INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS - *CAMPUS* OURO BRANCO
SISTEMAS DE INFORMAÇÃO

Lucas Da Costa Miranda

SEGURANÇA EM DOCKER MULTIUSUÁRIO: AVALIAÇÃO DE RISCOS DE PRIVILEGE ESCALATION E DATA TAMPERING

Ouro Branco - MG
2026

LUCAS DA COSTA MIRANDA

SEGURANÇA EM DOCKER MULTIUSUÁRIO: AVALIAÇÃO DE RISCOS DE PRIVILEGE ESCALATION E DATA TAMPERING

Trabalho de conclusão de curso apresentado ao Curso de Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais - *Campus* Ouro Branco para a obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Charles Tim Batista Garrocho

Ouro Branco - MG
2026

M672s Miranda, Lucas Da Costa.

Segurança em docker multiusuário: avaliação de riscos de privelege escalation e data tempering. / Lucas da Costa Miranda. – 2026.

14f.

Orientador: Charles Tim Batista Garrocho.

Trabalho de Conclusão de Curso (Sistemas de Informação) – Instituto Federal de Minas Gerais. Campus Ouro Branco, 2026.

1. Docker. 2. Segurança de contentores. 3. Escalonamento de privilégios. 4. Violação de integridade. 5. IntegrityGuard. I. Garrocho, Charles Tim Batista. II. Instituto Federal de Minas Gerais. Campus Ouro Branco. III. Título.

CDU: 004.056

Catalogação: Márcia Margarida Vilaça - CRB-6/2235
Biblioteca do Instituto Federal de Minas Gerais, *Campus* Ouro Branco

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS
CAMPUS OURO BRANCO
Av. Afonso Sardinha, nº 90, Bairro Pioneiros, CEP: 36.420-000, Ouro Branco - Minas Gerais
(31) 3742-2149 – gabinete.ourobranco@ifmg.edu.br

ANEXO II – ATA DE CONCLUSÃO DE TCC

Aos *19* dias do mês de *janeiro* de *2026*, às *21:00* horas, *Lucas Da Costa Miranda*, aluno(a) regularmente matriculado no Curso de Sistemas de Informação do Instituto Federal de Minas Gerais, campus Ouro Branco, matrícula *0071067*, concluiu o seu Trabalho de Conclusão de Curso por meio de:

( ) Publicação do artigo intitulado _____
na revista/conferência _____,
cujo comprovante de aceitação será anexado a esta ata, recebendo a nota _____ pelo trabalho. Eu, na qualidade de orientador do aluno, lavrei a presente ata atestando a conclusão do trabalho, a qual será assinada por mim e pelo aluno.

_____
Professor Orientador

_____
Aluno

(*X*) Defesa em sessão pública realizada às 21:00 horas, na sala auditório do Instituto Federal de Minas Gerais, campus Ouro Branco, na presença da banca examinadora composta pelos docentes:

1 - *Carlos Eduardo Paulino Silva*
2 - *Daniela Costa Terra*
3 - *Ederson Naves Fernandes Gonçalves Júnior*
4 - *Marcio Assis Miranda*

do artigo intitulado *Segurança em Docker Multiusuário: Avaliação de Riscos de Privilege Escalation e Data Tampering*

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS
***CAMPUS* OURO BRANCO**
Av. Afonso Sardinha, nº 90, Bairro Pioneiros, CEP: 36.420-000, Ouro Branco - Minas Gerais

(31) 3742-2149 – gabinete.ourobranco@ifmg.edu.br

A banca examinadora, após reunião em sessão reservada, deliberou pela *aprovação* do referido trabalho, atribuindo a nota *94,8*. Eu, na qualidade de presidente da banca examinadora, lavrei a presente ata que será assinada por mim, pelos demais examinadores e pelo aluno.
Observações pertinentes à defesa:

NOME E ASSINATURA DOS COMPONENTES DA BANCA E DO ORIENTADO

Documento assinado digitalmente
gov.br **CHARLES TIM BATISTA GARROCHO**
Data: 23/01/2026 19:21:43-0300
Verifique em https://validar.iti.gov.br

Professor Orientador: *Charles Tim Batista Garrocho*

Documento assinado digitalmente
gov.br **CARLOS EDUARDO PAULINO SILVA**
Data: 23/01/2026 17:25:24-0300
Verifique em https://validar.iti.gov.br

Examinador 1: *Carlos Eduardo Paulino Silva*

Documento assinado digitalmente
gov.br **DANIELA COSTA TERRA**
Data: 25/01/2026 18:43:21-0300
Verifique em https://validar.iti.gov.br

Examinador 2: *Daniela Costa Terra*

Documento assinado digitalmente
gov.br **EDERSON NAVES FERNANDES GONCALVES JUN**
Data: 26/01/2026 11:07:54-0300
Verifique em https://validar.iti.gov.br

Examinador 3: *Ederson Naves Fernandes Gonçalves Júnior*

Documento assinado digitalmente
gov.br **MARCIO ASSIS MIRANDA**
Data: 26/01/2026 21:03:25-0300
Verifique em https://validar.iti.gov.br

Examinador 4: *Marcio Assis Miranda*

Documento assinado digitalmente
gov.br **LUCAS DA COSTA MIRANDA**
Data: 23/01/2026 15:54:25-0300
Verifique em https://validar.iti.gov.br

Aluno(a): *Lucas Da Costa Miranda*

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS
CAMPUS OURO BRANCO
Av. Afonso Sardinha, nº 90, Bairro Pioneiros, CEP: 36.420-000, Ouro Branco - Minas Gerais
(31) 3742-2149 – gabinete.ourobranco@ifmg.edu.br

DECLARAÇÃO ANTI-PLÁGIO

Eu, *Lucas Da Costa Miranda*, estudante do curso Bacharelado em Sistemas de Informação do IFMG – Campus Ouro Branco, declaro, para os devidos fins e efeitos, e para fazer prova junto ao IFMG – Campus Ouro Branco, que, **sob as penalidades previstas no art. 299 do Código Penal Brasileiro**, que é de minha criação o Trabalho de Conclusão de Curso que ora apresento.

Art. 299 do Código Penal Brasileiro, que dispõe sobre o crime de *Falsidade Ideológica*:

"Omitir, em documento público ou particular, declaração que dele devia constar, ou nele inserir ou fazer inserir declaração falsa ou diversa da que devia estar escrita, com o fim de prejudicar direito, criar obrigação ou alterar verdade sobre fato juridicamente relevante:

Pena – reclusão, de 1 (um) a 5 (cinco) anos, e multa, se o documento é público, e reclusão de 1 (um) a 3 (três) anos, e multa, se o documento é particular.

Parágrafo único. Se o agente é funcionário público, e comete o crime prevalecendo-se do cargo, ou se a falsificação ou alteração é de assentamento de registro civil, aumenta-se a pena de sexta parte".

Este crime engloba plágio e compra fraudulenta de documentos científicos. Por ser verdade, e por ter ciência do referido artigo, firmo a presente declaração.

Ouro Branco, 23 de janeiro de 2026

gov.br Documento assinado digitalmente
LUCAS DA COSTA MIRANDA
Data: 23/01/2026 15:53:39-0300
Verifique em https://validar.iti.gov.br

Assinatura do aluno: _____

Segurança em Docker Multiusuário: Avaliação de Riscos de Privilege Escalation e Data Tampering

Lucas Da Costa Miranda[1], **Charles Tim Batista Garrocho**[1]

[1] Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais (IFMG)
Ouro Branco, Minas Gerais, Brasil

lucasdacostamiranda812@gmail.com, charles.garrocho@ifmg.edu.br

Resumo. *O uso do Docker em ambientes multiusuário cresceu significativamente devido à sua eficiência e portabilidade. Contudo, configurações inseguras, como a permissão de acesso ao grupo* `docker`*, criam brechas críticas para ataques de Escalonamento de Privilégios e Violação de Integridade (Data Tampering). Este trabalho analisa esses riscos por meio de um estudo de caso no IFMG, onde foi identificado o uso inadequado do* `docker.sock`*. Para quantificar essa exposição, propõe-se o Índice de Exposição à Violação de Integridade (IEVI), baseado no framework DREAD, além do desenvolvimento da ferramenta IntegrityGuard para diagnóstico automatizado. Os resultados revelam que a adoção do modo Docker Rootless reduz drasticamente a superfície de ataque, diminuindo o IEVI de 100 para 32 e garantindo maior resiliência em infraestruturas compartilhadas sob os princípios de Zero Trust.*

Abstract. *The adoption of Docker in multi-user environments has increased significantly due to its efficiency and portability. However, insecure configurations, such as granting access to the* `docker` *group, create critical gaps for Privilege Escalation and Data Tampering attacks. This work analyzes these risks through a case study at IFMG, where inappropriate use of* `docker.sock` *was identified. To quantify this exposure, we propose the Integrity Violation Exposure Index (IEVI), based on the DREAD framework, alongside the development of the IntegrityGuard tool for automated diagnosis. Results reveal that adopting Docker Rootless mode drastically reduces the attack surface, lowering the IEVI from 100 to 32 and ensuring greater resilience in shared infrastructures under Zero Trust principles.*

1. Introdução

A conteinerização de aplicações, notadamente através do *Docker*, revolucionou o desenvolvimento e a implantação de software, oferecendo vantagens significativas sobre máquinas virtuais tradicionais, como maior eficiência no uso de recursos e portabilidade (SULTAN; AHMAD; DIMITRIOU, 2019). A ampla adoção do *Docker* em ambientes de desenvolvimento e produção é uma tendência consolidada; contudo, sua utilização em hosts multiusuário, como laboratórios computacionais, centros de pesquisa e máquinas compartilhadas em equipes corporativas, impõe desafios de segurança específicos. Isso ocorre porque *containers* compartilham o mesmo *kernel* do sistema hospedeiro, ampliando a superfície de ataque e expondo o sistema a diferentes vetores de exploração (WONG et al., 2023).

A motivação para este estudo surge de uma análise prática realizada nos laboratórios do Instituto Federal de Minas Gerais Campi Ouro Branco (IFMG), onde foi identificada uma vulnerabilidade sistêmica em mais de 120 máquinas de uso compartilhado por alunos e professores. Essas estações de trabalho apresentavam uma falha crítica de governança: a inclusão de usuários comuns no grupo secundário `docker`. Essa configuração permite o acesso direto ao *socket* `/var/run/docker.sock`, o que equivale, na prática, à concessão de privilégios de *root* ao usuário local(RAJYASHREE et al., 2024). Em hosts compartilhados, a exploração dessa configuração insegura pode levar a duas classes de impacto diretamente relacionadas ao *framework* STRIDE(WONG et al., 2023).

Um cenário prático que ilustra ambas as ameaças é a montagem do sistema de arquivos do *host* dentro de um *container* e a modificação do arquivo `/etc/passwd`. Por exemplo:

Listing 1. Comando para criação de utilizador administrativo via vulnerabilidade *docker*.

```
docker run -v /etc:/mnt/host/etc -it --rm alpine \
  sh -c "echo 'hacker:x:0:0:Hacker:/root:/bin/sh' >> /mnt/host/etc/
     passwd"
```

Esse simples comando demonstra que, além de corromper dados (*Data Tampering*), um atacante pode criar um usuário com *User ID* (UID) 0, obtendo assim controle administrativo do *host*. O tempo de exploração *Time-to-Exploit* (TTE) para essa classe de ataque é baixo (segundos), o que a torna de alto risco em ambientes multiusuário.

A problemática central reside no fato de que, embora a conteinerização ofereça isolamento lógico, ela não garante o isolamento físico robusto característico das máquinas virtuais, uma vez que todos os containers compartilham o mesmo *kernel* do sistema hospedeiro (SULTAN; AHMAD; DIMITRIOU, 2019; WONG et al., 2023). Em cenários de uso compartilhado, como laboratórios acadêmicos e centros de pesquisa, a segurança é frequentemente negligenciada em favor da conveniência operacional. A inclusão de usuários no grupo `docker` para simplificar a execução de tarefas acadêmicas acaba por anular as barreiras de proteção do sistema operacional, transformando uma ferramenta de produtividade em um vetor crítico de ataque. Essa configuração insegura permite que um usuário comum interaja diretamente com o *socket* do *Docker*, o que equivale, na prática, à concessão de privilégios de *root* ao usuário local (RAJYASHREE et al., 2024). A gravidade dessa falha é acentuada pelo baixo tempo de exploração (*Time-to-Exploit*), permitindo que um atacante comprometa a integridade de arquivos sensíveis do *host* em poucos segundos, justificando a necessidade urgente de ferramentas de diagnóstico automatizado e métricas de risco quantitativas.

Diante desse cenário, o objetivo deste trabalho é analisar, quantificar e mitigar os riscos associados tanto à Violação de Integridade quanto à Escalada de Privilégios decorrentes do acesso indevido ao `docker.sock`. As principais contribuições deste artigo são:

1. A formulação do Índice de Exposição à Violação de Integridade (IEVI), uma métrica quantitativa derivada do DREAD, estendida para refletir impactos de tampering e de escalada de privilégio;

2. O desenvolvimento da ferramenta *IntegrityGuard*, que automatiza a detecção da exposição, executa testes controlados e recomenda contramedidas para reduzir o IEVI;
3. A validação empírica dos riscos e das mitigações propostas, demonstrando a eficácia do modo *Docker Rootless* na redução da superfície de ataque em ambientes reais.

O restante deste trabalho está organizado da seguinte forma: a Seção 2 aborda o referencial teórico sobre gestão de privilégios e a plataforma *Docker*, além de discutir os trabalhos relacionados. A Seção 3 descreve a metodologia experimental, os procedimentos de teste e a fundamentação da métrica IEVI. A Seção 4 detalha a arquitetura e funcionalidade da ferramenta *IntegrityGuard*. Na Seção 5, são apresentados e discutidos os resultados quantitativos e a convergência com o modelo *Zero Trust*. As limitações encontradas e as propostas de trabalhos futuros são exploradas nas Seções 6 e 7, respectivamente. Finalmente, a Seção 8 apresenta as conclusões deste estudo.

2. Referencial Teórico

2.1. Gestão de Privilégios e o Usuário Root no Linux

O modelo de segurança do sistema operacional *Linux* é fundamentado no controle de acesso discricionário, onde cada processo e arquivo é associado a um proprietário e a um grupo. No topo dessa hierarquia reside o usuário *root*, também conhecido como *superuser*, que detém controle absoluto sobre o sistema. Identificado tecnicamente pelo *User ID* (*UID*) 0, este usuário possui permissões para modificar qualquer arquivo, interromper processos e alterar configurações críticas do *kernel* (RAJYASHREE et al., 2024).

A arquitetura tradicional do *Docker* depende fortemente desses privilégios elevados, uma vez que o *Docker daemon* (*dockerd*) geralmente é executado com permissões de *root* para gerenciar recursos de baixo nível, como *namespaces* e *control groups* (*cgroups*) (MERKEL et al., 2014). O compartilhamento do *kernel* entre o *host* e os *containers* significa que a barreira de isolamento é lógica, e não física como em máquinas virtuais (SULTAN; AHMAD; DIMITRIOU, 2019; WONG et al., 2023), como exemplificado na Figura 1.

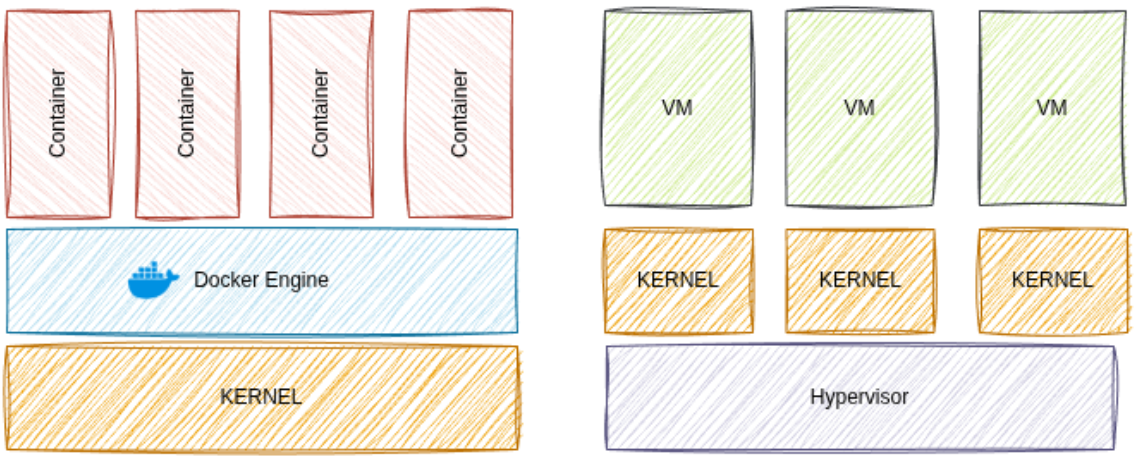

Figura 1. Docker vs VM

Em ambientes multiusuário, a concessão inadvertida de acesso ao *socket* do *Docker* permite que um usuário comum instrua o *daemon* a realizar tarefas em seu nome. Como o *daemon* opera com o *UID* 0, qualquer operação solicitada é executada com a autoridade máxima do sistema. Esse cenário é a base para ataques de *Privilege Escalation*, onde um ator mal-intencionado utiliza o *runtime* de *containers* para contornar as restrições impostas a usuários convencionais e obter controle total sobre a infraestrutura hospedeira (RAJYASHREE et al., 2024; WONG et al., 2023).

2.2. Plataforma e Arquitetura Docker

O *Docker* é uma plataforma aberta voltada para o desenvolvimento, envio e execução de aplicações, permitindo a separação entre o software e a infraestrutura subjacente. Ele utiliza uma arquitetura cliente-servidor, na qual o cliente *Docker* (`docker`) interage com o *Docker daemon* (`dockerd`), responsável por gerenciar objetos como imagens, *containers*, redes e volumes. A comunicação entre esses componentes ocorre por meio de uma *API REST*, podendo ser estabelecida via *sockets* UNIX ou interfaces de rede (DOCKER, 2025b).

Diferente das máquinas virtuais tradicionais, os *containers* são instâncias executáveis de imagens que operam de forma leve e isolada, compartilhando o mesmo *kernel* do sistema hospedeiro (WONG et al., 2023). Essa eficiência é alcançada através de tecnologias do *kernel* Linux, como `namespaces`, que criam espaços de trabalho isolados, e `control groups` (cgroups), que gerenciam a alocação de recursos de hardware (DOCKER, 2025b).

2.2.1. *Docker Rootless* e o Princípio do Privilégio Mínimo

O modo *rootless* surge como uma evolução crítica na arquitetura da plataforma, permitindo que tanto o *Docker daemon* quanto os *containers* sejam executados sem a necessidade de privilégios de *root* no sistema hospedeiro. Enquanto a arquitetura tradicional depende de um *daemon* privilegiado que opera com *User ID* (*UID*) 0, o modo *rootless* utiliza a tecnologia de *User Namespaces* do *kernel Linux* para mapear um usuário comum para o *root* interno do *container*, mantendo-o sem privilégios no *host* (DOCKER, 2025a; SULTAN; AHMAD; DIMITRIOU, 2019).

A importância desta abordagem é fundamentada na redução drástica da superfície de ataque. Ao eliminar a necessidade de inclusão de usuários no grupo *docker* e o acesso direto ao *socket* privilegiado, o sistema mitiga a principal vulnerabilidade explorada para *Privilege Escalation* (RAJYASHREE et al., 2024). Caso ocorra uma falha de segurança ou um *container escape*, o atacante fica restrito às permissões do usuário comum que iniciou o processo, impedindo o comprometimento total da infraestrutura e protegendo a integridade dos arquivos críticos do sistema operacional (VS; SETHURAMAN; KHAN, 2023; WONG et al., 2023). Em ambientes de uso compartilhado, como laboratórios acadêmicos, essa configuração é essencial para equilibrar a flexibilidade do desenvolvedor com a governança de segurança necessária (DOCKER, 2025a).

2.3. Modelagem de Ameaças: STRIDE e DREAD

A modelagem de ameaças é um processo estruturado essencial para identificar e mitigar riscos de segurança desde as fases iniciais do ciclo de vida de um sistema (SHOSTACK,

2014). No ecossistema de containers, essa prática permite mapear superfícies de ataque que abrangem desde o repositório de código até o runtime de execução no host (WONG et al., 2023). Este trabalho adota as metodologias STRIDE e DREAD para a categorização e priorização dos riscos identificados.

2.3.1. Framework STRIDE

O modelo STRIDE, desenvolvido pela Microsoft, é amplamente reconhecido por sua maturidade e aplicabilidade em sistemas distribuídos. Ele classifica as ameaças em seis categorias distintas, cada uma associada à violação de uma propriedade de segurança fundamental (WONG et al., 2023):

1. *Spoofing* (Falsificação): Ocorre quando um atacante assume a identidade de um usuário ou serviço legítimo, violando a propriedade de autenticidade. Em ambientes Docker, isso pode incluir a falsificação de credenciais de acesso ao registro de imagens.
2. *Tampering* (Adulteração): Refere-se à modificação não autorizada de dados ou processos, comprometendo a integridade. A injeção de comandos maliciosos em um `Dockerfile` é um exemplo crítico nesta categoria.
3. *Repudiation* (Repúdio): Envolve a capacidade de um ator negar a realização de uma ação devido à falta de evidências auditáveis. O comprometimento de logs do daemon Docker pode facilitar ataques desta natureza.
4. *Information Disclosure* (Vazamento de Informação): Consiste no acesso não autorizado a dados sensíveis, violando a confidencialidade. A exposição do `docker.sock` pode permitir que um usuário comum visualize informações privilegiadas do host.
5. *Denial of Service* (Negação de Serviço): Visa tornar um serviço ou recurso indisponível, afetando a disponibilidade. O consumo excessivo de CPU ou memória por um container mal configurado exemplifica esta ameaça.
6. *Elevation of Privilege* (Elevação de Privilégio): Ocorre quando um usuário obtém permissões superiores às autorizadas originalmente. Esta é uma das ameaças mais críticas em hosts multiusuário, frequentemente explorada via sockets mal configurados.

2.3.2. Métrica DREAD

Enquanto o STRIDE foca na identificação qualitativa, o modelo DREAD fornece uma abordagem quantitativa para a priorização de riscos. Ele permite calcular um índice de risco (de 1 a 10) com base na média de cinco critérios:

1. *Damage* (Dano): Avalia a extensão do impacto negativo caso a ameaça seja concretizada.
2. *Reproducibility* (Reprodutibilidade): Mede a facilidade com que o ataque pode ser replicado com sucesso.
3. *Exploitability* (Explorabilidade): Analisa o esforço e o conhecimento técnico necessários para executar o ataque.

4. *Affected Users* (Usuários Afetados): Estima a proporção de usuários ou sistemas impactados pela ocorrência da ameaça.
5. *Discoverability* (Descoberta): Avalia o quão fácil é para um atacante identificar a vulnerabilidade no sistema.

A integração desses dois modelos permite que administradores de ambientes acadêmicos e corporativos não apenas identifiquem onde estão as falhas no Docker, mas também quais devem ser mitigadas com maior urgência para garantir a resiliência operacional (VS; SETHURAMAN; KHAN, 2023; SULTAN; AHMAD; DIMITRIOU, 2019).

2.4. Trabalhos Relacionados

A literatura acadêmica recente apresenta diversas abordagens sobre a segurança de *containers*, focando em modelagem de ameaças e análise de vulnerabilidades em *runtimes* (SULTAN; AHMAD; DIMITRIOU, 2019; WONG et al., 2023). Esta subseção compara os estudos mais relevantes com a abordagem proposta neste trabalho.

O estudo desenvolvido por Rajyashree et al. (2024) realiza uma investigação empírica focada no *Docker socket* como vetor para *Privilege Escalation*. Os autores demonstram técnicas de exploração e sugerem defesas baseadas em *User Namespace Mapping*, porém não exploram cenários de *Data Tampering* em larga escala ou a automação do diagnóstico em ambientes com alta rotatividade de usuários.

Em uma perspectiva de modelagem sistêmica, Wong et al. (2023) utiliza o *framework STRIDE* para analisar a superfície de ataque de ecossistemas de *containers*. Embora forneçam uma taxonomia abrangente de ataques e mitigações, o estudo não propõe uma ferramenta de detecção ativa ou uma métrica de risco quantitativa específica para hosts compartilhados. Complementarmente, (VS; SETHURAMAN; KHAN, 2023) discute estratégias de mitigação e o uso do modelo *DREAD* para priorização de riscos em *microservices*, o que fundamenta teoricamente a criação do índice *IEVI* apresentado nesta pesquisa.

A Tabela 1 apresenta uma comparação detalhada entre as funcionalidades e focos de análise dos trabalhos citados e a solução proposta por este artigo.

Tabela 1. Análise comparativa entre trabalhos relacionados e a abordagem proposta.

Trabalho	*Escalation*	*Tampering*	Multiusuário	Métrica	Ferramenta
(RAJYASHREE et al., 2024)	Sim	Não	Não	Não	Não
(WONG et al., 2023)	Sim	Sim	Não	Não	Não
(VS; SETHURAMAN; KHAN, 2023)	Sim	Sim	Não	Sim	Não
(SULTAN; AHMAD; DIMITRIOU, 2019)	Sim	Sim	Não	Não	Não
Este Trabalho	Sim	Sim	Sim	Sim	Sim

Diferente dos trabalhos anteriores, esta pesquisa foca especificamente na governança de segurança em máquinas de uso coletivo, integrando a detecção via *IntegrityGuard* à quantificação de risco pelo *IEVI*. Essa integração permite que administradores de sistemas não apenas identifiquem a falha de configuração no *daemon*, mas visualizem o impacto imediato na integridade do sistema operacional.

3. Metodologia

3.1. Ambiente Experimental

A definição do cenário de testes baseou-se em uma auditoria de segurança prática realizada nos laboratórios do Instituto Federal de Minas Gerais (*IFMG*), *campus* Ouro Branco, onde identificou-se que mais de 120 máquinas de uso compartilhado apresentavam vulnerabilidades críticas de governança. Nessas estações, usuários comuns pertenciam ao *group docker*, permitindo o acesso irrestrito ao *socket UNIX* (RAJYASHREE et al., 2024). Para validar as funcionalidades da ferramenta *IntegrityGuard* sem comprometer a infra-estrutura institucional, foi desenvolvida uma *Virtual Machine* (*VM*) com configuração técnica idêntica às máquinas vulneráveis encontradas no *campus*.

O ambiente controlado foi configurado em um *host* operando com *Ubuntu* 22.04 *LTS* e o *Docker Engine* na versão 24.0.7 (MERKEL et al., 2014). No interior desta *VM*, reproduziu-se fielmente a falha de configuração original, garantindo que o *daemon* operasse em modo *rootful* e que usuários não privilegiados pudessem interagir diretamente com o *kernel* hospedeiro através da *API* do *Docker* (WONG et al., 2023). Não foram implementadas camadas adicionais de *hardening*, como *AppArmor* ou *SELinux*, a fim de simular o comportamento padrão observado nas estações de trabalho do *IFMG* e quantificar a eficácia da mitigação via *Rootless mode* (VS; SETHURAMAN; KHAN, 2023).

3.2. Procedimentos Experimentais

A metodologia de avaliação foi estruturada para validar a eficácia da ferramenta *IntegrityGuard* na detecção de vulnerabilidades críticas discutidas na literatura de segurança de *containers* (SULTAN; AHMAD; DIMITRIOU, 2019; WONG et al., 2023). O ambiente de teste consistiu em dois cenários: uma configuração padrão (*rootful*), onde o utilizador possui acesso ao *socket* do *Docker*, e uma configuração mitigada (*rootless*), fundamentada no princípio do privilégio mínimo (DOCKER, 2025a). A ferramenta executa quatro rotinas de diagnóstico baseadas em vetores de ataque conhecidos:

1. Verificação de Grupo *Docker* (*docker group check*): Identifica se o utilizador pertence ao grupo secundário *docker*. Esta configuração é recorrentemente apontada como o principal facilitador para o acesso sem restrições ao *daemon*, equivalente funcionalmente ao acesso *root* no sistema hospedeiro.
2. Verificação de Leitura Sensível (*etc shadow check*): Simula um ataque de *information disclosure* ao tentar instanciar um *container* que realiza a montagem (*bind mount*) do diretório *etc* do *host*. O objetivo é verificar a possibilidade de leitura do ficheiro *shadow*, que contém *hashes* de senhas, uma vulnerabilidade de integridade e confidencialidade documentada por .
3. Verificação de Escrita no Hospedeiro (*create file on host check*): Avalia o risco de *data tampering* ao tentar criar um ficheiro de teste no diretório *tmp* do hospedeiro a partir de um processo isolado. Este teste valida se as barreiras de isolamento do *runtime* são permeáveis, permitindo modificações não autorizadas no sistema de ficheiros.
4. Verificação de Escalonamento de Privilégios (*elevate privilege check*): Representa o cenário de maior risco, onde a ferramenta tenta injetar um novo utilizador administrativo no ficheiro *passwd* do *host* com *User ID* (*UID*) 0. O sucesso desta operação confirma a viabilidade de *privilege escalation* através da exploração da interface do *Docker*, permitindo o controle total da infra-estrutura.

Após a execução destas verificações no cenário vulnerável, aplica-se a mitigação através do modo *rootless*, que utiliza *user namespaces* para isolar as capacidades do utilizador dentro do *container*. Os testes são então repetidos para quantificar a redução da superfície de ataque e a conformidade com as melhores práticas de *hardening* (VS; SETHURAMAN; KHAN, 2023).

3.3. Mérito e Aplicação do IEVI

O *Índice de Exposição à Violação de Integridade* (IEVI) é a métrica proposta neste trabalho para quantificar de forma objetiva o risco operacional decorrente da exposição do *socket* do *Docker* em ambientes compartilhados. O IEVI utiliza os valores atribuídos a cada uma das categorias do *framework DREAD* para consolidar um *score* que reflete o impacto combinado de *data tampering* e *privilege escalation* no sistema hospedeiro (VS; SETHURAMAN; KHAN, 2023). A formulação do índice é dada pela média aritmética dos critérios pontuados:

$$IEVI = \frac{D + R + E + A + D}{5} \times 10 \tag{1}$$

Nesta equação, as variáveis representam o Dano Potencial (D), a Reprodutibilidade (R), a Explorabilidade (E), os Usuários Afetados no *host* (A) e a Descoberta (D). Ao contrário da análise puramente qualitativa, o IEVI permite que administradores de sistemas estabeleçam prioridades de intervenção com base em uma escala centesimal.

Valores próximos a 100 indicam uma vulnerabilidade crítica com alta probabilidade de comprometimento total da integridade do sistema operacional (SULTAN; AHMAD; DIMITRIOU, 2019). Por outro lado, resultados abaixo de 40 são classificados como risco tolerável, sendo este o patamar esperado após a implementação de políticas de segurança baseadas no modo *rootless* e na restrição de acesso ao *daemon* (VS; SETHURAMAN; KHAN, 2023). Esta quantificação é essencial para validar a eficácia das mitigações propostas e monitorar a postura de segurança do *cluster* ao longo do tempo.

3.4. Prova de Conceito (PoC)

A *Prova de Conceito* (*PoC*) foi estruturada como uma execução sistemática da ferramenta *IntegrityGuard*, visando automatizar a exploração da interface *docker.sock* para validar riscos de integridade e privilégio amplamente discutidos na literatura (RAJYASHREE et al., 2024; SULTAN; AHMAD; DIMITRIOU, 2019). O fluxo de ataque horizontal, detalhado na Figura 2, demonstra a progressão lógica desde a validação de acesso inicial até o compromisso total do hospedeiro, seguindo metodologias de análise de vulnerabilidades em *runtimes* (WONG et al., 2023; VS; SETHURAMAN; KHAN, 2023).

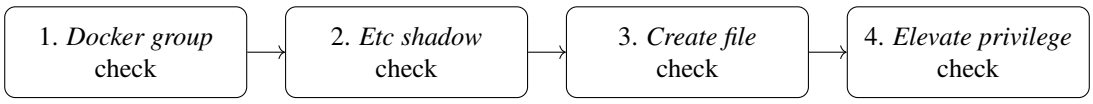

Figura 2. Fluxo de exploração fundamentado em vetores de ataque conhecidos.

Através desta sequência, a ferramenta comprova dois impactos críticos simultâneos categorizados pelo modelo *STRIDE* (SHOSTACK, 2014; WONG et al., 2023):

1. *Data Tampering*: a modificação não autorizada de arquivos do sistema operacional, como o *etc/passwd*, o que compromete a integridade dos dados e a confiança no *host*;
2. *Elevation of Privilege*: a criação de uma conta com *User ID* (*UID*) 0, garantindo ao atacante autoridade máxima sobre o *kernel* e os recursos de *hardware*.

O *TTE* registado foi inferior a 20 s, o que reforça a classificação desta ameaça como de alto risco e baixa barreira técnica em infra-estruturas partilhadas que não adotam o modo *rootless* (VS; SETHURAMAN; KHAN, 2023; RAJYASHREE et al., 2024).

4. *IntegrityGuard*

A ferramenta *IntegrityGuard* foi desenvolvida em *Python* como uma *command-line interface* (*CLI*), visando auxiliar administradores de *hosts* multiusuário na identificação e mitigação da exposição do *docker.sock*. O *software* adota uma arquitetura modular baseada em serviços, o que facilita a expansão de novos testes de vulnerabilidade e formatos de saída. Como marco de sua maturidade e disponibilidade para a comunidade de segurança, a ferramenta foi publicada no repositório oficial de pacotes *Python*, *PyPI*, sob o nome *integrityguard-cli*[1].

4.1. Arquitetura e Módulos Funcionais

A estrutura interna da ferramenta é composta por quatro componentes principais que colaboram para o diagnóstico completo do ambiente:

1. Interface de Linha de Comando (*Main CLI*): Provê os argumentos necessários para que o utilizador escolha o formato de saída, suportando tanto *JSON* quanto *HTML*, como demonstrado nas Figuras 3 e 4.
2. Orquestrador de Análise (*IntegrityGuardTool*): Responsável por gerir a execução sequencial dos *vulnerability checks* e coordenar a geração dos relatórios finais.
3. Calculadora de Risco (*IEVICalculator*): Este serviço implementa a lógica de mapeamento de severidade para os critérios *DREAD*. Ele utiliza um mapa de categorias que traduz níveis de risco (*Critical*, *High*, *Medium*, *Low* e *Info*) em pontuações numéricas para calcular a média final do IEVI.
4. Gerador de Relatórios (*HTMLReportGenerator*): Utiliza o motor de *templates* *Jinja2* para renderizar os resultados da análise em um formato visual acessível.

[1]Disponível em: ⟨https://pypi.org/project/integrityguard-cli/⟩

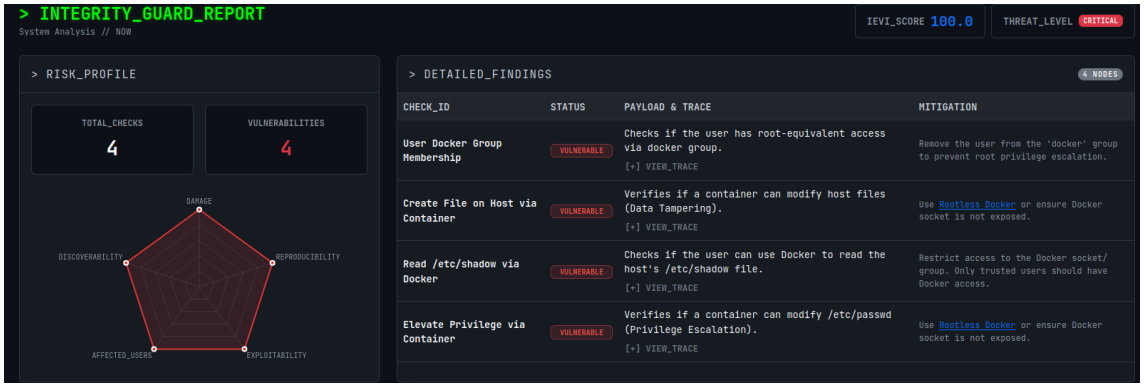

Figura 3. Interface do relatório visual em formato *HTML*.

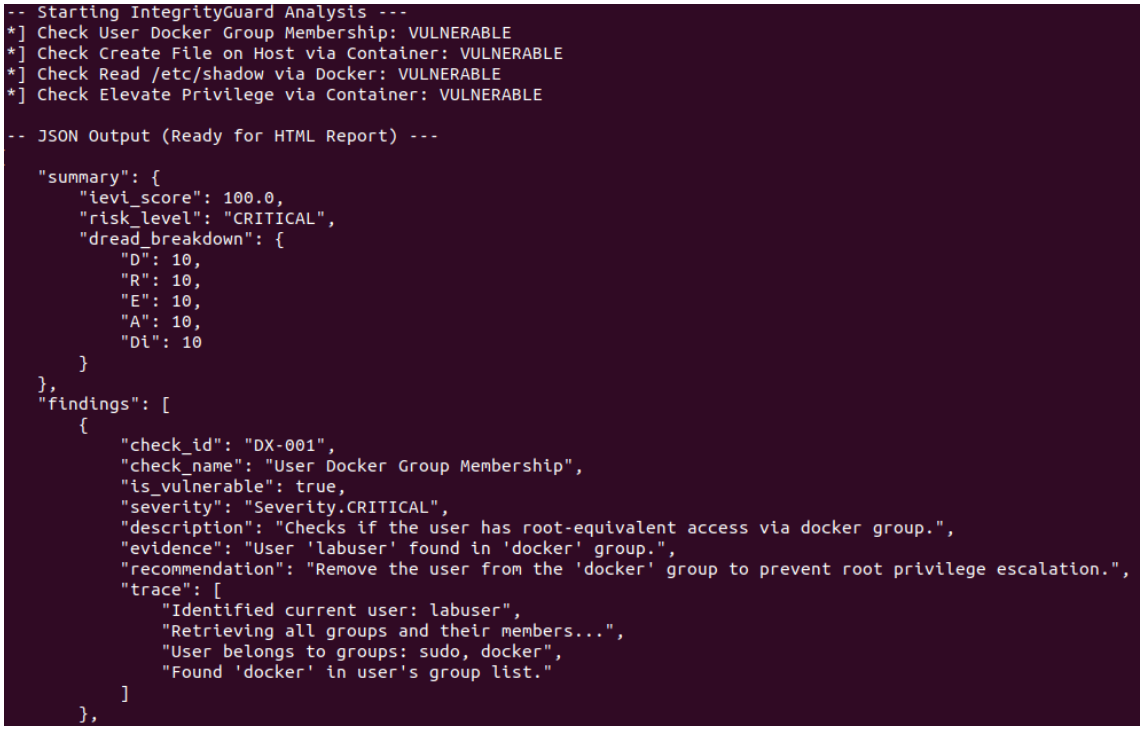

Figura 4. Interface de saída em formato *JSON* estruturado.

4.2. Fluxo de Operação e Cálculo do IEVI

O fluxo operacional inicia-se com a conexão ao *Docker engine* através da biblioteca *docker-py*. A ferramenta executa os *scripts* de verificação de grupo e a *Proof of Concept* (PoC) para determinar a viabilidade de exploração. A Figura 5 representa o fluxo de operação.

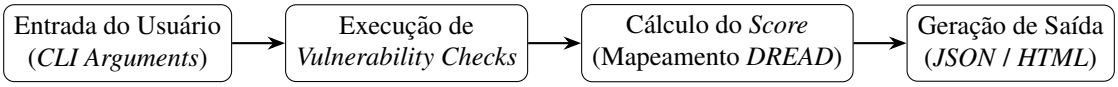

Figura 5. Fluxo de operação da ferramenta *IntegrityGuard* do *input* ao *reporting*.

Para permitir uma análise tanto automatizada quanto humana, a ferramenta disponibiliza dois formatos principais de saída. O formato *JSON* é ideal para integração com *pipelines* de *DevSecOps*, enquanto o relatório *HTML* provê uma visão executiva do nível de risco e recomendações de *hardening* (VS; SETHURAMAN; KHAN, 2023; SULTAN; AHMAD; DIMITRIOU, 2019).

Para cada vulnerabilidade encontrada, o *IEVICalculator* extrai o nível de severidade e aplica a fórmula de média aritmética ponderada, escalonando o resultado para uma base centesimal. Este processo garante que o índice reflita com precisão o dano potencial e a reprodutibilidade do ataque em cenários de *privilege escalation* e *data tampering* (WONG et al., 2023). Ao final, o administrador recebe um sumário executivo detalhando o *risk level* do sistema, permitindo ações rápidas de *hardening* como a transição para o modo *rootless* (VS; SETHURAMAN; KHAN, 2023; SULTAN; AHMAD; DIMITRIOU, 2019).

5. Resultados e Discussão

5.1. Avaliação Quantitativa com o *IEVI*

A aplicação da ferramenta *IntegrityGuard* permitiu mensurar o risco prático associado à exploração do *docker.sock* no *host* multiusuário analisado. O índice *IEVI*, calculado automaticamente pelo serviço *IEVICalculator* a partir das vulnerabilidades detectadas, resultou nos seguintes indicadores:

1. *IEVI* = 100 (*Risk Level: CRITICAL*) no Cenário Vulnerável (*rootful*). Figura 3;
2. *IEVI* = 32 (*Risk Level: LOW*) no Cenário Mitigado (*rootless*). Figura 6.

Esta redução de 68% na exposição valida a eficácia da migração para o modo *rootless* (DOCKER, 2025a). A Tabela 2 detalha as pontuações atribuídas a cada critério *DREAD* com base na severidade máxima identificada pelos módulos de verificação, refletindo a lógica de mapeamento implementada no *software*.

Tabela 2. Comparação da pontuação *IEVI* baseada no mapeamento de severidade da ferramenta.

Critério *DREAD*	Pré-Mitigação (*Critical*)	Pós-Mitigação (*Low*)
Dano Potencial (*D*)	10	2
Reprodutibilidade (*R*)	10	4
Exploitabilidade (*E*)	10	4
Usuários Afetados (*A*)	10	2
Descoberta (*Di*)	10	4
Total *IEVI*	100	32

5.2. Interpretação dos Resultados

No cenário vulnerável, a ferramenta *IntegrityGuard* confirmou a viabilidade de todos os vectores de ataque testados. O módulo *elevate privilege check* obteve sucesso na modificação do ficheiro *etc/passwd* em menos de 20 segundos, demonstrando que o isolamento lógico do *Docker* em modo privilegiado é insuficiente para conter um utilizador local mal-intencionado (RAJYASHREE et al., 2024). O diagnóstico apontou que a falha de integridade (*data tampering*) permitiu a escalada imediata para *root* (*elevation of privilege*) (WONG et al., 2023) 3.

No cenário mitigado por meio do modo *rootless*, os módulos *etc shadow check* e *create file on host check* falharam ao tentar aceder a recursos do hospedeiro. Isso ocorre porque o *daemon* opera dentro de um *user namespace* restrito, onde o utilizador, apesar de possuir *UID* 0 dentro do *container*, não detém privilégios reais sobre o *kernel* ou o sistema de ficheiros do *host* (DOCKER, 2025a; SULTAN; AHMAD; DIMITRIOU, 2019) 6. Assim, o impacto de uma possível invasão é limitado estritamente ao contexto do utilizador comum, impedindo o comprometimento sistémico observado anteriormente (VS; SETHURAMAN; KHAN, 2023).

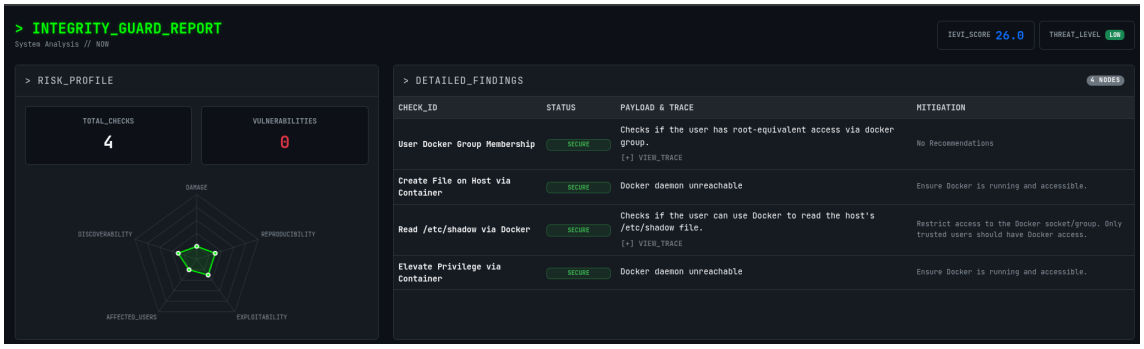

Figura 6. Interface do relatório em modo *rootless*

5.3. Convergência com o Modelo *Zero Trust*

Os resultados obtidos reforçam a necessidade de aplicar os princípios de *Zero Trust* na gestão de *containers*, especialmente o conceito de nunca confiar e sempre verificar (SHOSTACK, 2014). A mitigação automatizada sugerida pela ferramenta *IntegrityGuard* alinha-se a esta filosofia ao remover a confiança implícita concedida pelo grupo *docker* e pelo acesso directo ao *socket* privilegiado.

Ao adoptar o *hardening* via modo *rootless*, a infra-estrutura reduz significativamente o *blast radius* de ataques baseados em falhas de configuração (VS; SETHURAMAN; KHAN, 2023). Desta forma, a ferramenta não apenas detecta o risco, mas promove a consolidação de uma política de segurança dinâmica, essencial para manter a resiliência em ambientes onde múltiplos utilizadores partilham recursos computacionais sem supervisão constante (SULTAN; AHMAD; DIMITRIOU, 2019; WONG et al., 2023).

6. Limitações do Estudo

Apesar dos resultados demonstrarem a eficácia da ferramenta *IntegrityGuard* na detecção de riscos em *hosts* multiusuário, este estudo apresenta limitações técnicas e metodológicas que devem ser consideradas para a interpretação dos dados:

1. Escopo restrito ao vector *docker.sock*: A análise foca-se primordialmente na exploração da interface do *daemon* e na pertença inadequada ao grupo *docker* . Outros vectores críticos, como falhas de segurança no *kernel Linux*, o abuso de *capabilities* específicas do *runtime* ou vulnerabilidades em imagens provenientes do *Docker Hub*, não foram integrados no roteiro de testes automatizados da versão actual da ferramenta.
2. Dependência de mapeamento estático no *IEVI*: Embora o cálculo do índice seja automatizado pelo serviço *IEVICalculator*, as pontuações baseiam-se num mapeamento de severidade pré-definido (*Severity DREAD Map*) implementado no código-fonte. Esta abordagem, embora consistente, pode não capturar nuances específicas de infra-estruturas com políticas de segurança compensatórias ou monitorização dinâmica activa.
3. Ambiente de experimentação controlado: A validação foi conduzida em uma *Virtual Machine* desenhada para replicar as vulnerabilidades observadas nos laboratórios do *IFMG*. Em cenários de produção de larga escala, factores como a presença de *Intrusion Detection Systems* (*IDS*), latência de rede e sistemas de

logging centralizado podem influenciar tanto o *Time-to-Exploit* (*TTE*) quanto a facilidade de descoberta do ataque.
4. Foco exclusivo na mitigação via *Rootless*: O estudo validou o impacto da transição para o modo *rootless* como principal contramedida. No entanto, outras estratégias de *hardening* complementares, tais como a implementação de perfis de *AppArmor*, *SELinux* ou a utilização de *runtimes* de isolamento como o *gVisor*, não foram objecto de análise comparativa nesta fase da pesquisa.

7. Trabalhos Futuros

A partir dos resultados alcançados com o desenvolvimento da ferramenta *IntegrityGuard* e da aplicação da métrica *IEVI*, identificam-se diversas oportunidades para a evolução desta pesquisa:

1. Ampliação do escopo de vulnerabilidades: Expandir os módulos de teste para avaliar vectores adicionais de ataque, como o abuso de *capabilities* do *Linux*, a execução de *containers* privilegiados e a exploração de potenciais *escapes* de *kernel*.
2. Evolução do *IEVI* e do *IEVICalculator*: Refinar a lógica de mapeamento de severidade para incorporar factores dinâmicos de detecção e reduzir a dependência de ponderações estáticas, permitindo que o índice se adapte a diferentes contextos operacionais.
3. Integração em *Pipelines* de *DevSecOps*: Desenvolver conectores para que a ferramenta possa ser utilizada em fluxos de *Continuous Integration* e *Continuous Deployment* (*CI/CD*), permitindo a auditoria automatizada de *hosts* antes do *deployment* de novas aplicações.
4. Remediação Automática e *Hardening*: Integrar mecanismos que não apenas detectem a exposição, mas apliquem automaticamente medidas de mitigação, como a configuração assistida do modo *rootless* ou a geração de perfis restritivos de *AppArmor* e *SELinux*.
5. Validação em Infra-estruturas Heterogéneas: Conduzir estudos experimentais em *clusters* de larga escala e ambientes de nuvem pública para avaliar a eficácia do *IEVI* em sistemas com alta rotatividade de utilizadores e políticas de acesso complexas.
6. Modelagem Avançada de Ameaças: Aplicar formalismos complementares ao *STRIDE* e ao *DREAD*, como o *ATT&CK for Containers*, para enriquecer a análise de risco e a base de conhecimento da ferramenta.

Estas direcções indicam o potencial de evolução contínua na protecção de ambientes *Docker* multiusuário, alinhando as práticas de governança de segurança ao estado da arte em tecnologia de *containers*.

8. Conclusão

Este trabalho demonstrou que a configuração inadequada do *Docker* em ambientes multiusuário, especificamente a inclusão de utilizadores comuns no grupo *docker*, constitui uma vulnerabilidade crítica que compromete a integridade do sistema operacional . Através da execução da ferramenta *IntegrityGuard*, validou-se que o acesso ao *socket* privilegiado permite que actores mal-intencionados realizem ataques de *Data Tampering* e *Privilege Escalation* com baixo esforço técnico e elevado impacto sistémico.

A principal contribuição técnica deste estudo foi o desenvolvimento do ecossistema *integrityguard-cli*, publicado no repositório *PyPI*, que automatiza o diagnóstico de segurança e provê relatórios estruturados em *JSON* e *HTML*. A introdução do Índice de Exposição à Violação de Integridade (*IEVI*) permitiu uma quantificação objectiva do risco, fundamentada no *framework DREAD*, oferecendo aos administradores de sistemas uma métrica precisa para priorizar acções de *hardening* .

Os resultados experimentais confirmaram que a adopção do modo *rootless* reduz drasticamente a superfície de ataque, baixando o *score* do *IEVI* de um nível crítico para um patamar tolerável. Esta mitigação alinha-se aos princípios de *Zero Trust*, garantindo que mesmo em caso de falha no isolamento do *container*, o atacante permaneça restrito aos privilégios de um utilizador comum, protegendo o *kernel* e ficheiros sensíveis do hospedeiro.

Em última análise, este trabalho provê uma base metodológica e uma ferramenta prática para aprimorar a segurança em infra-estruturas partilhadas. A conscientização sobre os riscos do *socket* privilegiado e a transição para arquitecturas sem privilégios de *root* são passos fundamentais para a consolidação de políticas de segurança robustas no uso de tecnologias de conteinerização.

Referências

DOCKER, I. *Rootless mode — Docker Engine — Security*. 2025. Acessado em: 23 nov. 2025. Disponível em: ⟨https://docs.docker.com/engine/security/rootless/⟩.

DOCKER, I. *What is Docker? — Docker Docs*. 2025. Acessado em: 19 dez. 2025. Disponível em: ⟨https://docs.docker.com/get-started/docker-overview/#the-docker-daemon⟩.

MERKEL, D. et al. Docker: lightweight linux containers for consistent development and deployment. *Linux j*, v. 239, n. 2, p. 2, 2014.

RAJYASHREE, R. et al. An empirical investigation of docker sockets for privilege escalation and defensive strategies. *Procedia Computer Science*, Elsevier, v. 233, p. 660–669, 2024.

SHOSTACK, A. *Threat modeling: Designing for security*. [S.l.]: John wiley & sons, 2014.

SULTAN, S.; AHMAD, I.; DIMITRIOU, T. Container security: Issues, challenges, and the road ahead. *IEEE Access*, v. 7, p. 52976–52996, 2019.

VS, D. P.; SETHURAMAN, S. C.; KHAN, M. K. Container security: precaution levels, mitigation strategies, and research perspectives. *Computers & Security*, Elsevier, v. 135, p. 103490, 2023.

WONG, A. Y. et al. On the security of containers: Threat modeling, attack analysis, and mitigation strategies. *Computers & Security*, Elsevier, v. 128, p. 103140, 2023.