# Segurança em imagens Docker: um estudo de ferramentas de análise estática

**Yuri R. Fialho**[1], **Jacir L. Bordim**[1]

[1]Departamento de Ciência da Computação – Universidade de Brasília (UnB)
Brasília – DF – Brasil

yurirfialho@gmail.com, bordim@unb.br

***Resumo.*** *O uso da tecnologia de contêineres experimentou uma adoção massiva nos últimos anos sendo o Docker uma das tecnologias mais utilizadas. O Docker usa o conceito de imagens, que as armazena de forma hierárquica para melhorar a reutilização. Apesar de seus benefícios, as imagens podem apresentar vulnerabilidades de segurança. Quando não tratadas, essas aplicações podem ter riscos em potencial. Embora existam várias ferramentas de análise estática para identificar vulnerabilidades em imagens, suas vantagens e deficiências não são claras, o que torna a tarefa de selecionar uma ferramenta adequada bastante desafiadora. Este trabalho avalia o desempenho de quatro soluções populares na literatura: Anchore Grype, Clair, Dagda e Snyk. A avaliação considera sua capacidade de identificar vulnerabilidades e usabilidade. Usando um conjunto de vulnerabilidades conhecidas nas imagens como linha de base (controle), as ferramentas são comparadas entre si. Os resultados mostram que Anchore Grype obteve o melhor resultado entre as ferramentas avaliadas com uma taxa de identificação (TIV) de 23% das vulnerabilidades do controle utilizado. Entre as ferramentas, Anchore Grype encontrou duas vezes mais vulnerabilidades do que a segunda melhor alternativa.*

***Abstract.*** *The use of container technology has experienced a massive adoption over the past years being Docker one of the most used technologies. Docker uses the concept of images, which stores them in a hierarchical way to improve reuse. Despite its benefits, images may present security vulnerabilities. When not treated, such applications can be a potential treat. Although there are several static analysis tools to identify vulnerabilities in images, their advantages and deficiencies are not clear, which makes the task of selecting an appropriate tool a challenging task. This work evaluates the performance of four popular solutions in the literature: Anchore Grype, Clair, Dagda and Snyk. The evaluation considers their ability to identify vulnerabilities and usability. Using a set of known vulnerabilities in the images as a baseline (control), the tools are compared against each other. The results show that Anchore Grype obtained the best result among the evaluated tools with an identification rate (TIV) of 23% of the vulnerability of the control used. Among the tools, Anchore Grype reported over two times more vulnerabilities than the second-best alternative.*

## 1. Introdução

O Docker é uma das tecnologias de contêineres mais utilizadas dentre as existentes [Sultan et al. 2019]. Ela utiliza o conceito de "imagem" que define como um contêiner

será criado e para que será utilizado. Tais informações são armazenadas em camadas, permitindo a sua reutilização em outros contêineres [Docker 2021a]. Entretanto, há a possibilidade de que os contêineres criados possuam falhas de segurança, que podem ocorrer devido à descoberta de vulnerabilidades ou, até mesmo, pela falta de conhecimento do profissional que construiu a imagem. Segundo [Prevasio 2020], cerca de 4 milhões de imagens hospedadas no Docker Hub [Docker 2021b] possuem vulnerabilidades críticas. Estas vulnerabilidades podem resultar em pontos que podem ser explorados por indivíduos mal-intencionados e resultar em perdas de diversas naturezas.

Para serem mitigados os riscos de possíveis invasões e perdas de dados, existem diversas ferramentas e métodos que realizam a análise das vulnerabilidades de imagens Docker. Dentre as ferramentas existentes, destacam-se as ferramentas em nuvens como a Twistlock [Twistlock 2021] e IBM's Vulnerability Advisor [IBM 2021], bem como outras proprietárias e, consequentemente, pagas. Dentre as ferramentas de código aberto e gratuitas, podemos citar a Clair [Clair 2021], Anchore Grype [Anchore 2021a] e a ferramentas desenvolvidas pelos próprios autores [Kwon and Lee 2020, Shu et al. 2017, Zheng et al. 2021]. Entretanto, apesar de haver diversas ferramentas, não há na literatura uma avaliação que denote de forma clara e objetiva as vantagens e deficiências de modo a permitir a escolha da ferramenta mais adequada. Neste sentido, verifica-se uma carência de estudos que identifiquem e respondam objetivamente esta questão, principalmente no aspecto quanto ao desempenho na capacidade de detectar vulnerabilidades [Sultan et al. 2019]. Além disso, existem soluções privadas que utilizam Docker e não podem ser submetidas para análise em nuvem, o que caracteriza outro problema. Pois, seja por questões legais, política organizacional ou por questões de segurança, dados sensíveis não podem ser extraídos ou submetidos a um terceiro. Neste sentido, diversos trabalhos [Brady et al. 2020, Shu et al. 2017, Tunde-Onadele et al. 2019] abordam a utilização de ferramentas de código aberto ou desenvolvidas pelos próprios autores. Entretanto, não há uma justificativa que motive a escolha da ferramenta ou uma comparação da solução abordada com outras. Portanto, surge a necessidade de que a ferramenta seja a melhor possível no aspecto de detectar vulnerabilidades existentes em imagens Docker e que execute localmente (i.e. *on premise*), em virtude das restrições as quais a organização estiver submetida.

Como principal contribuição, este trabalho apresenta uma análise comparativa entre ferramentas de código livre de avaliação de vulnerabilidade em imagens Docker com foco no desempenho, ou seja, na capacidade de detectar vulnerabilidades de maneira efetiva. O restante deste documento é organizado da seguinte forma. A Seção 2 apresenta os trabalhos relacionados e o diferencial deste trabalho perante outros. Nas Seções 3 e 4 são apresentados os principais conceitos e ferramentas que guiarão este estudo. A Seção 5 define os objetivos da avaliação, métricas, parâmetros e métodos utilizados. Na Seção 6 são apresentados os cenários e os resultados obtidos nos experimentos. Por fim, na Seção 7 é realizada uma conclusão do trabalho, com os principais resultados e a visão futura.

## 2. Trabalhos Relacionados

Segundo Sultan [Sultan et al. 2019], diversas soluções de aplicações estão migrando para a arquitetura em contêineres. Pois é uma tecnologia mais leve, de rápida disponibilização e permite uma melhor utilização dos recursos. Contudo, muito dos usuários desta tecnologia negligenciam a segurança, principalmente reusando diversas imagens prontas.

Portanto, surge a necessidade de verificar as imagens quanto a segurança, de modo a se evitar surpresas desagradáveis quando a aplicação já estiver em produção.

Apesar de haver diversas ferramentas [Twistlock 2021, IBM 2021, Clair 2021, Anchore 2021a, Kwon and Lee 2020, Shu et al. 2017, Zheng et al. 2021], há um questionamento que deve ser respondido de modo que o desenvolvedor ou profissional saiba qual a mais adequada. Conforme reportado em [Sultan et al. 2019], há uma carência de estudo que identifique e compare diferentes ferramentas quanto ao desempenho, usabilidade, automação e integração com ferramentas de implantação e orquestração. Tal fato, pode resultar na escolha de ferramentas inadequadas ou ineficientes e, consequentemente, no aumento do risco de segurança das soluções disponibilizadas.

Em [Brady et al. 2020] os autores apresentam um modelo de integração e implantação contínua que utiliza uma ferramenta de análise de vulnerabilidade de imagem Docker, para garantir a qualidade e evitar possíveis problemas de segurança. Entretanto, os autores utilizaram o CoreOS Clair [Clair 2021] e o Anchore Engine [Anchore 2021a], mas não há uma justificativa clara para a utilização destas ferramentas em detrimento de outras. Isto é, o estudo não apresenta elementos que permitam o estabelecimento de um comparativo.

Atualmente, há diversas ferramentas que prestam serviços e já realizam a verificação de segurança automaticamente quando o cliente disponibiliza uma imagem em seu registro de imagem, como a Google [Google 2021], a IBM [IBM 2021] e a Amazon [Aws 2021]. Entretanto, este serviço não está disponível para os usuários que possuem registros privados em suas empresas. Com o intuito de amenizar essa questão, os autores em [Shu et al. 2017] propuseram o DIVA, o qual permite a análise de vulnerabilidade de imagens Docker. O trabalho acima é semelhante ao trabalho de [Kwon and Lee 2020], o qual propôs um sistema de diagnóstico de vulnerabilidade em imagens Docker, denominado *Docker Image Vulnerability Diagnostic System (DIVDS)*, o qual utiliza o Clair [Clair 2021] em sua solução. Contudo, não há um comparativo quanto a eficácia e eficiência frente a outras alternativas.

Em [Tunde-Onadele et al. 2019], os autores avaliaram a utilização de ferramentas de análise de vulnerabilidade estática e técnicas de análises dinâmicas. Para tanto, foi utilizada a base de vulnerabilidade Vulhub [Vulhub 2021]. Como ferramenta de análise estática, a Clair foi utilizada. Entretanto, não há justificativa do porquê os autores utilizaram somente a Clair ao invés de outra ferramenta.

Comprovando a incidência de vulnerabilidade em imagens Docker, inclusive no próprio Docker Hub, Lin [Lin et al. 2020] analisaram 3.364.529 imagens, cobrindo 98,38% das imagens. Os autores constataram um alto índice de imagens com vulnerabilidades. Diversas imagens analisadas apresentaram algum tipo de problema de segurança que poderia ser facilmente corrigido se fosse identificado e tratado. O estudo demonstra a importância da identificação e tratamento precoce de problemas de segurança.

## 3. Docker

A utilização de contêineres é uma das técnicas mais populares atualmente. Pois, permite o compartilhamento de recursos de *hardware* de uma forma simples e flexível, quando comparado com as técnicas de virtualização tradicionais. Para utilização deste tipo de

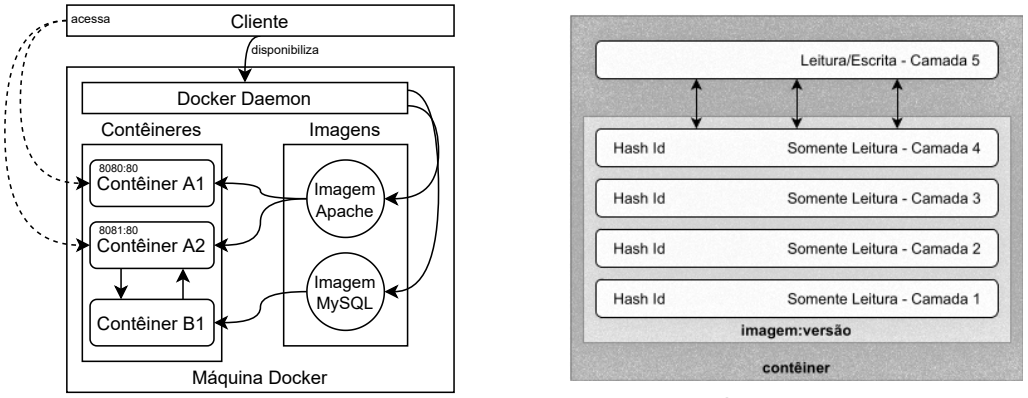


**Figura 1. (a) Funcionamento do Docker, e (b) estrutura das imagens no Docker.**

solução na prática, há a necessidade de utilização de uma tecnologia que implemente esta técnica. O Docker é uma das tecnologias baseadas em contêiner mais utilizadas [Sultan et al. 2019]. A Figura 1a apresenta uma visão simplificada do funcionamento do Docker, onde o cliente solicita para o Docker Daemon a disponibilização de dois servidores (contêineres) Apache a partir de uma imagem que possui todas as configurações necessárias para a disponibilização. Pode-se observar a reutilização da mesma imagem como matriz para dois serviços distintos. Por fim, o cliente realiza o acesso através de requisições para os contêineres A1 e A2 nas portas mapeadas 8080 e 8081, respectivamente.

A distribuição das aplicações Docker é feita através de imagens como ilustra a Figura 1b. Estas imagens são compartilhadas entre diversos desenvolvedores em camadas de maneira hierárquica, este compartilhamento permite o reaproveitamento e simplifica o processo de disponibilização dos contêineres[Shu et al. 2017]. As imagens possuem todas as configurações, bibliotecas e aplicações necessárias para a correta execução dos contêineres. As imagens podem ser armazenadas em servidores públicos, na internet, ou em servidores privados. Entretanto, existe a possibilidade de que as imagens possuam vulnerabilidades de segurança [Brady et al. 2020], como em qualquer solução tecnológica, que podem ocasionar diversos prejuízos para a organização que as utiliza. Para resolver este problema, surge a necessidade de verificar vulnerabilidades nas imagens Docker de maneira automática.

## 4. Ferramentas de Análise Estática de Vulnerabilidade

Como reportado por [Sultan et al. 2019, Prevasio 2020], imagens oficiais possuem vulnerabilidades de segurança e cerca de 4 milhões de imagens disponibilizadas no Docker Hub possuem vulnerabilidades críticas. Ainda, 90% das imagens do Docker Hub possuem vulnerabilidades consideradas de alta severidade [Shu et al. 2017]. Para uma crescente utilização da conteinerização das aplicações e da popularidade da tecnologia Docker, o grande volume de vulnerabilidades encontrado nas imagens é algo bastante alarmante. Pois, segundo o que foi apresentado, somente 10% são seguras para serem utilizadas, o que leva a acreditar que as aplicações que fazem utilização dessas imagens e não trataram explicitamente este problema estão correndo sérios riscos de serem atacadas. Entretanto, para a correção dos problemas surge a necessidade de que as vulnerabilidades presentes

nas imagens sejam identificadas em sua totalidade. Para tanto, a identificação depende de ferramentas de análise de vulnerabilidades que sejam efetivas quanto a detecção de maneira oportuna.

Neste sentido e com o foco no problema de tentar determinar quais as ferramentas são melhores para analisar estaticamente e identificar vulnerabilidades em imagens Docker restritas no ambiente corporativo (*on premise*), devido às restrições corporativas legais ou políticas, foram identificadas algumas ferramentas para serem avaliadas como candidatas. Os critérios de seleção considerados para a escolha foram: ser de código aberto; capaz de identificar as vulnerabilidades classificadas seguindo o *Common Vulnerabilities and Exposures* (CVE) [Mell et al. 2007], que fornecem uma estrutura para quantificar, avaliar e compartilhar vulnerabilidades e problemas de exposições; e que a ferramenta execute integralmente no ambiente interno da empresa que a utilizará. Portanto, seguindo estes critérios, as ferramentas Anchore Grype v0.11.0 [Anchore 2021a], Clair 4 [Clair 2021] e Dagda 0.7.0 [Dagda 2021] foram escolhidas. Por fim, esta não é uma lista exaustiva, mas sim as mais citadas entre os artigos analisados e resultados de buscadores [Bhat 2018, Avi 2021].

Em novembro de 2020, a ferramenta Snyk [Snyk 2021] se tornou a ferramenta oficial da Docker para fazer análises de vulnerabilidades [Armstrong 2020]. A ferramenta foi incorporada aos comandos Docker (*docker scan*) que permite a realização de verificação de maneira integrada. Entretanto, esta solução não é código aberto, mas possui a possibilidade de se utilizar de forma gratuita com limitações de 200 testes de código e 100 contêineres por mês, o qual se passar deste limite deverá ser contratado o plano pago. Destacamos que as análises são realizadas *off premise* na própria nuvem da Snyk. Deste modo, tal ferramenta não se enquadraria nos critérios de seleção definidos para estes estudos. Contudo, por se tratar da ferramenta escolhida como oficial para o Docker, optou-se por se analisar a solução em comparação com as outras escolhidas.

### 4.1. Anchore Grype

O Grype é uma ferramenta de código aberto da Anchore [Anchore 2021b] que realiza análise estática de vulnerabilidades em imagens Docker e contêineres que executa em linha de comando. A ferramenta realiza uma inspeção profunda e varredura nas diversas camadas da imagem que ao final apresenta um relatório com as vulnerabilidades diretamente no terminal que foi executado. O Grype permite que sua execução seja realizada diretamente na máquina ou em contêineres, além da possibilidade de adaptação com ferramentas de integração contínua ou entrega contínua. Desta forma, o usuário consegue rapidamente integrar a ferramenta em seu processo de automação.

Quanto ao relatório disponibilizado, a ferramenta exibe diretamente no terminal que foi executado e o formato de saída pode ser definido pelo próprio usuário. De maneira geral, os dados apresentados são o código CVE da vulnerabilidade, o pacote que foi identificado como problemático, a severidade e a versão do pacote que corrige a vulnerabilidade, caso haja. Deste modo, o usuário consegue verificar quais são as ações a serem tomadas para que a vulnerabilidade seja sanada.

### 4.2. Clair

O Clair é um aplicativo de código aberto para analisar conteúdos de imagens e relatar vulnerabilidades, composto por diversos módulos como a interface HTTP, notificador e

o núcleo. A análise é realizada camada a camada de forma estática e não em tempo de execução [Shu et al. 2017, Tunde-Onadele et al. 2019]. O aplicativo submete para a análise imagens que forem submetidas ao registro local. O Quay foi utilizado em conjunto do Clair, para que as imagens sejam armazenadas e as devidas análises de vulnerabilidade realizadas. Como saída, a ferramenta apresenta o relatório com os resultados obtidos. No relatório, os dados dos pacotes presentes na imagem, a severidade, o código CVE e a versão do pacote afetado são exibidos. Dentre as ferramentas analisadas nos trabalhos correlatos, o Clair é a ferramenta mais citada e a mais utilizada.

A solução Clair para execução da análise de vulnerabilidade é bem diferente da Anchore Grype, onde toda execução é realizada em um único aplicativo. A solução completa é composta de diversos componentes de terceiros, desde servidores de banco de dados a gerenciamento de filas, que trabalham em conjunto e entregam uma solução completa, desde o repositório de imagem (Quay) até o armazenamento do histórico de todas as análises que a imagem foi submetida. Ainda, o relatório da análise não é obtido após a execução do comando de análise, pois a análise é realizada automaticamente nas imagens que são submetidas ao Quay e é gerenciada por uma fila de processamento. Portanto, a obtenção do tempo médio de execução das análises fica prejudicada.

### 4.3. Dagda

O Dagda é uma ferramenta desenvolvida em Python que realiza análises estáticas de vulnerabilidades conhecidas e categorizadas no CVEs, BIDs (*Bugtraq IDs*), RHSAs (*Red Hat Security Advisories*) e RHBAs (*Red Hat Bug Advisories*) [Son 2019]. Ainda, ele utiliza o ClamAV para verificar e identificar vírus e *malwares* nas imagens Docker. Além disso, ele também se integra com Sysdig Falco para monitorar contêineres Docker durante a execução para identificar anomalias. A Figura 2 demonstra a arquitetura da ferramenta e seus componentes. Semelhante ao Clair, o Dagda não responde com um relatório do resultado de imediato, mas sim quando solicitada a análise de uma imagem e o processo é colocado em uma fila, para ser processado posteriormente. Somente ao finalizar a análise, o relatório contendo os pacotes vulneráveis, as versões dos pacotes e a classificação da vulnerabilidade encontrada são fornecidos. O Dagda lista todos os pacotes que a imagem utiliza, independentemente de haver ou não vulnerabilidade.

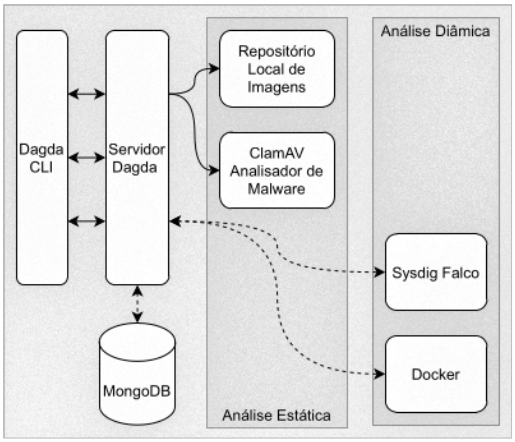


**Figura 2. Componentes do Dagda**

### 4.4. Snyk

O Snyk é uma ferramenta proprietária, diferente das outras soluções apresentadas que realizam análise de vulnerabilidade estática em imagens Docker. A ferramenta pode ser utilizada gratuitamente, entretanto possui algumas limitações na quantidade de execuções por mês. Embora o usuário requisite a análise via linha de comando (terminal), a execução da análise não é realizada localmente, mas na nuvem da Snyk que, após executada, envia a resposta ao requisitante. Deste modo, a resposta é disponibilizada quando solicitada, sem a necessidade de que a análise seja enfileirada.

O resultado que a ferramenta retorna é bastante completo. Ele é composto pela severidade dos problemas encontrados, os pacotes vulneráveis, sua versão e a classificação do CVE. Desta forma, o Snyk permite uma rápida intervenção para correção do problema. No que se trata de correção, no aplicativo da internet da Snyk o usuário pode realizar a correção. Além disso, o Snyk pode ser utilizado com repositórios e ferramentas de entrega e integração contínua. Entretanto, para as soluções que, por questões legais ou outras limitações, não podem ser removidas do ambiente o qual foi concebido esta solução pode não ser adequada.

## 5. Proposta e Metodologia

Este trabalho realiza um estudo comparativo entre as soluções disponíveis em *software* livre quanto ao desempenho e eficácia em identificar problemas e vulnerabilidades em imagens Docker. Os resultados deste trabalho permitem que os desenvolvedores e empresas identifiquem as ferramentas mais apropriadas para a sua necessidade. Como relatado em [Sultan et al. 2019], há uma ausência de estudos que apresentem, de forma clara, as potencialidades e deficiências destas ferramentas. Para atingir este objetivo, foram levantadas algumas ferramentas de análise de vulnerabilidade estática para imagens Docker, conforme os critérios estabelecidos na Seção 4. Um conjunto de imagens Docker contendo vulnerabilidades conhecidas será utilizado como controle, criado a partir do repositório de Dockerfile com vulnerabilidades [Vulhub 2021], que será detalhado na Seção 5.2. A **taxa de identificação de vulnerabilidade** (TIV) será utilizada como métrica para avaliar o desempenho, conforme prescreve a abordagem sistemática definida em [Jain 1991]. Deste modo, todas as ferramentas candidatas serão avaliadas sobre as mesmas circunstâncias e com a lista de vulnerabilidades controlada. Por fim, será criada uma classificação para a avaliação dos resultados.

O TIV é calculado como

$$TIV = \left( \frac{Qtd.\,Vul.\,Encontrada}{Total\,Vul.\,Controle} \right) \times 100, \tag{1}$$

isto é, a quantidade de vulnerabilidades encontrada pela ferramenta analisada é dividida pela quantidade de vulnerabilidades de controle (especificada na Seção 5.2) e multiplicado por 100 que resulta em um percentual podendo ir de 0% a 100%. Assim, é possível realizar uma comparação entre as ferramentas analisadas e com base no TIV aferir a capacidade da ferramenta encontrar vulnerabilidades em imagens Docker. Ou seja, quanto maior o TIV, melhor a ferramenta é na capacidade de encontrar vulnerabilidades.

### 5.1. Ambiente de Testes

Os experimentos foram realizados em um servidor Ubuntu 16.04-LTS provisionado em uma nuvem pública para cada ferramenta analisada. Para cada um destes servidores foi alocado 4 núcleos virtuais, 16 GB de memória RAM e um disco SSD de 128 GB, com 500 IOPS máximo e uma taxa de transferência máxima de 100 Mbps. Para servir de base para toda a infraestrutura de aplicativos e pacotes para que a solução da ferramenta execute em conformidade com as recomendações contidas na sua documentação. Por fim, todas as ferramentas analisadas foram executadas em contêineres Docker de forma que houvesse uma padronização, tanto de máquina, quanto de ambiente e, além do Docker, também foi instalado o Git no servidor, para ser possível obter as soluções armazenadas no Github.

### 5.2. Cenário analisado

Os testes foram executados em 106 imagens previamente preparadas com 109 vulnerabilidades classificadas no CVE. A lista de imagens foi obtida no Vulhub [Vulhub 2021]. Para gerar as imagens foi utilizado o Vulhub, o qual disponibiliza diversos Dockerfiles para o próprio usuário construir imagens com as vulnerabilidades predefinida na documentação [Vulhub 2021, Tunde-Onadele et al. 2019]. Deste modo, foram obtidos todos os Dockerfiles disponíveis no repositório e avaliado um a um para catalogar as imagens e suas respectivas vulnerabilidades. Foram descartados os arquivos Dockerfiles que não foram identificadas as vulnerabilidades classificadas pela CVE. Assim, foi obtido o "gabarito", disponível em [Fialho and Bordim 2021], para aferir a taxa de identificação de vulnerabilidade (TIV).

As imagens foram utilizadas como parâmetros para o experimento. Cada imagem foi submetida para as ferramentas analisarem a presença de vulnerabilidades, tanto as conhecidas com outras que porventura as imagens em questão tiverem. Os resultados da execução e análises serão apresentados na seção subsequente.

## 6. Resultados

Esta seção apresenta os resultados de avaliação utilizando a taxa de identificação de vulnerabilidades (TIV) definida na Eq. 1, seguindo a metodologia apresentada na seção anterior.

De posse dos relatórios obtidos das ferramentas selecionadas, foi realizada uma comparação entre elas. A Tabela 1 apresenta a taxa de identificação de vulnerabilidades (TIV) obtidas por cada ferramenta e a quantidade de vulnerabilidades identificadas (Qtd. Vul. Controle), onde foram observadas a quantidade de vulnerabilidades do controle e a quantidade de vulnerabilidades distintas que a ferramenta encontrou nas imagens analisadas (Qtd. Vul. Total). Assim, das 109 vulnerabilidades submetidas para a análise, o Anchore Grype identificou 25, obtendo um TIV de 23% em relação ao total. Entretanto, apesar da baixa taxa de precisão em relação ao controle ao qual foi submetida, ela registrou outras vulnerabilidades nos pacotes instalados nas imagens.

Foi analisada, também, a quantidade total de vulnerabilidades distintas encontradas por cada ferramenta. Estas informações estão apresentadas na Tabela 1, independentemente de estarem ou não contidas na base de controle. Assim, a ferramenta Anchore Grype encontrou 5.731 vulnerabilidades (CVEs) nas 106 imagens analisadas. A Snyk, a Dagda e a Clair encontraram 2.539, 784 e 270 vulnerabilidades nas imagens analisadas,

**Tabela 1. Quantidade e severidade das vulnerabilidades identificadas**

| Ferramentas | Qtd. Vul. Controle | TIV | Qtd. Vul. Total | Severidade (CVSS) | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | Nenhuma | Baixa | Média | Alta | Crítica |
| Anchore Grype | 25 | 23% | 5.731 | 0,04% | 10,46% | 67,82% | 16,64% | 5,04% |
| Clair | 2 | 2% | 270 | 0,00% | 15,12% | 67,44% | 11,63% | 5,81% |
| Dagda | 0 | 0% | 784 | 0,00% | 12,76% | 60,08% | 21,30% | 5,87% |
| Snyk | 0 | 0% | 2.539 | 0,00% | 7,96% | 75,69% | 14,51% | 1,83% |

respectivamente. Entretanto, não foi analisada a presença de falsos positivos ou falsos negativos em cada uma. Com base no volume de vulnerabilidades encontradas, a ferramenta Anchore Grype apresenta um desempenho bastante superior em relação as demais.

A ferramenta Clair conseguiu identificar apenas 2 vulnerabilidades, a CVE-2017-12794 e a CVE-2020-16846, das 109 vulnerabilidades da lista de controle. Assim, a Clair obteve no TIV 2% em relação ao total de vulnerabilidades do controle. No estudo [Tunde-Onadele et al. 2019], que utilizou o Clair para obter o TIV em conjunto com outras ferramentas, foi reportado um TIV de 10,71%. No referido estudo, foram utilizadas um total 28 de CVEs na base de controle e, destes, somente 3 CVEs foram identificados positivamente. Ao realizar uma comparação do resultado obtido no estudo [Tunde-Onadele et al. 2019] com o presente trabalho, destacamos que 15 dos CVEs utilizadas como controle em [Tunde-Onadele et al. 2019] constam da base de 106 CVEs utilizados aqui. Dos 15 CVEs em comum, 14 obtivemos o mesmo resultado. Entretanto, em um dos CVEs (CVE-2014-6271) obtivemos um resultado diferente, onde neste experimento o CVE não foi identificado pela ferramenta. Assim como a ferramenta Anchore Grype, a Clair identificou diversas outras vulnerabilidades. Portanto, é possível afirmar que a eficácia em relação ao controle é muito baixa, o que confirma o estudo [Tunde-Onadele et al. 2019].

No que tange as ferramentas Dagda e Snyk, esta última é a ferramenta oficial Docker para análise de vulnerabilidades, elas não obtiveram sucesso em identificar as vulnerabilidades da lista de controle. Portanto, ambas foram classificadas com o TIV em 0%, não obtendo eficácia para as CVEs utilizadas neste estudo. Entretanto, apesar de não terem identificado as CVEs de controle, outras vulnerabilidades foram apresentadas nos pacotes utilizados nas camadas da imagem analisadas. Portanto, ambas tiveram o pior desempenho neste estudo ao se considerar somente as vulnerabilidades do controle, o que foi surpresa pelo fato da Snyk ser a ferramenta oficial, esperava-se que tivesse um desempenho superior as demais o que não se provou verdade.

Para todos os CVEs encontrados pelas ferramentas, foi verificado o grau de severidade baseado no *Common Vulnerability Scoring System* (CVSS)[Circl 2021], apresentado na Tabela 1. Na Anchore Grype, das 5.731 vulnerabilidades encontradas, apenas 2 (0,04%) são classificadas como nenhuma severidade; 599 são de baixa severidade, o que corresponde a 10,46% das 5.731; nas de média severidade, foram encontradas 3.887 (67,82%); para as vulnerabilidades de alta severidade se observou um total 954, o que representa 16,64% do total encontrado pela ferramenta; sendo 289 (5,04%) classificadas como críticas. Já para a ferramenta Clair, observa-se que das 270 vulnerabilidades encontradas 41 (15,12%) são de severidade baixa; 182 são classificadas como média, correspondendo a 67,44% do que foi encontrado; 31 (11,63%) como de alta severidade; e 16 (5,81%) classificadas como críticas. Das 25 vulnerabilidades do controle que o An-

chore Grype identificou, 8 são críticas, 10 são altas e 7 são médias. A ferramenta Dagda encontrou em sua grande maioria vulnerabilidades de média severidade, semelhantes as demais, sendo 471 de 784, o que corresponde a 60,08% do total encontrado pela ferramenta, seguido das de altas severidade, 368 (21,30%), as de baixa severidade, 100 (12,76%) e as críticas 46 (5,87%). Na Snyk, que encontrou 2.539 vulnerabilidades no total, 202 (7,96%) foram de baixa severidade; 1.922 (75,69%) de média severidade; 368 (14,51%) de alta severidade; e, 47 (1,83%) de críticas. As 2 vulnerabilidades do controle encontrada pelo Snyk, uma é alta e outra é média. Neste sentido, é possível verificar que a maior parte das vulnerabilidades encontradas são de média severidade e que a proporção é semelhante nas demais severidades com exceção da Snyk, que encontrou 1,83% de CVEs classificados como críticos, no que nas demais a média foi de 5,57% de CVEs críticos. Por fim, considerando as 106 imagens analisadas, todas apresentaram vulnerabilidades críticas identificadas em pelo menos uma das ferramentas. Ao se comparar proporcionalmente o que foi encontrado pelas ferramentas, observa-se que todas obtiveram resultados bem semelhantes, com exceção do Snyk que teve uma concentração dos resultados na severidade média e poucos críticos em relação aos demais, o que pode indicar uma deficiência da ferramenta em identificar algumas vulnerabilidades críticas, o que aumenta o risco de imagens serem utilizadas em ambientes de produção com estas vulnerabilidades.

**Tabela 2. Quantidade de CVEs comuns entre ferramentas**

| Ferramentas | Grype | Clair | Snyk | Dagda |
|---|---|---|---|---|
| **Grype** | 5731 (100%) | 153 (3%) | 2341 (41%) | 413 (7%) |
| **Clair** | 153 (57%) | 270 (100%) | 157 (58%) | 5 (2%) |
| **Snyk** | 2341 (92%) | 157 (6%) | 2539 (100%) | 198 (8%) |
| **Dagda** | 413 (53%) | 5 (1%) | 198 (25%) | 784 (100%) |

Os CVE-2018-7738, CVE-2016-6313, CVE-2017-7526, CVE-2017-9526, CVE-2015-7511 são os CVEs encontrados por todas as ferramentas. Ou seja, do total de vulnerabilidades reportadas pelas ferramentas, apenas 5 CVEs são comuns entre elas. Este fato denota a disparidade entre as ferramentas e sua capacidade de identificar vulnerabilidades.

Com o intuito de melhor avaliar a capacidade de identificação e similaridade entre as ferramentas, avaliamos o percentual de similaridade entre elas. Os resultados de correlação são apresentados na Tabela 2, onde cada linha e coluna denotam a correlação das vulnerabilidades encontradas com as demais ferramentas. Na tabela, a primeira linha e coluna apresentam os resultados para o Grype e o comportamento das demais ferramentas em relação a ela. O mesmo ocorre na diagonal principal da tabela, onde a interseção da linha e coluna denotam 100% para a ferramenta em avaliação. Observando a primeira coluna, verifica-se que das 2.539 vulnerabilidades encontradas pelo Snyk, 92% (i.e, 2.341) foram também encontradas pelo Grype. Em relação ao Clair, das 270 vulnerabilidades reportadas, 57%, ou seja, 153 vulnerabilidades são comuns ao Grype. Ao analisar as demais colunas, verifica-se que o Grype e o Snyk são as ferramentas que possuem maior quantitativo de vulnerabilidades em comum com as demais ferramentas e entre elas. De forma semelhante, cada linha apresenta os resultados em comparação com as demais. Por exemplo, ao comparar a coluna do Snyk com as demais ferramentas, verifica-se que das 2.539 vulnerabilidades reportadas, 2.341 foram também identificadas pelo Grype. Ou seja, o Snyk encontrou 41% das 5.731 vulnerabilidades reportadas pelo Grype. Com relação ao Dadga, o Snyk encontrou 198 das 784 reportadas, um percentual de correlação entre as ferramentas de 25%. Com base nos resultados da Tabela 2, e desconsiderando falsos po-

sitivos e negativos, pode-se afirmar que a Grype e Snyk apresentam melhores resultados em comparação com as demais ferramentas. Dada a grande similaridade com a Snyk e sua superioridade em relação ao quantitativo de vulnerabilidades encontradas, o Grype apresenta-se como a melhor alternativa dentre as ferramentas avaliadas.

### 6.1. Vulnerabilidades nos pacotes

Como mencionado na Seção 4, as ferramentas avaliam as vulnerabilidades com base nos pacotes instalados e utilizados nos sistemas operacionais disponíveis nas imagens. Estes pacotes servem de base para execução das soluções nos contêineres. Neste estudo, identificamos $\approx$ 1000 pacotes com algum tipo de problemas de vulnerabilidade nas diversas imagens analisadas e $\approx$ 50% dos pacotes possuem vulnerabilidades classificadas como críticas. Na Tabela 3, estão listados os pacotes que mais afetaram as imagens analisadas. Todos os pacotes listados possuem vulnerabilidades e estão presentes em mais de 80% das imagens. Ainda, é possível verificar que a maioria destes pacotes pertence às camadas base das imagens, que normalmente estão presentes e são essenciais para execução básica do sistema operacional. Consequentemente, a correção e atualização de alguns destes pacotes poderiam resolver o problema em um quantitativo expressivo de imagens. Estas informações denotam a necessidade e importância dos cuidados com as camadas base das imagens, pois as vulnerabilidades são propagadas para as camadas superiores.

**Tabela 3. Pacotes que mais impactam as imagens**

| Pacote | Img. Afetadas | Pacote | Img. Afetadas |
|---|---|---|---|
| bash | 100% | coreutils | 100% |
| tar | 100% | apt | 100% |
| libgcrypt20 | 99% | libtasn1-6 | 97% |
| util-linux | 96% | openssl | 95% |
| curl | 94% | perl-base | 93% |
| bzip2 | 93% | perl | 92% |
| e2fsprogs | 89% | gpgv | 87% |
| libc-bin | 87% | libc6 | 87% |
| libudev1 | 87% | login | 84% |
| passwd | 84% | libpcre3 | 83% |
| libstdc++6 | 83% | libsystemd0 | 83% |
| systemd | 82% | libgcc1 | 79% |

### 6.2. Avaliação de usabilidade, disponibilização e documentação

Outros pontos que foram avaliado indiretamente, foram a dificuldade de utilização da ferramenta (usabilidade) e a complexidade e dificuldade para disponibilizar e instalar a ferramenta (disponibilização). Ainda, foram avaliados os aspectos da documentação técnica encontrada durante a realização do experimente e na avaliação da qualidade dos relatórios. Para tanto, a ferramenta foi classificada como fácil, médio ou difícil, considerando a quantidade de passos e de comandos necessários para a disponibilização da solução completa e da documentação disponível nos repositórios para a usabilidade e disponibilização. Classificamos como ruim, boa ou ótima no aspecto da qualidade e disponibilidade da documentação técnica encontrada. Os resultados são apresentados na Tabela 4.

Dentre os classificados como fáceis se destacam o Anchore Grype e o Snyk. O Anchore Grype se destaca pelo fato de utilizar a imagem já pronta e chamar a linha de comando ou instalar e utilizar os comandos no terminal e de ter uma documentação suficiente para um uso básico no próprio repositório. O Snyk possui uma vasta documentação

**Tabela 4. Complexidade de Uso da Ferramenta**

| Ferramenta | Usabilidade | Disponibilização | Documentação |
|---|---|---|---|
| Anchore Grype | Fácil | Fácil | Boa |
| Clair | Fácil | Médio | Boa |
| Dagda | Médio | Difícil | Ruim |
| Snyk | Fácil | Fácil | Ótima |

de uso e está disponível em conjunto com o Docker. Entretanto, há a necessidade do cadastro no site da Snyk e gerar um *token*, pois, caso contrário, não poderá fazer mais do que 10 execuções.

A ferramenta Clair foi classificada com o nível de usabilidade fácil, disponibilização média e documentação boa. O Clair exige a execução de toda uma infraestrutura acessória para o pleno funcionamento. Entretanto, há uma vasta documentação para auxiliar a execução e a instalação. Portanto, esta ferramenta ocupa o terceiro lugar entre as ferramentas nos quesitos avaliados. Já a Dagda, foi classificada como a mais difícil de ser utilizada. Apesar de ter menos componentes de infraestrutura que compõem a solução em comparação com a Clair, conforme ilustra a Figura 2, a sua documentação é deficitária e requer diversos comandos de permissão para conseguir executar a análise. Os logs apresentados pela ferramenta para identificação de vulnerabilidades são limitados, comprometendo sua usabilidade. Diante do exposto, a Dadga foi classificada como média, no quesito usabilidade, difícil no quesito disponibilização e ruim no quesito documentação.

Por fim, as ferramentas analisadas tiveram um baixo desempenho na capacidade de identificar as vulnerabilidades listadas nos experimentos. O Snyk e Dagda não conseguiram identificar nenhuma das vulnerabilidades da lista de controle e tiveram o pior desempenho quanto a usabilidade. Embora o Anchore Grype tenha apresentado os melhores resultados, somente 23% das vulnerabilidades da base de controle foram identificadas. Porém, o Grype obteve o melhor desempenho quanto a usabilidade. Quando consideramos a quantidade de vulnerabilidade identificadas, a Anchore Grype se destaca dentre as ferramentas avaliadas, seguida da Snyk, da Dagda e da Clair. Portanto, este estudo demonstra que as ferramentas analisadas são bastante incipientes para identificar o problema de vulnerabilidade em imagens, o que sugere uma boa prática realizar a combinação com outras ferramentas de modo a se obter um melhor resultado, assim como nos estudos realizados em [Brady et al. 2020, Shu et al. 2017].

## 7. Conclusão

O presente trabalho realiza um estudo comparativo entre as principais soluções de análise de vulnerabilidade em imagens considerando o desempenho na capacidade de identificar vulnerabilidades e na sua usabilidade. A metodologia utilizada neste trabalho permite avaliar a qualidade das respostas das ferramentas Anchore Grype, Clair, Dagda e Snyk, as quais foram escolhidas por serem de código aberto e passíveis de serem utilizadas *on premise*, com exceção da Snyk, ferramenta oficial Docker, a qual foi utilizada para comparação entre as ferramentas. As soluções foram disponibilizadas em um servidor cada uma e executadas em contêineres. Neste trabalho utilizamos o conjunto de Dockerfiles disponíveis no Vulhub [Vulhub 2021], o qual lista as vulnerabilidades identificadas nestas imagens. Este conjunto de imagens foi utilizado para controle na avaliação das

soluções. Assim, cada imagem foi submetida para análise e foram extraídos os relatórios para serem avaliados posteriormente. Dos resultados obtidos o Anchore Grype foi o que obteve melhor resultado nas métricas definidas neste trabalho: taxa de identificação de vulnerabilidade (TIV) e usabilidade. No quesito usabilidade e disponibilização a ferramenta foi classificada como fácil uso e obteve um valor de TIV de 23% com relação ao conjunto de controle. Ainda, dentre as ferramentas avaliadas foi a que mais identificou vulnerabilidades em imagens, totalizando 5.731.

Como trabalhos futuros, pretende-se expandir a lista de ferramentas dado que agora foram consideradas as mais citadas em trabalhos correlatos. Ainda, por questão de limitações de tempo, não foram analisados os resultados das outras vulnerabilidades encontradas pelas ferramentas e nem considerados os falsos positivos e falsos negativos.

## Referências


Anchore (2021a). Anchore grype. Avaliable: `https://github.com/anchore/grype`. Last accessed 18 May 2021.

Anchore (2021b). Anchore open-source. Avaliable: `https://anchore.com/opensource/`. Last accessed 19 May 2021.

Armstrong, J. (2020). Docker desktop with snyk and new docker vulnerability cheat sheet available. Last accessed 18 May 2021.

Avi (2021). 10 container security scanners to find vulnerabilities. Avaliable: `https://geekflare.com/container-security-scanners/`. Last accessed 17 May 2021.

Aws (2021). Aws image scanning. Avaliable: `https://docs.aws.amazon.com/AmazonECR/latest/userguide/image-scanning.html`. Last accessed 21 May 2021.

Bhat, S. (2018). 5 open source tools for container security. Avaliable: `https://opensource.com/article/18/8/tools-container-security`. Last accessed 17 May 2021.

Brady, K., Moon, S., Nguyen, T., and Coffman, J. (2020). Docker container security in cloud computing. In *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0975–0980.

Circl (2021). cve-search - common vulnerabilities and exposure web interface and api. Avaliable: `https://cve.circl.lu/`. Last accessed 20 May 2021.

Clair (2021). Clair. Avaliable: `https://quay.github.io/clair/`. Last accessed 18 May 2021.

Dagda (2021). Dagda. Avaliable: `https://github.com/eliasgranderubio/dagda`. Last accessed 18 May 2021.

Docker (2021a). Docker. Avaliable: `https://www.docker.com/`. Last accessed 17 May 2021.

Docker (2021b). Docker hub. Avaliable: `https://hub.docker.com/`. Last accessed 17 May 2021.

Fialho, Y. and Bordim, J. (2021). Imagens e vulnerabilidades de controle. Avaliable: https://github.com/yurifialho/vulhub_controle/blob/main/Vulnerabilidade_Controle.pdf. Last accessd 16 Jun 2021.

Google (2021). Google automatic vulnerability scanning. Avaliable: https://cloud.google.com/container-analysis/docs/vulnerability-scanning. Last accessed 21 May 2021.

IBM (2021). Ibm's vulnerability advisor. Avaliable: https://cloud.ibm.com/docs/Registry?topic=va-va_index. Last accessed 21 May 2021.

Jain, R. (1991). *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley.

Kwon, S. and Lee, J.-H. (2020). Divds: Docker image vulnerability diagnostic system. *IEEE Access*, 8:42666–42673.

Lin, C., Nadi, S., and Khazaei, H. (2020). A large-scale data set and an empirical study of docker images hosted on docker hub. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 371–381.

Mell, P., Scarfone, K., and Romanosky, S. (2007). The common vulnerabi-lity scoring system (CVSS) and its applicability to federal agency sys-tems, NIST IR 7435. *USA: Department of Commerce*.

Prevasio (2020). Industry's first dynamic analysis of 4 millionpublicly available docker hub container images. Last accessed 18 May 2021.

Shu, R., Gu, X., and Enck, W. (2017). A study of security vulnerabilities on docker hub. In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, CODASPY '17, page 269–280, New York, NY, USA. Association for Computing Machinery.

Snyk (2021). Snyk. Avaliable: https://snyk.io/. Last accessed 19 May 2021.

Son, D. (2019). dagda: perform static analysis of known vulnerabilities, trojans, viruses, malware & other malicious threats. Last accessed 18 May 2021.

Sultan, S., Ahmad, I., and Dimitriou, T. (2019). Container security: Issues, challenges, and the road ahead. *IEEE Access*, 7:52976–52996.

Tunde-Onadele, O., He, J., Dai, T., and Gu, X. (2019). A study on container vulnerability exploit detection. In *2019 IEEE International Conference on Cloud Engineering (IC2E)*, pages 121–127.

Twistlock (2021). Twistlock - prima cloud. Avaliable: https://www.paloaltonetworks.com/prisma/cloud. Last accessed 21 May 2021.

Vulhub (2021). Vulhub: Docker-compose file for vulnerability environment. Avaliable: http://vulhub.org. Last accessed 18 May 2021.

Zheng, Y., Dong, W., and Zhao, J. (2021). Zerodvs: Trace-ability and security detection of container image based on inheritance graph. In *2021 IEEE 5th International Conference on Cryptography, Security and Privacy (CSP)*, pages 186–192.