivamente pela geração, unificação e assinatura dos documentos; a arquitetura do sistema desenvolvido, detalhando as funcionalidades implementadas, como geração, assinatura, verificação, análise e separação dos documentos; e os resultados experimentais que descrevem o ambiente de testes com cargas de trabalho (*workloads*) reais e compara os resultados obtidos com e sem a unificação dos materiais. Por fim, a seção de conclusões discute os benefícios da abordagem proposta, suas limitações e aponta possíveis melhorias futuras.

2. Fundamentação Teórica

O SBOM consiste em uma lista estruturada e detalhada de todos os componentes de um software, incluindo bibliotecas, componentes de código aberto, pacotes de terceiros e suas respectivas versões, etc [1]. A Figura 1 mostra a composição de um SBOM.

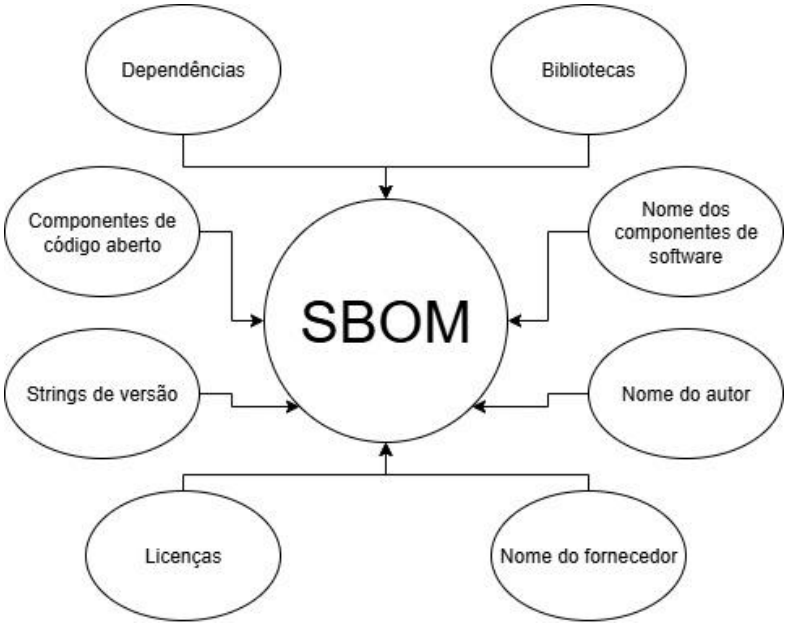

Figura 1. Composição de uma Lista de Materiais de Software (SBOM)

A principal contribuição do SBOM é oferecer visibilidade e rastreabilidade sobre as dependências utilizadas, permitindo a rápida identificação de vulnerabilidades conhecidas, frequentemente catalogadas no sistema CVE (*Common Vulnerabilities and Exposures*, Vulnerabilidades e Exposições Comuns), que padroniza e organiza falhas de segurança em softwares e bibliotecas. Um exemplo é o CVE-2021-44228, que corresponde à falha crítica no Log4j (Log4Shell), descoberta em 2021.

Para exemplificar de forma genérica, considere um servidor web *nginx* 1.21.0 que depende das bibliotecas OpenSSL 1.1.1k (para conexões seguras HTTPS/SSL) e zlib 1.2.11 (para compressão de dados). Caso seja descoberta uma vulnerabilidade crítica no OpenSSL 1.1.1k, o SBOM possibilita responder de forma imediata à questão: "Esta versão vulnerável está presente no meu sistema?". A partir dessa visibilidade, a organização pode adotar medidas corretivas, como atualização da versão, aplicação de patches ou isolamento do componente afetado.

A relevância prática desse recurso foi evidenciada em incidentes amplamente divulgados, como a vulnerabilidade Log4Shell, que afetou milhões de sistemas em empresas como Apple, Amazon e Minecraft. Essa falha permitia a execução remota de código em aplicações Java que utilizavam o Log4j e expôs a dificuldade das organizações em identificar, de forma ágil, se seus sistemas estavam ou não utilizando a biblioteca comprometida. Nesse contexto, o SBOM se apresenta como um mecanismo essencial para reduzir o tempo de resposta a incidentes, aumentando a capacidade de mitigação de riscos na cadeia de suprimentos de software. Sem um SBOM, seria necessário vasculhar código; verificar configurações de *build*; abrir imagens de contêiner para ver bibliotecas embutidas, etc.

Além da gestão de vulnerabilidades, o SBOM desempenha papel estratégico em diferentes dimensões da segurança e governança de software, tais como:

- Identificação proativa de vulnerabilidades em componentes de terceiros;
- Garantia de conformidade com licenças de software, reduzindo riscos legais;
- Facilitação de auditorias e manutenções, ao fornecer documentação precisa das dependências;
- Mitigação de riscos na cadeia de suprimentos, ao aumentar a transparência e reduzir a probabilidade de inclusão de dependências maliciosas ou não autorizadas.

Enquanto o SBOM descreve bibliotecas e dependências de software de uma aplicação, o KBOM descreve os principais componentes de um *cluster* Kubernetes, incluindo plano de controle (*control plane*), nós (*nodes*) e complementos (*addons*), com suas respectivas versões e imagens. Ele permite que equipes de segurança identifiquem rapidamente vulnerabilidades associadas à infraestrutura em componentes de *cluster*, como as registradas no CVE [2]. Por exemplo, o CVE-2018-1002105 é uma vulnerabilidade crítica no *kube-apiserver*, que permitia escalonamento de privilégios via conexão proxy persistente, impactando diretamente a segurança da infraestrutura Kubernetes. Correções foram introduzidas nas versões v1.10.11, v1.11.5 e v1.12.3, com recomendações urgentes de atualização. A falha evidencia a importância de inventários como KBOM, já que permite identificar rapidamente componentes vulneráveis e responder a riscos de segurança na orquestração do *cluster*.

O KBOM surge da necessidade de visibilidade na cadeia de suprimentos da infraestrutura, uma vez que falhas não ocorrem apenas em software de aplicação, mas também em sistemas de orquestração como Kubernetes. O Kubernetes é complexo, e a maioria dos administradores não têm plena consciência de todos os componentes individuais que compõem seu *cluster*. Sem o KBOM, é possível até estar analisando vulnerabilidades nas aplicações, mas não na infraestrutura [2] , [3].

O Kubernetes é uma plataforma de orquestração de contêineres de código aberto que automatiza a implantação, o dimensionamento e o gerenciamento de aplicativos em contêineres. No desenvolvimento de aplicativos modernos, é uma ferramenta essencial para aprender a gerenciar várias configurações de infraestrutura. Originalmente desenvolvido pelo Google, se tornou o padrão de fato para a execução de contêineres em escala. Ele abstrai a complexidade do gerenciamento de contêineres individuais e permite que os desenvolvedores se concentrem na criação e na implantação de seus aplicativos.

O cenário atual de segurança de infraestrutura Kubernetes é dominado por ferramentas de detecção de configurações incorretas (*misconfiguration*) e *hardening*, mas há uma lacuna quando se trata de avaliação de vulnerabilidades dos componentes do próprio *cluster*. O Kubernetes já foi chamado de "sistema operacional da nuvem". Ele é uma peça crítica de *middleware* entre diversas camadas fundamentais, como rede, *runtime* de contêiner, infraestrutura em nuvem, armazenamento e outras [3]. Isso posiciona o Kubernetes como um componente central no modelo de ameaças. O KBOM surge justamente nesse contexto, para responder questões como: Qual versão do *kube-apiserver* está em execução? Qual *flavor* do *kubelet* roda em cada nó? Qual *plugin* de rede está sendo utilizado? E, o mais importante: existem vulnerabilidades conhecidas que afetam esses componentes?

3. Ferramentas Estudadas

Nesta seção são apresentadas as ferramentas mais populares e amplamente utilizadas para geração e análise de SBOMs e KBOMs, além de ferramentas de unificação (*merge*) e, por fim, aquela voltada para assinatura digital. Cada grupo é detalhado nas subseções seguintes, destacando sua aplicação no contexto do trabalho. Também, são apresentados os resultados de diversos testes comparativos realizados com as ferramentas estudadas para a escolha da melhor ferramenta de geração e análise de SBOM e KBOM a ser utilizada neste trabalho.

3.1. Ferramentas de Geração e Análise de SBOM e KBOM

A geração e análise de SBOMs e KBOMs foram realizadas com ferramentas amplamente utilizadas e reconhecidas na comunidade de segurança. Entre as ferramentas analisadas, destacam-se Trivy [4], Syft [5], Grype [6], SBOM-tool [7], BOM Kubernetes [8] e KBOM (RAD Security) [9].

O Trivy, desenvolvido pela Aqua Security, é uma ferramenta de código aberto (*open source*) que se sobressai por sua versatilidade e desempenho. O Trivy permite a geração de SBOMs em diferentes padrões, como CycloneDX [10] e SPDX (*Software Package Data Exchange*) [11], realizando análises de vulnerabilidades a partir desses

documentos [4]. Além disso, é a única ferramenta gratuita e de código aberto capaz de gerar KBOMs, apenas para o padrão CycloneDX, o que a torna especialmente relevante para contextos de análise de infraestrutura Kubernetes [2], [3].

O SPDX e o CycloneDX são padrões abertos amplamente utilizados para representar SBOMs. O SPDX foi originalmente desenvolvido como uma forma de gerenciar licenças de software de código aberto e compartilhar informações sobre os pacotes utilizados. Por outro lado, o CycloneDX permite a criação de SBOMs com informações detalhadas sobre os componentes de software, sendo voltado especialmente à segurança da cadeia de suprimentos [10], [11].

O Syft, mantido pela Anchore, é voltado especificamente para a geração de SBOMs de imagens de contêiner e sistemas de arquivos. Ele fornece relatórios detalhados sobre os pacotes e dependências presentes no seu software e pode ser utilizado em conjunto com o Grype, que realiza a análise de vulnerabilidades com base nos SBOMs gerados. Ambos formam uma dupla eficiente, sendo Syft responsável pela geração e Grype pela avaliação de segurança [5], [6].

O SBOM-tool da Microsoft é focado na geração de SBOMs durante o processo de *build* de software. Utiliza o padrão SPDX e é indicado para ambientes empresariais que exigem integração com processos automatizados de desenvolvimento [7].

A ferramenta BOM Kubernetes, desenvolvida pelo SIG do Kubernetes, permite gerar SBOMs a partir de arquivos, diretórios ou imagens. Possui um classificador de licenças embutido e é capaz de gerar documentos compatíveis com o padrão SPDX [8].

Em relação à geração de KBOMs, a ferramenta KBOM da RAD Security fornece um mapeamento detalhado dos componentes do *cluster* Kubernetes [9]. Porém, sua utilização prática ainda é limitada, especialmente se comparada ao Trivy, que além de gerar KBOMs, os analisa. Esta ferramenta não apresenta compatibilidade com o Trivy para análise de vulnerabilidades.

Na realização desta pesquisa, diversos testes comparativos foram realizados para a escolha da melhor ferramenta de geração e análise de SBOM e KBOM. Para esses testes foi utilizada a imagem `registry.k8s.io/kube-apiserver:v1.27.3` e o repositório do SPIRE (*SPIFFE Runtime Environment*) 1.6.0, sendo gerado pelo comando `trivy fs` que faz uma varredura do diretório local.

A imagem `registry.k8s.io/kube-apiserver:v1.27.3`, hospedada no *registry* oficial do Kubernetes (registry.k8s.io) e referente à versão 1.27.3, corresponde ao servidor de API (*Applicatiom Programming Interface*) do Kubernetes, componente central do plano de controle, responsável por expor a API do cluster, gerenciar o estado desejado dos objetos (como *pods*, *deployments* e *services*) e processar requisições de usuários, do *kubectl* e de outros componentes do cluster.

O SPIRE é um conjunto de ferramentas que implementa o padrão SPIFFE (*Secure Production Identity Framework for Everyone*) com o objetivo de estabelecer confiança entre sistemas de software em diferentes ambientes, como servidores, máquinas virtuais, containers ou clusters Kubernetes. Ele disponibiliza a API de carga de trabalho SPIFFE, responsável por atestar a identidade de aplicações e serviços e emitir identificadores confiáveis, os IDs SPIFFE e SVIDs (*SPIFFE Verifiable Identity*

Documents), que permitem que cargas de trabalho se autentiquem e se comuniquem com segurança.

As tabelas e figuras a seguir apresentam os resultados dos testes realizados com as ferramentas analisadas (Trivy, Syft, Grype, SBOM-tool, KBOM/Rad-Security e BOM Kubernetes). São apresentadas tabelas comparativas dos aspectos avaliativos de interesse, tais como: área de atuação, efeito, padrão adotado, compatibilidade com softwares, severidade das vulnerabilidades e se as vulnerabilidades estão corrigidas ou não.

A Tabela 1 compara as ferramentas estudadas de acordo com a área de atuação. Observa-se que o Trivy pode operar em todas as áreas, o que representa uma vantagem em relação às demais ferramentas. As ferramentas Syft, Grype, Bom Kubernetes e SBOM-tools trabalham com *scanner* de imagens e arquivos de código fonte. O Rad-Security opera apenas com o Kubernetes.

A Tabela 2 compara as ferramentas estudadas de acordo com o efeito, ou seja, geração de SBOM/KBOM ou análise desses documentos. Nota-se que o Trivy realiza tanto a geração quanto a análise, enquanto as outras ferramentas realizam apenas geração, com exceção do Grype, que realiza apenas a análise.

Tabela 1. Comparação das ferramentas estudadas de acordo com a área de atuação

Área de Atividade	**Softwares**					
	Trivy	Syft	Grype	Bom Kubernetes	SBOM-tool	KBOM/Rad-Security
Kubernetes	X	-	-	-	-	X
Imagem	X	X	X	X	X	-
Código fonte	X	X	X	X	X	-
Repositório remoto	X	-	-	-	-	-

Tabela 2. Comparação das ferramentas estudadas de acordo com o efeito (geração/análise)

Efeito	**Softwares**					
	Trivy	Syft	Grype	Bom Kubernetes	SBOM-tool	KBOM/Rad-Security
Geração	X	X	-	X	X	X
Análise	X	-	X	-	-	-

A Tabela 3 apresenta as ferramentas estudadas de acordo com os padrões utilizados. Observa-se que Trivy, Syft e Grype utilizam ambos os padrões (SPDX e CycloneDX).

Tabela 3. Comparação das ferramentas estudadas de acordo com os padrões adotados

Padrão	**Softwares**					
	Trivy	Syft	Grype	Bom Kubernetes	SBOM-tool	KBOM/Rad-Security
SPDX	X	X	X	X	X	-
CYCLONEDX	X	X	X	-	-	X

A Tabela 4 apresenta a compatibilidade entre softwares. Podemos observar as seguintes combinações compatíveis entre as ferramentas:

- Trivy (geração) / Trivy (análise)
- Trivy (geração) / Grype (análise)
- Syft (geração) / Trivy (análise)
- Syft (geração) / Grype (análise)
- SBOM-tool (geração) / Trivy (análise)
- SBOM-tool (geração) / Grype (análise)

Tabela 4. Comparação de compatibilidade entre as ferramentas

Compatibilidade dos Softwares		
Geração	**Análise**	
	Trivy	Grype
Trivy	X	X
Syft	X	X
Bom Kubernetes	-	-
KBOM/Rad-Security	-	-
SBOM-tool	X	X

Para fins comparativos, foram realizados testes com os softwares que apresentaram compatibilidade. Dois SBOMs foram gerados a partir da imagem `registry.k8s.io/kube-apiserver:v1.27.3`, pertencente à infraestrutura do *cluster*. A Tabela 5 mostra melhor desempenho das ferramentas Trivy (geração) / Trivy (análise) em relação ao número de vulnerabilidades encontradas. O Trivy prioriza a severidade atribuída pelo fornecedor, enquanto as pontuações do CVSS3 podem estar indisponíveis ou inconsistentes entre os alertas de segurança do NVD e do fornecedor. Além disso, existem diferenças fundamentais entre o CVSS v2 e v3. É importante ressaltar que as propriedades do CVSS (pontuações e vetores) são fornecidas apenas para fins informativos e não foram projetadas para comparação direta de severidade.

A Figura 2 apresenta o gráfico com os resultados da Tabela 5. Foram realizados quatro testes:

- Trivy (geração) / Trivy (análise)
- Trivy (geração) / Grype (análise)
- Syft (geração) / Grype (análise)
- Syft (geração) / Trivy (análise)

Os resultados são classificados conforme a severidade das vulnerabilidades encontradas: Crítica, Alta, Média, Baixa ou Desconhecida.

Tabela 5. Comparação das ferramentas Trivy e Syft/Grype de acordo com a severidade das vulnerabilidades

Severidade	**Softwares**			
	Trivy/Trivy	Trivy/Grype	Syft/Grype	Syft/Trivy
Crítica	3	1	3	2
Alta	14	6	20	15
Média	43	7	21	24
Baixa	1	1	2	2
Desconhecida	1	0	1	1
Total	62	15	47	44

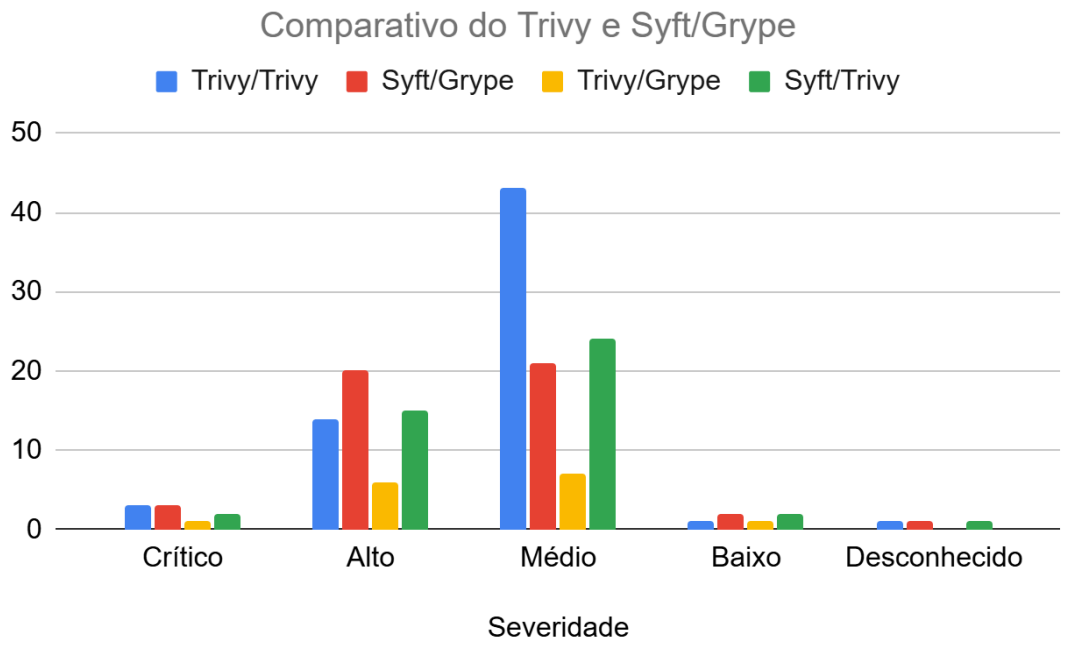

Figura 2. Comparativo dos resultados do Trivy e Syft/Grype segundo o número de vulnerabilidades encontradas

Testes foram realizados com o SPIRE versão 1.6.0, que representa uma aplicação no *cluster*, para fins de comparação. Na Tabela 6 observa-se um desempenho expressivo da combinação Syft (geração) / Grype (análise) em relação às vulnerabilidades encontradas. A Figura 3 mostra o gráfico com os resultados da Tabela 6. Foram realizados cinco testes:

- SBOM-tool (geração) / Trivy (análise)
- SBOM-tool (geração) / Grype (análise)
- Trivy (geração) / Trivy (análise)
- Trivy (geração) / Grype (análise)
- Syft (geração) / Grype (análise)

Não foi possível incluir o teste Syft/Trivy na comparação, devido a incompatibilidade. Os resultados foram comparados de acordo com o número de vulnerabilidades encontradas nos níveis: Crítica, Alta, Média, Baixa ou Desconhecida.

Tabela 6. Comparação das ferramentas Trivy, Syft/Grype e SBOM-tool de acordo com a severidade das vulnerabilidades

Severidade	Softwares				
	SBOM-tool /Trivy	SBOM-tool/ Grype	Trivy/Trivy	Trivy/Grype	Syft/Grype
Crítica	7	7	4	4	17
Alta	31	31	13	18	43
Média	30	30	30	33	59
Baixa	5	5	2	2	2
Desconhecida	0	0	0	0	2
Total	73	73	49	57	123

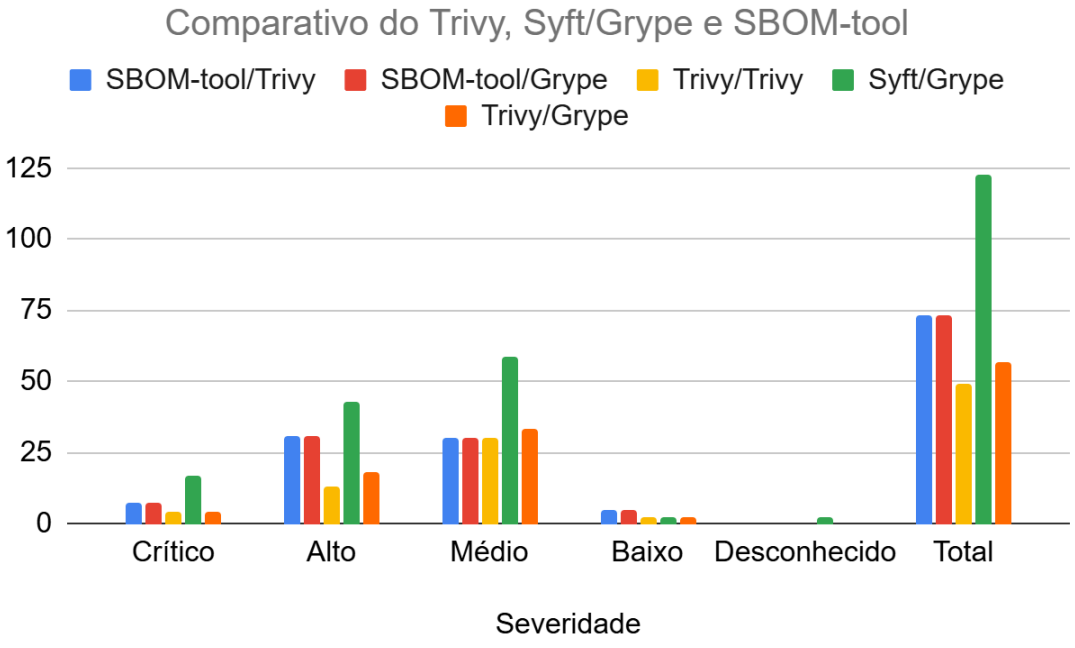

Figura 3. Comparativo dos resultados do Trivy, Syft/Grype e SBOM-tool segundo o número de vulnerabilidades encontradas

A Tabela 7 mostra uma comparação entre as ferramentas em relação ao status das vulnerabilidades, classificadas como corrigidas ou não corrigidas. As ferramentas Trivy (geração) / Trivy (análise) e Syft (geração) / Grype (análise) apresentaram, respectivamente, 46 e 120 vulnerabilidades corrigidas. As ferramentas detectaram 3 vulnerabilidades não corrigidas. As combinações SBOM-tool / Trivy e SBOM-tool /

Grype apresentaram os mesmos resultados: 62 vulnerabilidades corrigidas e 11 não corrigidas. Na Figura 4 observam-se os resultados da Tabela 7.

Tabela 7. Comparação das ferramentas estudadas de acordo com o status das vulnerabilidades

Status	Softwares				
	SBOM-tool /Trivy	SBOM-tool /Grype	Trivy/Trivy	Trivy/Grype	Syft/Grype
Corrigido	62	62	46	54	120
Não corrigido	11	11	3	3	3

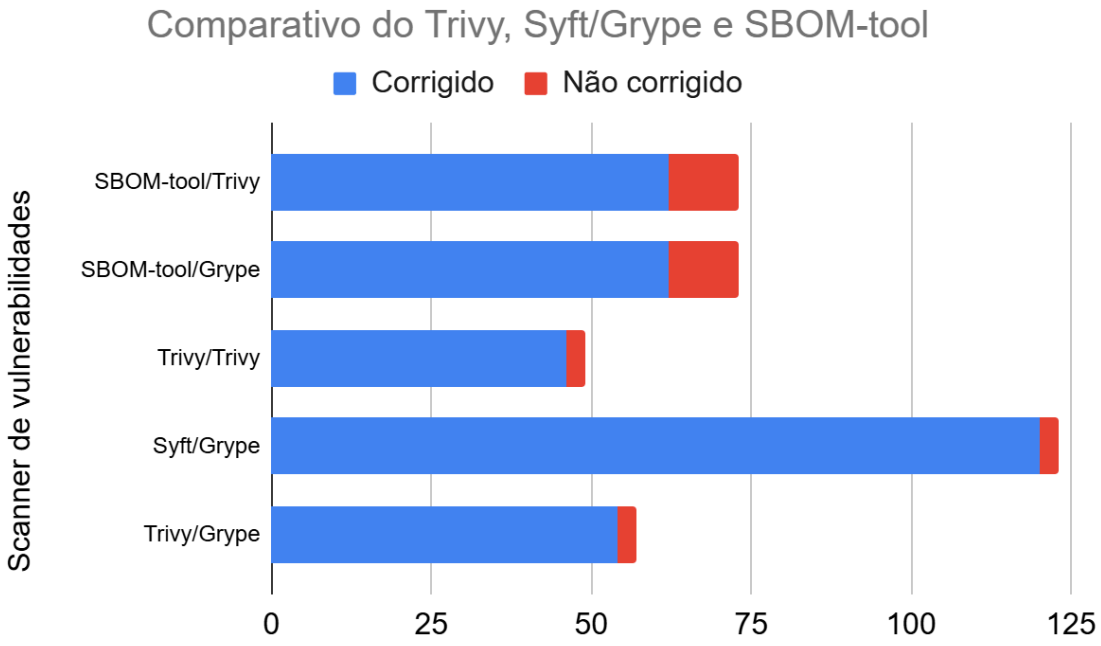

Figura 4. Comparativo dos resultados do Trivy, Syft/Grype e SBOM-tool segundo o status das vulnerabilidades

Apesar do Trivy não ter apresentado o maior número de vulnerabilidades corrigidas nos testes comparativos, optou-se por utilizá-lo como ferramenta principal de geração e análise no fluxo automatizado do projeto. Essa escolha se justifica por sua abrangência, suporte a múltiplos formatos, capacidade de gerar e analisar tanto SBOMs quanto KBOMs, e eficácia na detecção de vulnerabilidades. As representações gráficas incluídas nesta seção reforçam essas conclusões, evidenciando algumas vantagens do Trivy em relação às demais ferramentas avaliadas de acordo com o que foi justificado.

3.2. Ferramentas de Merge de SBOMs

O processo de unificação, chamado de *merge* dos SBOMs, em um único documento é fundamental para simplificar a análise e a gestão de vulnerabilidades em ambientes com múltiplos componentes. Neste sentido, duas ferramentas foram estudadas:

- CycloneDX CLI: Suporta funcionalidades como: *merge*, análise, conversão de formato e assinatura. Apesar de ser robusta, é desenvolvida em C#, o que pode dificultar sua integração com outras aplicações escritas em Go, como o orquestrador utilizado neste projeto [12].
- SBOM Assembler (*sbomasm*): Ferramenta escrita em Go com foco em montagem de SBOMs a partir de múltiplos arquivos de entrada. Foi a escolhida neste projeto por ser mais simples de integrar à solução, além de fornecer resultados equivalentes ao CycloneDX CLI. Seu principal ponto negativo é a limitação de suporte à versão 1.5 do padrão CycloneDX [13].

Ambas as ferramentas apresentaram os mesmos resultados iniciais, porém, pensando em um processo de integração futuro, optamos por fazer uso do SBOM Assembler, devido a linguagem de programação utilizada. Optamos pelo modo hierárquico, pois o modo *flat* remove as duplicatas e não permite a recomposição dos SBOMs individuais.

3.3. Ferramenta de Assinatura e Verificação de Documentos

Para garantir a integridade e autenticidade dos documentos gerados (SBOM, KBOM ou IBOM), foi utilizada a ferramenta Cosign, desenvolvida como parte do projeto Sigstore (um projeto de código aberto para aprimorar a segurança da cadeia de suprimentos de software) [14]. O Cosign permite a assinatura e verificação de artefatos de forma segura, suportando dois modos [15]:

- Assinatura com chave (*key pair*): Utiliza um par de chaves (pública e privada) gerado localmente.
- Assinatura sem chave (*keyless*): Baseada em autenticação via OIDC (OpenID *Connect*) com provedores como Google, GitHub ou Microsoft. OIDC é um protocolo moderno de autenticação que permite verificar a identidade de usuários ou serviços de forma segura. Ideal para integração com *pipelines* CI (*Continuous Integration*) / CD (*Continuous Delivery*), que facilita integração de ferramentas de assinatura ou acesso a recursos, garantindo que apenas identidades autenticadas possam executar ações sensíveis.

A assinatura e a verificação dos documentos utilizaram as funções `cosign sign-blob` para a assinatura e `cosign verify-blob` para a verificação dos SBOMs.

4. Arquitetura Proposta do IBOM

O processo de construção do IBOM (*Infrastructure Bill of Materials*) a partir da integração entre o KBOM e os SBOMs das cargas de trabalho em execução no cluster é exemplificado na Figura 5.

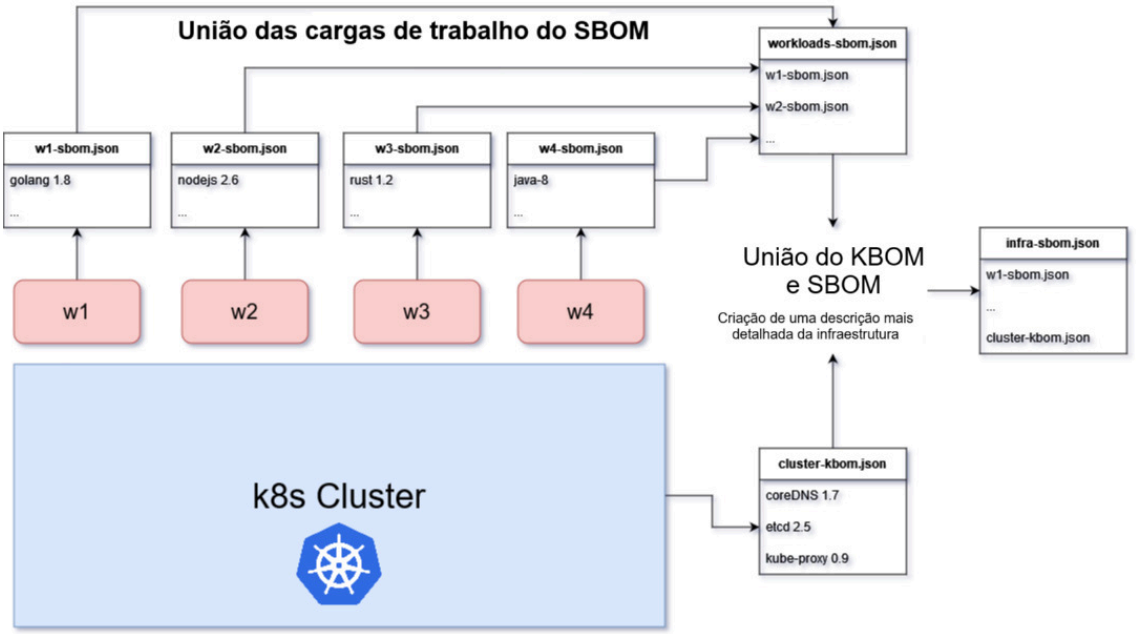

Figura 5. Arquitetura proposta do IBOM

Inicialmente, cada carga de trabalho em execução no cluster Kubernetes (**w1**, **w2**, **w3**, **w4**) possui seu próprio SBOM gerado (por exemplo, **w1-sbom.json**, **w2-sbom.json**), no qual estão descritas as dependências de software utilizadas pela aplicação, como versões específicas de linguagens de programação e bibliotecas (por exemplo, **golang 1.8**, **nodejs 2.6**, **rust 1.2** e **java-8**). Esses SBOMs individuais são consolidados em um artefato denominado **workloads-sbom.json**, que representa o inventário agregado dos softwares empregados pelas cargas de trabalho do cluster.

Paralelamente, é gerado o KBOM do cluster (**cluster-kbom.json**), que contém informações sobre os componentes essenciais da infraestrutura Kubernetes, tais como **coreDNS 1.7**, **etcd 2.5** e **kube-proxy 0.9**.

A etapa seguinte consiste na união do KBOM com os SBOMs consolidados. O **workloads-sbom.json** (software das aplicações) é combinado com o **cluster-kbom.json** (infraestrutura do cluster), resultando no artefato **infra-sbom.json**. Esse artefato, denominado IBOM, descreve de forma unificada tanto os elementos de infraestrutura do cluster quanto as dependências de software das cargas de trabalho nele executadas.

A arquitetura da solução foi desenvolvida em Go para criar, consolidar e validar o IBOM em ambientes Kubernetes, para o padrão CycloneDX. Nesta seção será descrito como integramos as ferramentas selecionadas, formando um fluxo automatizado que

engloba geração, merge, assinatura, análise e separação de artefatos: o Trivy para descoberta e análise de SBOM/KBOM; o SBOM Assembler para unificação de documentos; e o Cosign para assinatura e verificação.

Inicialmente, detalharemos o papel de cada ferramenta no *pipeline*:

- Trivy extrai SBOMs das imagens e do *cluster* (KBOM), além de realizar análise de vulnerabilidades;
- SBOM Assembler recebe os SBOMs e KBOMs, executa o merge no formato CycloneDX e entrega o documento unificado denominado IBOM;
- Cosign, por fim, aplica assinaturas digitais garantindo a integridade e autenticidade do IBOM.

Em seguida, serão apresentados os principais componentes da interface desenvolvida (funções: **generate**, **sign**, **verify**, **split**, **gen-keypair**, **analyze**), explicando seu funcionamento e a orquestração das etapas. Por fim, será mostrado como o sistema lida com desafios técnicos, como o suporte a múltiplos sistemas operacionais.

4.1. Funcionalidades do Software Desenvolvido

O software desenvolvido possui funcionalidades voltadas à geração, manipulação, assinatura e análise do IBOM.

A função generate segue as seguintes etapas, como observado na Figura 6:

- Identifica as imagens, aplicações em execução no *cluster* (excluindo o namespace kube-system), e utiliza o Trivy para gerar os respectivos SBOMs com o comando Trivy image;
- Gera o KBOM do *cluster* por meio do comando Trivy k8s;
- Realiza o merge entre os SBOMs e o KBOM, resultando no documento unificado denominado IBOM.

A **função sign** permite a assinatura digital do IBOM, ou de qualquer outro documento, com suporte tanto para assinaturas baseadas em chaves pública e privada quanto para autenticação via OIDC, utilizando provedores como GitHub, Google ou Microsoft. Essa assinatura é realizada por meio da ferramenta Cosign, que garante a integridade e autenticidade do documento.

A **função verify** realiza a verificação da assinatura digital do documento, também utilizando o Cosign. Caso qualquer modificação seja detectada, o sistema retorna um erro, assegurando que o conteúdo foi alterado após a assinatura.

A **função split** executa a operação inversa ao generate, separando o IBOM novamente em seus componentes individuais com base no mapeamento dos componentes.

A **função gen-keypair** é responsável pela geração de um par de chaves, pública e privada, para uso na assinatura digital com Cosign no modo baseado em chave.

Por fim, a **função analyze** verifica se o IBOM foi assinado, e caso esteja em conformidade, será utilizada a função split, gerando os resultados individuais e consolidando-os de maneira unificada.

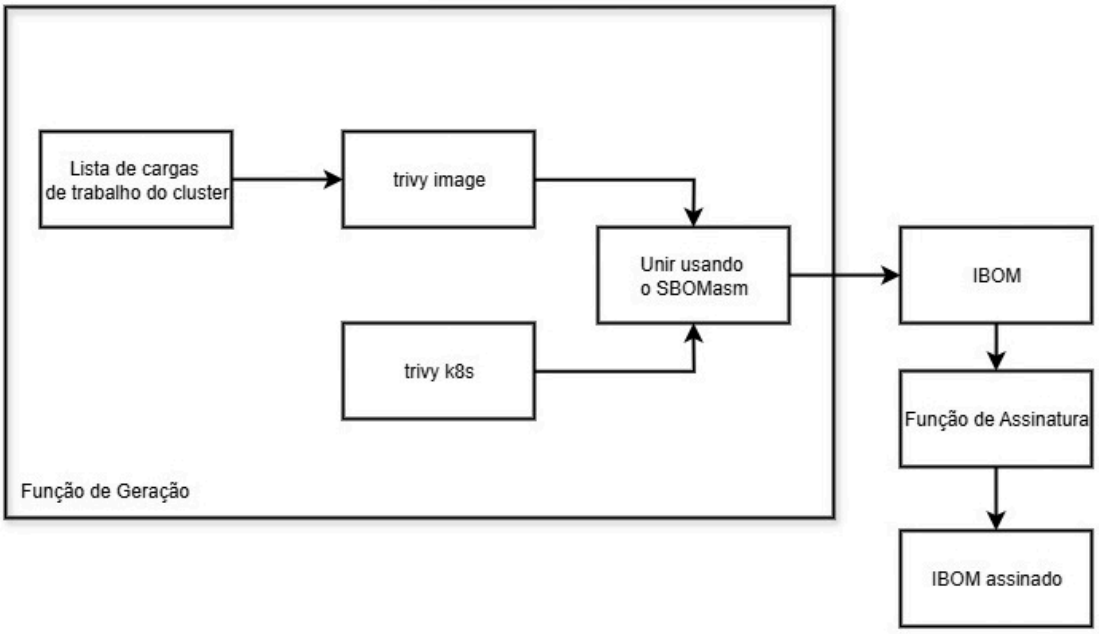

Figura 6. Processo de geração e assinatura do IBOM

Conforme ilustrado na Figura 7, o processo de análise do IBOM ocorre nas seguintes etapas:

- Verificação da assinatura, garantindo a integridade do documento.
- Separação do IBOM, sendo necessário devido à limitação do comando Trivy sbom, que aceita apenas um sistema operacional por vez.
- Análise com Trivy, onde cada SBOM é analisado individualmente.
- Unificação dos resultados, sendo consolidados os dados obtidos, removendo duplicações.

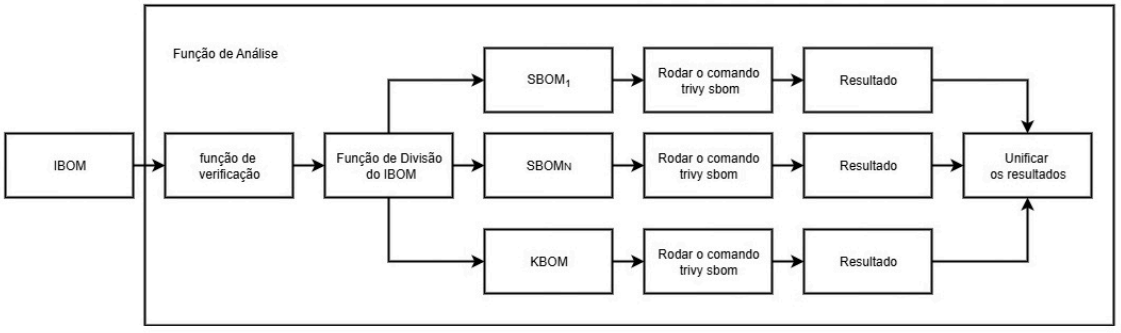

Figura 7. Processo de análise do IBOM

5. Resultados Obtidos

Os experimentos foram conduzidos em um ambiente Kubernetes composto por um plano de controle e um nó de trabalho. As imagens utilizadas nos *pods* foram:

- **Redis**, baseado no sistema operacional Debian 12.11
- **Httpd**, também baseado em Debian 12.11
- **Nginx:1.19.6-alpine**, baseado em Alpine Linux 3.12.3

A Tabela 8 apresenta o total de vulnerabilidades identificadas nas imagens dos componentes Redis, Httpd, Nginx (SBOMs), bem como no *cluster* Kubernetes (KBOM), em dois cenários distintos de análise:

- **Split + Trivy sbom:** o IBOM foi gerado, dividido com a função split e os arquivos resultantes analisados com Trivy sbom.
- **Trivy sbom:** os SBOMs e o KBOM foram gerados e analisados individualmente com o Trivy, sem o uso do IBOM.

Tabela 8. Total de vulnerabilidades identificadas nos SBOMs e KBOM por imagem/sistema

SBOM/KBOM	sistema	função split + Trivy sbom	Trivy sbom
Kubernetes	ubuntu 22.04.4	2	2
Redis	debian 12.11	80	80
Httpd	debian 12.11	125	125
Nginx	alpine 3.12.3	92	92

Como Redis e Httpd utilizam a mesma base de sistema operacional (Debian 12.11), o IBOM gerado não diferencia automaticamente entre os dois contextos. Para evitar ambiguidade na estrutura de dependências, optamos por construir uma lista de dependências a partir dos componentes individualmente, utilizando suas referências. Dessa forma, foi possível manter a separação lógica entre os ambientes e preservar a rastreabilidade dos pacotes até seus respectivos SBOMs de origem. Embora o ideal fosse adotar uma estrutura hierárquica com os componentes aninhados sob cada sistema ou aplicação, essa abordagem foi inviabilizada justamente pela sobreposição na camada de sistema operacional, o que tornou necessário o uso de uma estrutura plana, sem aninhamento.

Considerando a análise do documento IBOM, foi obtido um resultado unificado de 219 vulnerabilidades, pois o Redis e o Httpd apresentaram as mesmas vulnerabilidades. O mapeamento dessas vulnerabilidades é realizado a partir de duas variáveis: VulnerabilityID que está no formato CVE; e PkgID, que representa o pacote de software vulnerável.

6. Conclusões

Este trabalho apresentou uma abordagem unificada para geração, análise e validação de vulnerabilidades em ambientes Kubernetes por meio da criação do IBOM, consolidando os conceitos de SBOM e KBOM em um único documento estruturado.

Ao integrar ferramentas como Trivy, *sbomasm* e Cosign, foi possível automatizar todo o fluxo de geração, assinatura e análise de documentos de segurança, promovendo maior visibilidade sobre os componentes que integram a infraestrutura e suas respectivas vulnerabilidades. O Trivy oferece uma solução completa para geração e análise de SBOM e KBOM, eliminando a necessidade de ferramentas adicionais, o que simplifica o todo o processo.

Os experimentos realizados mostraram que a utilização de um IBOM facilita a centralização e o tratamento das informações, especialmente ao lidar com múltiplos componentes em sistemas operacionais distintos.

A proposta do IBOM demonstrou ser promissora, especialmente para cenários que exigem controle, rastreabilidade e segurança na cadeia de suprimento de software. Como trabalho futuro, destaca-se a possibilidade de realizar a integração das ferramentas Trivy, *sbomasm* e Cosign ao software proposto e avaliar a possibilidade de integração do IBOM com validações automáticas via políticas (ex.: *Sigstore Policy Controller*).

Com isso, conclui-se que a unificação de SBOMs e KBOMs em um único documento assinável, verificável e analisável amplia significativamente a capacidade de governança e segurança em ambientes de infraestrutura modernos, ao invés do usuário fazer esse processo repetidas vezes para cada SBOM.

Referências

[1] AQUA SECURITY. Software supply chain security with Trivy and SBOM. Disponível em: https://www.aquasec.com/blog/software-supply-chain-security-trivy-sbom/. Acesso em: 25 jul. 2025.

[2] AQUA SECURITY. Scanning KBOM for vulnerabilities with Trivy. Disponível em: https://www.aquasec.com/blog/scanning-kbom-for-vulnerabilities-with-trivy/. Acesso em: 25 jul. 2025.

[3] AQUA SECURITY. Introducing KBOM: Kubernetes Bill of Materials. Disponível em: https://www.aquasec.com/blog/introducing-kbom-kubernetes-bill-of-materials/. Acesso em: 25 jul. 2025.

[4] AQUA SECURITY. Trivy: find vulnerabilities, misconfigurations, secrets, SBOM in containers, Kubernetes, code repositories, clouds and more. Disponível em: https://github.com/aquasecurity/trivy. Acesso em: 25 jul. 2025.

[5] ANCHORE. Syft: CLI tool and library for generating SBOMs. Disponível em: https://github.com/anchore/syft. Acesso em: 25 jul. 2025.

[6] ANCHORE. Grype: vulnerability scanner for container images and filesystems. Disponível em: https://github.com/anchore/grype. Acesso em: 25 jul. 2025.

[7] MICROSOFT. SBOM Tool. Disponível em: https://github.com/microsoft/sbom-tool. Acesso em: 25 jul. 2025.

[8] KUBERNETES SIGS. BOM: Kubernetes-native SBOM generator. Disponível em: https://github.com/kubernetes-sigs/bom. Acesso em: 25 jul. 2025.

[9] RAD SECURITY. KBOM: Kubernetes Bill of Materials generator. Disponível em: https://github.com/rad-security/kbom. Acesso em: 25 jul. 2025.

[10] CYCLONEDX. CycloneDX Specification. Disponível em: https://cyclonedx.org/. Acesso em: 25 jul. 2025.

[11] SPDX. Software Package Data Exchange (SPDX) Project. Disponível em: https://spdx.dev/. Acesso em: 25 jul. 2025.

[12] CYCLONEDX. CycloneDX CLI Tool. Disponível em: https://github.com/CycloneDX/cyclonedx-cli. Acesso em: 25 jul. 2025.

[13] INTERLYNK. SBOM Assembler (sbomasm). Disponível em: https://github.com/interlynk-io/sbomasm. Acesso em: 25 jul. 2025.

[14] SIGSTORE. Sigstore overview. Disponível em: https://docs.sigstore.dev/about/overview/. Acesso em: 25 jul. 2025.

[15] SIGSTORE. Cosign: container signing tool. Disponível em: https://github.com/sigstore/cosign/tree/main. Acesso em: 25 jul. 2025.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
Campus Campina Grande - Código INEP: 25137409
R. Tranqüilino Coelho Lemos, 671, Dinamérica, CEP 58432-300, Campina Grande (PB)
CNPJ: 10.783.898/0003-37 - Telefone: (83) 2102.6200

Documento Digitalizado Ostensivo (Público)

Trabalho de conclusão de curso

Assunto:	Trabalho de conclusão de curso
Assinado por:	Marcelo Ribeiro
Tipo do Documento:	Anexo
Situação:	Finalizado
Nível de Acesso:	Ostensivo (Público)
Tipo do Conferência:	Cópia Simples

Documento assinado eletronicamente por:

- **Marcelo Ribeiro da Silva**, **ALUNO (202021250020) DE BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO - CAMPINA GRANDE**, em 24/08/2025 21:47:22.

Este documento foi armazenado no SUAP em 24/08/2025. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1583994
Código de Autenticação: 983bbfd6f5