ANDERSON FRASÃO

DETECÇÃO DE ANOMALIAS EM CONTÊINER: UMA AVALIAÇÃO CONSIDERANDO O NÍVEL DE OBSERVAÇÃO DAS APLICAÇÕES NO CONTÊINER

(*versão pré-defesa, compilada em 27 de julho de 2023*)

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Ciência da Computação, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Carlos Alberto Maziero.

Coorientador: Tiago Heinrich.

CURITIBA PR

2023

RESUMO

A adoção crescente de tecnologias de virtualização ao longo da última década contribuiu com o desenvolvimento da virtualização baseada em contêineres. No entanto, o aumento do uso e implementação desses ambientes isolados também tem gerado preocupações quanto à sua segurança. Diante da proximidade entre o hospedeiro e o contêiner, surgem abordagens que utilizam detecção de intrusões através de anomalias como uma opção para monitorar e identificar comportamentos inesperados. Este estudo propõe o emprego de interações entre contêiner e Sistema operacional para detectar ameaças em um ambiente de contêiner, sendo a extração de dados feita internamente ao ambiente, para diminuir a sobrecarga de dados e empregando algoritmos de Aprendizado de Máquina (ML) para distinguir entre comportamentos normais e anômalos.

Palavras-chave: Detecção de Intrusão, Segurança de Computadores, e Contêineres.

ABSTRACT

The increasing adoption of virtualization technologies over the past decade has contributed to the development of container-based virtualization. However, the increased use and deployment of these isolated environments has also raised concerns about their security. In light of the proximity between the host and the container, approaches are emerging that use intrusion detection through anomalies as an option to monitor and identify unexpected behavior. This study proposes employing interactions between container and OS to detect threats in a container environment, with data extraction done internally to the environment to decrease data overload and employing Machine Learning (ML) algorithms to distinguish between normal and anomalous behaviors.

Keywords: Intrusion Detection, Computer Security, and Containers.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

BoSC	Bag of System Calls
HIDS	Host-based Intrusion Detection System
IDS	Intrusion Detection System
ML	Machine Learning
MLP	Multilayer Perceptron
KNN	K-Nearest Neighbors
NIDS	Network Intrusion Detection System
RF	Random Forest
RCE	Remote Code Execution
SO	Sistema Operacional
STIDE	Sequence Time-Delay Embedding
SVM	Support Vector Machine
XSS	Cross-site Scripting

SUMÁRIO

1 INTRODUÇÃO

Os contêineres oferecem vantagens em termos de desempenho e escalabilidade em comparação com maquinas virtuais. Sendo ideais para executar múltiplas aplicações com o mesmo sistema operacional, pois possuem menos sobrecarga, permitindo um maior número de instâncias a ser executado em comparação com o recurso computacional utilizado por máquinas virtuais. No entanto, os contêineres têm um isolamento mais próximo ao sistema operacional do hospedeiro em relação às máquinas virtuais. (Joy, 2015) demonstra que os contêineres Linux mostraram-se superiores em termos de desempenho e escalabilidade, processando mais solicitações em menos tempo em comparação com as máquinas virtuais.

A introdução de aplicações em contêineres é um marco importante na evolução tecnológica. Os contêineres são baseados no conceito de contêiner utilizado na indústria de frete, onde a carga é transportada de forma segura e eficiente em contêineres padronizados. O ambiente contêiner é uma tecnologia baseada em Linux que permite que várias instâncias de aplicativos compartilhem um único sistema operacional, eliminando a necessidade de cada instância ter seu próprio sistema operacional. Embora os contêineres não sejam uma tecnologia nova, eles ganharam destaque recentemente devido à demanda por implantação rápida de aplicativos. Os contêineres oferecem diversos benefícios, como portabilidade, menor consumo de memória, facilidade de migração e implantação rápida. Em comparação com a virtualização baseada em *hypervisor*, os contêineres compartilham o mesmo sistema operacional *host*, resultando na redução de recursos utilizados e permitindo que mais instâncias sejam hospedadas em um único servidor (Sturm et al., 2017).

Os riscos intrínsecos ao uso de contêineres estão relacionados à imagem, ao orquestrador, ao próprio contêiner, ao sistema operacional do *host* e à implementação. Os `riscos da imagem` incluem componentes desatualizados, vulnerabilidades e inclusão de arquivos maliciosos. Os `riscos do orquestrador` envolvem práticas inadequadas de gerenciamento de contas e autenticação. Os `riscos do contêiner` incluem acesso ilimitado à rede e exploração de vulnerabilidades. Os `riscos do sistema operacional do host` são ampliados pelo uso de um *kernel* compartilhado. Os `riscos de implementação` estão relacionados à falta de segurança, configurações incorretas e vulnerabilidades em componentes específicos. Para a mitigação dos riscos associados com o uso de contêiner, são necessárias boas práticas de segurança, como atualizações regulares, controle de privilégios, monitoramento e adoção de políticas de segurança adequadas. Também é importante escolher imagens confiáveis e garantir a assinatura digital antes da implantação, (Martínez-Magdaleno et al., 2021).

A ameaça atual no campo da tecnologia da informação requer o monitoramento e detecção precoce de ataques em sistemas, (Röhling et al., 2019). Os Sistemas de Detecção de Intrusão Baseados em *Host* (HIDS) desempenham um papel crucial nesse processo, permitindo a

inspeção das chamadas de sistema e a análise dos processos que acessam os sistemas. Os HIDS baseados em anomalias são particularmente eficazes na detecção de ataques desconhecidos, pois são treinados com comportamentos normais e identificam comportamentos anômalos durante um ataque. No entanto, a qualidade de um HIDS baseado em anomalias depende significativamente dos dados e da estratégia para a identificação de anomalias. Algumas das solução no meio de detecção de intrusão em contêiner são:

Integridade de instâncias do contêiner: O Amazon ECS oferece monitoramento de integridade para instâncias de contêiner, verificando problemas que possam impedir sua execução. As verificações automatizadas ocorrem a cada dois minutos, retornando status de aprovação ou falha. Se todas as verificações forem aprovadas, o status geral será OK; caso contrário, será *IMPAIRED* (Amazon, 2023).

Análise de comportamento: Os contêineres Linux estabelecem comunicação com o núcleo do sistema operacional do hospedeiro por meio de chamadas de sistema. Ao observar as chamadas de sistema entre o contêiner e o núcleo do sistema operacional do hospedeiro, é viável compreender o comportamento do contêiner e identificar possíveis alterações que possam indicar uma tentativa de invasão contra o contêiner (Abed et al., 2015b).

Sistemas de prevenção de intrusão (IPS): Os sistemas de Prevenção de Invasões (IPS - *Intrusion Prevention System*) são considerados como uma ampliação dos sistemas de Detecção de Invasões (IDS - *Intrusion Detection System*), pois agem no sistema após a detecção de ameaças ou atividades maliciosas. (Lopez et al., 2014) propõe o BroFlow, um sistema de Detecção e Prevenção de Intrusão (IDPS) distribuído para a defesa contra ataques de negação de serviço (DoS).

Este trabalho de conclusão de curso, apresenta um estudo de estratégias para a detecção de intrusão baseado em anomalia em um ambiente de virtualização em nível de contêiner. Considerando limitações de estudos anteriores como o ponto de observação dos contêiner em nível do sistema hospedeiro (Castanhel et al., 2021). A respectiva proposta foca em um processo de detecção que não esteja sendo prejudicado pelo ruído gerado pelo *container engine*, sem prejudicar o isolamento das aplicações em execução. O estudo consisti em executar uma aplicação *Web* em um contêiner Docker, com o objetivo de monitorar seu comportamento tanto em situações normais de uso quanto em casos de ataques ou operações maliciosas. Em seguida, será testada uma técnicas de classificação para avaliar a eficiência da detecção de anomalia nesse contexto.

1.1 OBJETIVOS

Objetivo geral: Conduzir uma pesquisa focada na identificação de invasões por meio de anomalias em um ambiente de contêiner Docker.

Objetivos específicos:

- Avaliar a eficácia de um método de classificação de aprendizado de máquina na detecção de anomalias em contêiner;
- Investigar a viabilidade ao considerar o isolamento do contêiner e diferentes níveis de observação ao utilizar uma estratégia de detecção de anomalia dentro do contêiner;
- Criar uma base de dados que represente o comportamento da aplicação web em um ambiente de contêiner, com ênfase no comportamento anômalo; e
- Apresentar uma nova estratégia para a detecção de anomalias em um ambiente de contêiner.

1.2 METODOLOGIA

Primeiramente foi realizada uma investigação de ferramentas que permitiam a coleta de dados no ambiente de contêiner, sem prejudicar o isolamento da aplicação. Para as próximas etapas será criado um conjunto de dados relacionados ao comportamento do contêiner, utilizando uma aplicação web, mais especificamente o *Wordpress*. Para gerar esses dados sera feita uma coleção de testes reproduzindo tanto interações comuns, quanto interações maliciosas. Após sendo realizada a avaliação dos testes, será feito o treinamento de um modelo de aprendizado de máquina, para então realizar testes sobre a identificação de anomalias em contêineres.

1.3 ORGANIZAÇÃO TEXTUAL

Neste trabalho, há uma divisão em seis partes. O capítulo 2 aborda a fundamentação teórica, explorando conceitos relacionados à detecção de anomalia, sistemas de detecção de intrusão, virtualização baseada em contêiner e diversos métodos de detecção. O capítulo 3 realiza uma revisão bibliográfica da literatura existente sobre detecção de anomalia. No capítulo 4, é apresentada a proposta, descrevendo a estratégia adotada, o processo de coleta e filtragem dos dados.

2 FUNDAMENTAÇÃO TEÓRICA

Nas Seção 2.1 é apresenta os conceitos para detecção de intrusão. Na Seção 2.2 é apresentado detecção por anomalias. Seção 2.3 apresenta sistemas de virtualização, destacando o ambientes de produção a segurança. A Seção 2.4 apresenta *Machine Learning* (ML) e seu uso para a detecção de ameaças.

2.1 SISTEMA DE DETECÇÃO DE INTRUSÃO

Existe uma crescente preocupação com a segurança na Internet e nos sistemas de computadores, devido ao aumento do uso de redes e das ameaças à segurança desses sistemas, (Liao et al., 2013). Invasão e vulnerabilidades em computadores ou sistemas de informação podem ter resultados desastrosos como violar políticas de segurança de computadores.

Sistemas de Detecção de Intrusão (IDS) são ferramentas usadas pelos administradores de segurança para detectar comportamentos ou ações intrusivas. Efetuando a análise dos dados da auditoria e ajudam a estabelecer a culpabilidade do atacante, detectar a extensão dos danos e prevenir futuros ataques. O IDS é uma ferramenta valiosa tanto para análises em tempo real como após a ocorrência de um ataque (Laureano et al., 2003).

As três principais categorias de detecção de intrusão são: detecção baseada em assinatura (SD), detecção baseada em anomalia (AD) e análise de protocolo com estado (SPA). A detecção baseada em assinatura compara padrões de eventos capturados com padrões conhecidos de ameaças ou ataques, enquanto a detecção baseada em anomalia compara comportamentos observados com perfis normais previamente definidos (Liao et al., 2013).

Dentre os IDS existentes, ha dois que se destacam, um deles é o Sistema de Detecção de Intrusão de Rede (NIDS) que utiliza mineração de dados para detectar intrusões em redes. Ele recebe dados capturados pelo dispositivo de captura de dados, que coleta informações do cabeçalho do pacote. Esses dados são armazenados em arquivos e, em seguida, passam por uma filtragem para remover o tráfego indesejado (Vinchurkar e Reshamwala, 2012). Já o Sistema de Detecção de Intrusão de Hospedeiro (HIDS) é um sistema de detecção de intrusão que monitora as atividades em um *host*, como *logs* do sistema e do *shell*, para identificar comportamentos não autorizados. Ele utiliza métodos de mineração de dados, como redes neurais artificiais, para analisar os dados de auditoria do *host* e detectar ataques (Liu et al., 2018).

Uma técnica para a detecção de intrusão em *host* é baseado na observação de *logs* do sistema (Bridges et al., 2019). Esse tipo de análise é desafiador porque pode ser intensivo em termos de processamento e geralmente requer especialização humana. Alguns métodos destacados incluem o uso da teoria da probabilidade *naive-bayes*, técnicas de recuperação de informações e análise de *clustering* de usuários. Em geral, os resultados mostram que, embora a

análise de *logs* possa produzir uma alta taxa de falsos positivos e muitas ameaças não detectadas, ela ainda é uma parte valiosa de um IDS.

Diversos estudos abordam à detecção de intrusão em sistemas de informação a partir de *logs* de auditoria. Os *logs* são mais detalhados do que os *logs* de sistema e contêm informações como conexões de rede, ações em linha de comando, escalonamentos de privilégios e alterações em arquivos. Pesquisadores utilizam diversas técnicas para a análise desses *logs*, incluindo modelos de Markov, lógica difusa e máquinas de vetores de suporte. Os resultados obtidos variam, mas mostram que a análise dos eventos de auditoria é uma técnica útil para detectar atividades maliciosas. No entanto, a coleta e gerenciamento desses *logs* é caro e consome muitos recursos.

2.2 DETECÇÃO DE ANOMALIA

Detecção de anomalia é uma estratégia para a detecção de padrões em dados que não seguem um comportamento normal e podem ser causados por atividades maliciosas ou falhas no sistema (Chandola et al., 2009). De acordo com (Lakhina et al., 2004), as anomalias são alterações significativas nos níveis de tráfego que afetam várias conexões.

A detecção de anomalias é uma tarefa importante de análise de dados que envolve a descoberta de padrões raros nos dados (Ahmed et al., 2016). As anormalidade são consideradas importantes porque indicam eventos significativos, mas raros, e podem levar a ações críticas em uma ampla gama de domínios de aplicação. Três classificações de detecção estatística de irregularidade são:

Modelo de mistura: Proposto por (Eskin, 2000), como uma abordagem para detectar anomalias em dados ruidosos. Nesse modelo, cada elemento é classificado como tendo uma pequena probabilidade de pertencer a uma classe anômala ou como pertencente à maioria dos elementos com probabilidade de 1-λ. Os autores assumem que os elementos com probabilidade de 1-λ representam o uso legítimo do sistema, enquanto as anormalidade têm probabilidade de λ.

Técnica de processamento de sinais: Método baseado em detecção de mudanças abruptas é apresentado por (Thottan e Ji, 2003), que descrevem anomalias como falhas de rede, problemas de desempenho e ataques de negação de serviço. O método utiliza bases de informações de gerenciamento para produzir uma função de saúde da rede e detectar comportamentos incomuns através de mudanças abruptas em suas estatísticas.

Análise de componentes principais (PCA): A técnica PCA usa uma combinação linear de variáveis aleatórias para criar componentes principais, que são descorrelacionados e ordenados em ordem decrescente de variância. Esses componentes podem ser usados para detecção de irregularidade com baixa complexidade computacional.

As principais técnicas para detecção de anomalias, incluindo métodos baseados em distância, densidade, modelo estatístico, aprendizado de máquina e mineração de dados (Hodge e Austin, 2004).

Distância: Esta abordagem envolve a definição de uma medida de distância entre os pontos de dados e a identificação de pontos que estão distantes da maioria dos outros pontos. Tais como o k-Nearest Neighbor e o Local Outlier Factor (LOF).

Densidade: efetua a identificação de pontos de dados que têm uma densidade menor do que a dos pontos ao seu redor. Como o DBSCAN e o OPTICS, sendo limitados pela sensibilidade à escolha dos parâmetros de densidade e distância.

Estatístico: Esta abordagem envolve a modelagem da distribuição dos dados e a identificação de pontos que não se encaixam no modelo. Tais como o método de Boxplot e o método de Dixon.

Aprendizado de máquina: Esta abordagem envolve o uso de técnicas de aprendizado de máquina para identificar padrões nos dados. Algoritmos como o *Isolation Forest* e o *One-class SVM* aprendem uma classe de características e classificam padrões que diferem destes comportamentos como anomalia.

Mineração de dados: é responsável pela identificação de padrões anormais nos dados usando técnicas de mineração de dados, tais como associação de regras e agrupamento. Como o algoritmo de detecção de anomalias baseado em regras (ROD) e o algoritmo de agrupamento com detecção de anormalidade (COP).

2.3 VIRTUALIZAÇÃO BASEADA EM CONTÊINER

A virtualização baseada em contêineres aproveita os recursos do kernel para criar um ambiente isolado para os processos. Ao contrário da virtualização baseada em hipervisor, os contêineres não possuem seu próprio hardware virtualizado, mas utilizam o hardware do sistema hospedeiro. Isso significa que o software em execução nos contêineres se comunica diretamente com o kernel do hospedeiro e precisa ser executado no sistema operacional e na arquitetura da *Central Processing Unit* (CPU) em que o hospedeiro está sendo executado. A ausência da necessidade de emular hardware e inicializar um sistema operacional completo permite que os contêineres sejam iniciados rapidamente, em questão de milissegundos, e sejam mais eficientes do que as máquinas virtuais convencionais. As imagens dos contêineres geralmente são menores do que as imagens das máquinas virtuais, pois não precisam incluir uma cadeia completa de ferramentas para executar um sistema operacional, como drivers de dispositivo, kernel ou sistema de inicialização, (Eder, 2016).

A virtualização baseada em contêiner é usada para isolar aplicativos Linux ou contêineres em um mesmo servidor, enquanto a virtualização de *hardware* completo é usada para melhorar o

desempenho. A virtualização baseada em contêiner é vantajosa para agendar tarefas em tempo real através de uma reserva de CPU, (Merkel et al., 2014).

Na figura 2.1 é ilustrada a comparação entre arquiteturas de virtualização baseada em contêiner e a virtualização baseada em *hypervisor*.

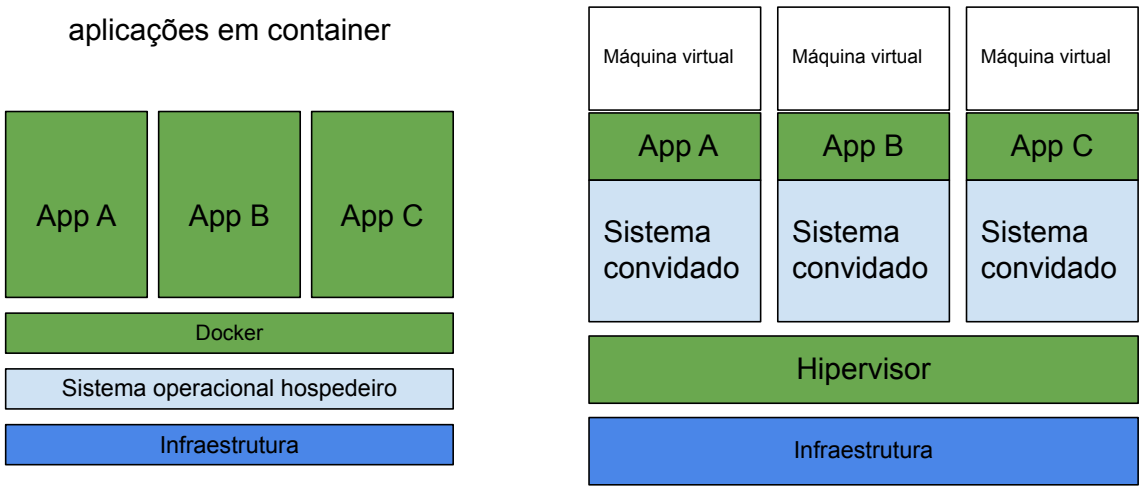

Figura 2.1: Comparação entre VM e Contêiner, com base em (Docker, 2023).

A virtualização baseada em contêineres cria instâncias isoladas do espaço do usuário em vez de máquinas virtuais completas. Essa forma de virtualização compartilha um único *kernel* do sistema operacional entre as instâncias virtuais, o que resulta em um isolamento mais fraco em comparação com a virtualização baseada em hipervisor. Alguns exemplos de sistemas de virtualização baseados em contêineres (Xavier et al., 2013):

Linux-VServer: Implementações mais antigas de sistema baseado em contêiner Linux. Em vez de usar *namespaces* para garantir o isolamento, o Linux-VServer introduziu seus próprios recursos no kernel do Linux, como isolamento de processos, isolamento de rede e isolamento de CPU. Usando a chamada do sistema *chroot* para limitar o escopo do sistema de arquivos para processos. A isolação de processo é realizada por meio de um espaço PID global. O Linux-VServer não virtualiza subsistemas de rede, mas usa identificadores de contêiner em pacotes para garantir que apenas o contêiner correto receba o tráfego. O Linux-VServer usa um esquema de *token bucket* para fazer isolamento de CPU. Os limites de recursos são estabelecidos por meio de chamadas de sistema e a ferramenta *rlimit*. O Linux-VServer é escalável, mas não suporta técnicas de virtualização usuais, como migração ao vivo, *checkpoint* e retomada. As versões recentes suportam *cgroups* e são gerenciadas pelo pacote de ferramentas *util-vserver*.

Docker: Uma ferramenta de linha de comando que permite criar, gerenciar e implantar contêineres. O Docker requer pouco conhecimento técnico em comparação com o uso do LXC para a mesma tarefa e oferece vários recursos essenciais para trabalhar com contêineres

em um ambiente de produção. O Docker combina tecnologias para isolar processos com outras ferramentas úteis, facilitando a criação de imagens de contêiner baseadas em outras imagens de contêiner (Eder, 2016). De acordo com (RedHat, 2023a), o Docker é uma tecnologia que utiliza o kernel do Linux para criar e executar processos de forma independente em containers. Isso permite que vários processos e aplicativos sejam executados separadamente, maximizando o uso da infraestrutura e mantendo a segurança. O Docker usa um modelo de implantação baseado em imagem, facilitando o compartilhamento de aplicativos e serviços, incluindo todas as suas dependências. Além disso, o Docker automatiza a implantação de aplicativos em ambientes de containers. Essas ferramentas baseadas em containers Linux tornam o Docker fácil de usar, proporcionando aos usuários acesso a uma ampla variedade de aplicativos e controle total sobre versões e distribuição, além da capacidade de implantar rapidamente.

Kubernetes: É composto por diferentes componentes que trabalham juntos para criar e gerenciar um cluster. É uma plataforma de orquestração de contêineres de código aberto. Permite o gerenciamento automatizado de implantação, escalonamento e operação de aplicativos em contêineres. O Kubernetes fornece recursos avançados, como balanceamento de carga, gerenciamento de armazenamento, auto-recuperação e dimensionamento automático, tornando-se uma escolha popular para implantação e gerenciamento de aplicativos em grande escala. Funciona com contêineres Docker e outras tecnologias de contêineres, fornecendo uma camada de abstração adicional para simplificar a implantação e o gerenciamento de aplicativos distribuídos, (Kubernetes, 2023).

A tecnologia de virtualização de nível de sistema operacional conhecida como Linux container permite isolar e conter um ou mais processos em um ambiente que aparenta ser o sistema inteiro, mas que na verdade compartilha o *kernel* Linux com outros containers. A tecnologia utiliza dois mecanismos principais, *namespace* e *cgroup*, para fornecer isolamento e limitar o uso de recursos, como CPU e memória. No entanto, a interdependência e a relação de influência mútua entre esses mecanismos de segurança do kernel podem fazer com que prejudiquem sua capacidade de proteção, (Lin et al., 2018).

2.3.1 Isolamento

Os *namespaces* do *kernel* Linux são usados para separar processos uns dos outros, criando diferentes espaços de nomes para processos que precisam ser isolados. Os *namespaces* permitem a criação de contêineres contendo processos com identificadores de processo (PID) que já estão sendo usados no sistema hospedeiro ou em outros contêineres. Isso facilita a migração de contêineres suspensos.

Os grupos de controle (*cgroups*) são usados para rastrear processos e grupos de processos, permitindo o controle e a limitação flexível de recursos. Embora não sejam obrigatórios para isolar processos, os *cgroups* fornecem um mecanismo para controlar e gerenciar o uso de

recursos, garantindo que recursos físicos não sejam desperdiçados ou indisponíveis devido a outros processos.

O Controle de Acesso Obrigatório (MAC) é um conjunto de conceitos que define a gestão do controle de acesso e é usado para fortalecer os mecanismos de controle de acesso no Linux/Unix. Ao qual, permite restringir o acesso a recursos sensíveis que não precisam ser acessados em determinado contexto.

Os *namespaces* do kernel e os *cgroups* são fundamentais para a virtualização baseada em contêineres pois fornecem isolamento de processos e recursos, portabilidade, eficiência e escalabilidade. A aplicação dessas ferramentas de segurança aos contêineres adiciona mais uma camada de proteção para mitigar ataques contra o hospedeiro e outros contêineres de dentro do contêiner.

2.3.2 Recursos do ambiente de container

Desde a criação de imagens personalizadas até a orquestração eficiente de containers em *clusters*, os recursos disponíveis oferecem um amplo espectro de possibilidades para otimizar a produtividade e a confiabilidade no ciclo de vida de uma aplicação. Os ambientes de contêiner fornecem diversos recursos, como o isolamento de recursos, a portabilidade entre ambientes, a gestão de dependências, a escalabilidade horizontal, a integração contínua e a implantação contínua.

O Docker é uma plataforma de código aberto criada pela Docker.Inc, usando a linguagem Go. Com alta performance, essa ferramenta facilita a criação e gestão de ambientes isolados, permitindo a rápida disponibilização de programas aos usuários finais. A seguir é apresentado algumas características de Docker:

Modularidade: A abordagem do Docker para a conteinerização permite desativar partes de uma aplicação para reparo ou atualização sem interromper completamente sua execução. Além disso, o Docker possibilita o compartilhamento de processos entre várias aplicações, seguindo uma abordagem semelhante à arquitetura orientada a serviços (SOA).

Camadas e controle de versão de imagens: O Docker utiliza camadas para compor suas imagens, permitindo a reutilização e aceleração do processo de criação de containers. Cada modificação na imagem gera uma nova camada, otimizando velocidade, tamanho e eficiência. Estas mudanças são registradas, proporcionando controle completo sobre as imagens do contêiner.

Reversão: É possível recuperar a versão anterior usando a função de reversão. Isso acontece devido às camadas criadas no ambiente contêiner. Esse processo é ainda mais eficiente porque é compatível com a abordagem de desenvolvimento ágil. Assim, a equipe pode contar facilmente com as práticas de integração e implantação contínua, mantendo a eficiência no desenvolvimento da aplicação.

Implantação rápida: A implantação de *software* costumava ser um processo demorado que levava dias. No entanto, com os containers Docker, é possível implantar em apenas alguns segundos. Ao criar um contêiner para cada processo, é possível compartilhá-los facilmente com novas aplicações. Não é necessário reiniciar o sistema operacional para adicionar ou mover um contêiner, o que reduz significativamente o tempo de implantação. Além disso, é fácil e econômico criar e destruir dados associados aos containers, sem qualquer preocupação.

Além do Docker, o Kubernetes oferece várias vantagens ao ambiente de desenvolvimento e implementação de aplicativos na nuvem. A principal vantagem é a capacidade de programar e executar containers em clusters de máquinas virtuais ou físicas de forma eficiente. Isso permite a otimização do desenvolvimento de apps para a nuvem e o uso de uma infraestrutura baseada em containers confiável para ambientes de produção (RedHat, 2023b).

Além disso, o Kubernetes possui outros recursos desejáveis. Tais como: (i) A possibilidade de orquestrar containers em vários *hosts*; (ii) Maximizar o uso de *hardware* para executar *apps* empresariais; (iii) Controlar e automatizar atualizações e implantações de aplicações; (iv) Adicionar armazenamento para *apps stateful*; (v) Escalar aplicações e recursos correspondentes rapidamente; (vi) Gerenciar serviços de forma assertiva para garantir a execução adequada das aplicações; (vii) Permitir a autorrecuperação e verificação de integridade das *apps* por meio de automação. Desse modo, este software é especialmente útil para implementação de vários contêineres e/ou vários *hosts*.

2.4 MACHINE LEARNING

A aprendizagem é um fenômeno complexo que envolve a aquisição de conhecimento, o desenvolvimento de habilidades e a descoberta de novos fatos. Desde o advento dos computadores, os pesquisadores têm buscado capacitar as máquinas com essas habilidades, o que representa um desafio fascinante na área de inteligência artificial. A aprendizagem de máquina é o campo de estudo que investiga e modela esses processos de aprendizagem. Os objetivos da aprendizagem de máquina incluem melhorar o desempenho em tarefas específicas, simular o aprendizado humano e explorar métodos e algoritmos de aprendizagem. Embora cada objetivo tenha seu foco específico, o progresso em um geralmente impulsiona o progresso nos outros, um exemplo é quando a investigação da aprendizagem humana pode ser utilizada para orientar a exploração de diferentes métodos de aprendizagem. Além disso, a análise teórica pode fornecer perspectivas valiosas para pesquisas no campo da psicologia. Essa interação desafia e fortalece o campo da inteligência artificial como um todo (Carbonell et al., 1983).

Existem várias categorias nas quais os algoritmos de aprendizado de máquina são agrupados, algumas delas são apresentadas a seguir, (Nasteski, 2017):

Aprendizado supervisionado: Os diferentes algoritmos produzem uma função que relaciona as entradas aos resultados desejados. Uma forma comum de descrever a tarefa de

aprendizado supervisionado é o problema de classificação, no qual o objetivo é que o algoritmo aprenda a função que associa um vetor a uma das diversas classes. Isso é realizado através da observação de vários exemplos de entrada e saída dessa função, buscando se aproximar do seu comportamento.

Aprendizado não supervisionado: Cria um modelo para um conjunto de entradas onde exemplos rotulados não estão disponíveis.

Aprendizado semi supervisionado: Utiliza tanto exemplos rotulados quanto não rotulados em conjunto para gerar uma função ou classificador adequado.

Aprendizado por reforço: O algoritmo adquire uma política de ação com base na observação do ambiente. Cada ação realizada pelo algoritmo causa um impacto no ambiente, e o ambiente fornece um *feedback* que orienta o processo de aprendizado do algoritmo.

Com a grande quantidade de dados produzidos diariamente, é necessário atualizar constantemente os modelos de ML para lidar com ameaças emergentes. O uso de métodos totalmente supervisionados é problemático devido ao desequilíbrio extremo entre as classes de ataques e exemplos normais, o que torna a rotulação dos dados um gargalo no processo de aprendizado. Além disso, é mostrado um esquema para desenvolver e avaliar soluções de aprendizagem automática para a cibersegurança, esse esquema inicia com coleta de dados, em seguida é feita a extração de atributos, extração de características, após essa etapa, há a opção de treinar um modelo ou ir para os testes. Os testes tem duas etapas, a avaliação e a atualização do modelo (Ceschin et al., 2020).

O Aprendizado de Máquina é um campo que usa experiência e dados para melhorar o desempenho e fazer previsões precisas. Os algoritmos de aprendizagem são projetados para serem eficientes e precisos, levando em consideração a complexidade dos dados e a quantidade de amostras necessárias (Mohri et al., 2018). A aprendizagem de máquina combina conceitos de ciência da computação, estatística, probabilidade e otimização para análise de dados e tomada de decisões.

A seguir são mostrados alguns cenários de aprendizado de máquina:

Aprendizado supervisionado: Nesse cenário, o aprendiz recebe exemplos de treinamento com rótulos e faz previsões para novos pontos não vistos. É comumente usado em problemas de classificação, regressão e ranking, como a detecção de spam.

Aprendizado não supervisionado: Aqui, o aprendiz recebe apenas dados de treinamento não rotulados e faz previsões para pontos não vistos. Não há exemplos rotulados disponíveis para avaliar quantitativamente o desempenho do aprendiz. É usado em problemas de agrupamento e redução de dimensionalidade.

Aprendizado semi-supervisionado: O aprendiz recebe uma amostra de treinamento com dados rotulados e não rotulados e faz previsões para pontos não vistos. É útil quando há acesso fácil aos dados não rotulados, mas obter rótulos é caro. Pode ser aplicado em problemas de classificação, regressão ou ranking, buscando melhorar o desempenho em relação ao aprendizado supervisionado.

3 REVISÃO BIBLIOGRÁFICA

A Seção 3.1 apresenta a revisão da literatura voltada a detecção de intrusão no ambiente de contêiner, bem como suas limitações e desafios. A Seção 3.2 faz uma revisão geral sobre os principais pontos vistos em cada artigos presente na revisão bibliográfica.

3.1 DETECÇÃO DE INTRUSÃO EM CONTÊINERES

Em (Abed et al., 2015b), é proposta uma técnica baseada em chamadas de sistema para detectar comportamentos anômalos em aplicativos em contêineres Linux. O método envolve a criação de um modelo de comportamento normal do aplicativo com base nas chamadas de sistema que ele faz durante a execução normal. Quando o aplicativo se desvia desse modelo, a atividade é considerada suspeita e pode ser um comportamento malicioso.

No trabalho (Abed et al., 2015a) é apresentado um sistema de detecção de intrusão baseado em contêineres para aplicativos em execução no Linux. O IDS usa um modelo de perfil de comportamento do aplicativo para detectar atividades maliciosas em tempo real. O sistema coleta informações sobre as chamadas de sistema feitas pelo aplicativo em execução no contêiner e usa essas informações para criar um modelo de comportamento normal. Quando o aplicativo faz uma chamada de sistema que não está dentro do modelo normal, o IDS identifica essa atividade como suspeita e gera um alerta.

(Du et al., 2018) propõem uma abordagem para detectar e diagnosticar anomalias em microsserviços baseados em contêineres, usando monitoramento de desempenho. A abordagem usa uma combinação de análise de séries temporais e aprendizado de máquina para detectar anomalias em microsserviços. As métricas de desempenho dos microsserviços, como uso de CPU e memória, são coletadas e usadas para criar um modelo de comportamento normal do microsserviço. Quando o microsserviço se desvia do modelo, a atividade é considerada suspeita e pode ser uma anomalia. Também é apresenta um método para diagnosticar anomalias usando árvores de decisão. O sistema identifica quais métricas são mais relevantes para a anomalia em questão e fornece uma explicação para a causa da anomalia.

O trabalho de (Sultan et al., 2019) aborda os desafios e problemas de segurança relacionados ao uso de contêineres na computação em nuvem. Entre os principais desafios de segurança enfrentados pelas organizações que utilizam contêineres, estão a segurança do *host*, gerenciamento de identidade e acesso, segurança do registro de contêineres e segurança dos próprios contêineres. Algumas das soluções existentes para proteger contêineres são *namespaces*, *cgroups*, uso de plataformas confiáveis, entre outros. Outro aspecto importante para a proteção de contêineres é a colaboração entre as equipes de segurança e desenvolvimento para garantir a segurança ao longo do ciclo de vida dos contêineres.

Em (Zou et al., 2019) é apresentado um sistema de monitoramento de anomalias para contêineres Docker baseado na técnica de *Isolation Forest* otimizada. O objetivo é detectar anomalias em contêineres Docker em tempo real, garantindo a segurança e o desempenho do sistema. A técnica de *Isolation Forest* (iForest) é um método eficaz para detecção de anomalias em dados não supervisionados, podendo ser adaptada para o ambiente de contêineres Docker considerando as peculiaridades deste ambiente e pode ser usada para detectar anomalias em tempo real. Resultados de testes mostram que o sistema é capaz de detectar anomalias com alta precisão e eficiência em ambientes de contêineres Docker. Além disso, o sistema proposto apresenta desempenho superior em termos de precisão e eficiência, comparado com uma implementação baseada em iForest e outra baseada em densidade.

(Flora e Antunes, 2019) analisa a eficácia de diferentes abordagens de detecção de intrusão em ambientes de contêineres *multi-tenant*. O estudo avaliou três abordagens de detecção de intrusão: detecção de anomalias baseada em rede, detecção de anomalias baseada em comportamento e detecção de assinaturas. Os resultados mostraram que a detecção de anomalias baseada em comportamento foi a abordagem mais eficaz para detectar ataques em contêineres *multi-tenant*. O artigo também discute possibilidades para futuras pesquisas, como o uso de técnicas de aprendizado de máquina para aprimorar a detecção de anomalias em contêineres *multi-tenant*.

O artigo (Srinivasan et al., 2019) apresenta uma solução para a detecção de intrusões em tempo real em contêineres Docker. A arquitetura da solução proposta utiliza dados de monitoramento de contêineres para construir um modelo probabilístico baseado em redes Bayesianas. Esse modelo é treinado usando n-gramas de chamadas do sistema e a probabilidade de ocorrência desse n-grama é calculada e, posteriormente, é capaz de avaliar a probabilidade de comportamentos anormais em tempo real.

O estudo realizado por (Castanhel et al., 2020) aborda a detecção de anomalias em ambientes de contêineres, explorando o impacto da visão parcial das informações nos resultados. O objetivo é investigar como o tamanho da janela afeta a detecção de anomalias nesse contexto. Os contêineres são amplamente utilizados para virtualização de aplicações, e o conjunto de dados utilizado foi capturado em um ambiente de contêineres. Os dados consistem em chamadas e sinais emitidos pelos contêineres, representando comportamentos normais e anômalos que exploram vulnerabilidades de *Remote Code Execution* (RCE). Dois métodos de Classificação do tipo Classe Única (OCC) foram avaliados: *Isolation Forest* e *One-Class SVM*. Foram realizados experimentos com dois tipos de conjuntos de dados: um com dados brutos e outro com dados filtrados, removendo chamadas de sistema classificadas como de baixo nível de ameaça. A avaliação considerou o crescimento do tamanho da janela de treinamento e teste. Os resultados mostraram que tanto os dados brutos quanto os dados filtrados alcançaram resultados razoáveis com janelas pequenas, e houve variação nos resultados à medida que o tamanho da janela aumentou. O algoritmo *Isolation Forest* obteve melhores resultados conforme o tamanho da janela aumentou, superando o desempenho da *One-Class SVM*.

No trabalho subsequente, (Castanhel et al., 2021) houve foco em uma aplicação executada em um ambiente de contêiner e observada a partir do hospedeiro. Com o objetivo de explorar técnicas de machine learning para identificar ataques e entender como os sistemas de detecção de intrusão se comportam com essa perspectiva externa. Foram apresentadas três abordagens para coletar informações das chamadas de sistema dos processos dentro de um contêiner: (i) executar o Sistema de Detecção de Intrusão (IDS) no hospedeiro; (ii) usar uma aplicação de rastreio dentro do mesmo contêiner da aplicação; e (iii) utilizar um contêiner privilegiado para a execução do IDS. A primeira abordagem foi utilizada, coletando os dados das chamadas de sistema com a ferramenta *strace*. Foram realizados testes com um protótipo da proposta, utilizando o *Wordpress* como aplicação de teste devido à sua popularidade e ao fato de possuir diversas vulnerabilidades conhecidas. Foram testados diferentes tamanhos de janela e quatro algoritmos de classificação: *K-Nearest Neighbors* (KNN), *Random Forest* (RF), *Multilayer Perceptron* (MLP) e *AdaBoost* (AB). Os resultados foram avaliados utilizando métricas como precisão, *recall* e *f1-score*. Os resultados mostraram que a proposta é eficaz na detecção de anomalias.

(Shen et al., 2022) aborda a tecnologia de contêineres e sua vulnerabilidade a ataques de segurança, propondo uma nova estrutura de detecção de anomalias utilizando algoritmo de cluster. O artigo destaca que a detecção de intrusão baseada em anomalias é uma abordagem eficaz para a detecção de comportamentos anômalos de contêineres, e que o novo método proposto pode identificar recipientes construídos na mesma imagem de aplicação sem rotulagem manual, reduzindo a Taxa de Falso Positivo da detecção de anomalias e aumentando a Taxa de Positivo Verdadeiro em comparação com o método tradicional. A análise estática de imagens de contentores e a detecção de anomalias de um contentor em funcionamento também é discutida, sendo as principais abordagens utilizadas para a segurança de contentores atualmente.

A Tabela 3.1 mostra, em resumo, na coluna Estratégia utilizada, os métodos implementados para coletar os dados necessários para os testes, na coluna Ambiente, o modelo de contêiner de onde esses dados foram coletados e na coluna Dataset, qual o conjunto de dados utilizado para os testes, para cadas trabalho revisado na subseção anterior.

3.2 CONSIDERAÇÕES

A revisão bibliográfica apresentada aborda várias técnicas para a detecção de anomalias e intrusões em ambientes de contêineres Linux, destacando a importância da segurança nesse contexto. Foram apresentados sistemas de detecção baseados em chamadas de sistema, perfil de comportamento do aplicativo e monitoramento de desempenho, além de discussões sobre os desafios de segurança enfrentados pelas organizações que utilizam contêineres na computação em nuvem.

Os artigos apresentados também abordam a eficácia de diferentes abordagens de detecção de intrusão, como detecção de anomalias baseada em rede, comportamento e assinaturas, e destacam a importância do uso de técnicas de aprendizado de máquina para aprimorar a detecção

Trabalho	**Estratégia utilizada**	**Ambiente**	**Dataset**
(Abed et al., 2015b)	Strace	Contêiner Docker	-
(Du et al., 2018)	cAdvisor, Heapster, InfluxDB e Grafana	Contêiner Kubernetes	Dados próprios
(Zou et al., 2019)	Ganglia, Nagios, Akshay, cAdviosr e InfluxDB	Contêiner Docker	-
(Flora e Antunes, 2019)	Strace e sysdig	Contêiner (Docker e LXC)	-
(Sultan et al., 2019)	Starlight, OpenWhisk	Contêiner Docker	-
(Srinivasan et al., 2019)	N-grams para análise probabilística	Contêiner Docker	Dataset UNM
(Castanhel et al., 2021)	Strace	Contêiner Docker	Dados próprios
(Rocha et al., 2022)	Sysdig	Contêiner Kubernetes	Dados próprios
(Shen et al., 2022)	Sysdig	Contêiner (Docker e LXC)	-

Tabela 3.1: Trabalhos relacionados

de anomalias em contêineres multi-inquilinos. Após analise de todos os trabalhos, também ficou claro a dificuldade em encontras os dataset utilizados para os testes. Conjunto de dados disponíveis são limitados, ao considerar o ponto de observação a perspectiva de observação dos contêineres acaba sendo voltada para coleta de informações no nível do sistema hospedeiro e não do contêiner (Du et al., 2018; Rocha et al., 2022; Castanhel et al., 2021).

4 PROPOSTA

O capítulo 4 apresenta a proposta da pesquisa. A Seção 4.1 abrange as problemáticas na área. A Seção 4.2 aborda a estratégia proposta. A Seção 4.3 apresenta o processo de avaliação a ser realizado. Por fim, na seção 4.4 é apresentado o cronograma para esse trabalho.

4.1 PROBLEMÁTICA

O problema central abordado neste trabalho é a falta de uma solução abrangente e eficaz para a detecção de anomalias e intrusões em contêineres Docker. As soluções tradicionais de HIDS se concentram em monitorar as atividades do sistema operacional hospedeiro, sem levar em consideração o ambiente contido no contêiner em si. Isso limita a capacidade de detecção de ameaças específicas aos contêineres e torna a resposta a incidentes menos precisa.

Além disso, a maioria das soluções com base em HIDS estão limitadas ao uso de chamadas de sistema para detecção de intrusões. Estas chamadas são coletadas sem considerar o impacto que o ambiente de contêineres Docker pode estar tendo no contêiner em execução. As características únicas dos contêineres, como o uso de *namespaces* para o isolamento de recursos, requerem uma abordagem específica para a detecção de anomalias que considere o ambiente de isolamento.

Uma ferramenta conhecida para coleta de informações por chamadas de sistema é o *strace*, que utiliza a função *ptrace*, fornecida pelo Linux e outros sistemas operacionais, para instrumentar um processo-alvo e "escutar" as chamadas de sistema desse processo. No momento que uma chamada de sistema é invocada, o *strace* interrompe o processo rastreado, captura a chamada, decodifica-a e, em seguida, retoma a execução do processo, como mostrado na figura 4.1. No entanto, essa abordagem possui uma desvantagem significativa: cada mudança de contexto original é transformada em várias mudanças de contexto, o que resulta em um desempenho ineficiente e um processo rastreado aguardando o *strace* para realizar sua decodificação, (Sysdig, 2014).

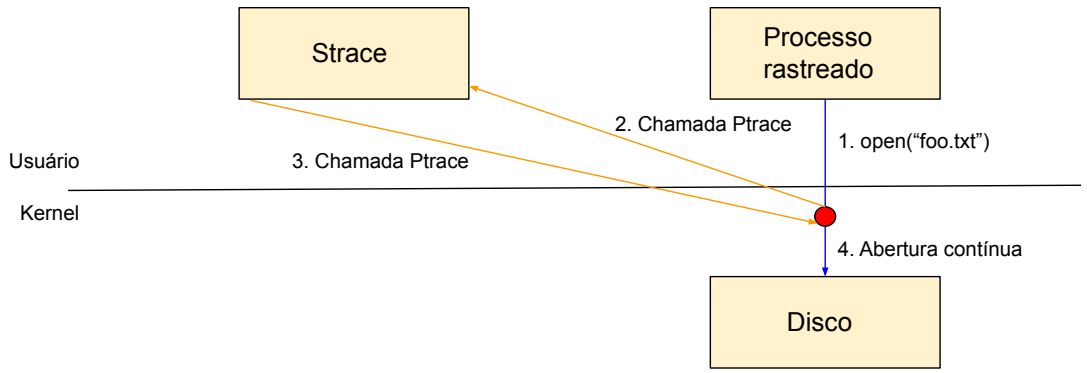

Figura 4.1: Interação do *strace*, com base em (Sysdig, 2014).

Por outro lado, o *sysdig* adota uma arquitetura diferente. Ele utiliza um *driver* chamado *sysdig-probe*, que captura eventos no *kernel* por meio de uma funcionalidade chamada *tracepoints*. Esses *tracepoints* permitem que um "*handler*" seja instalado e chamado em funções específicas do *kernel*. O *handler* do *sysdig-probe* é simples e limitado a copiar os detalhes do evento em um *buffer* compartilhado, codificado para consumo posterior. Os eventos são então mapeados em espaço de usuário, permitindo acesso direto ao *buffer* sem cópias adicionais, minimizando o uso da CPU e as perdas de cache, como mostrado na figura 4.2. Duas bibliotecas, `libscap` e `libsinsp`, fornecem suporte para leitura, decodificação e análise desses eventos. O *sysdig* age como uma camada de abstração simples em torno dessas bibliotecas, (Sysdig, 2014).

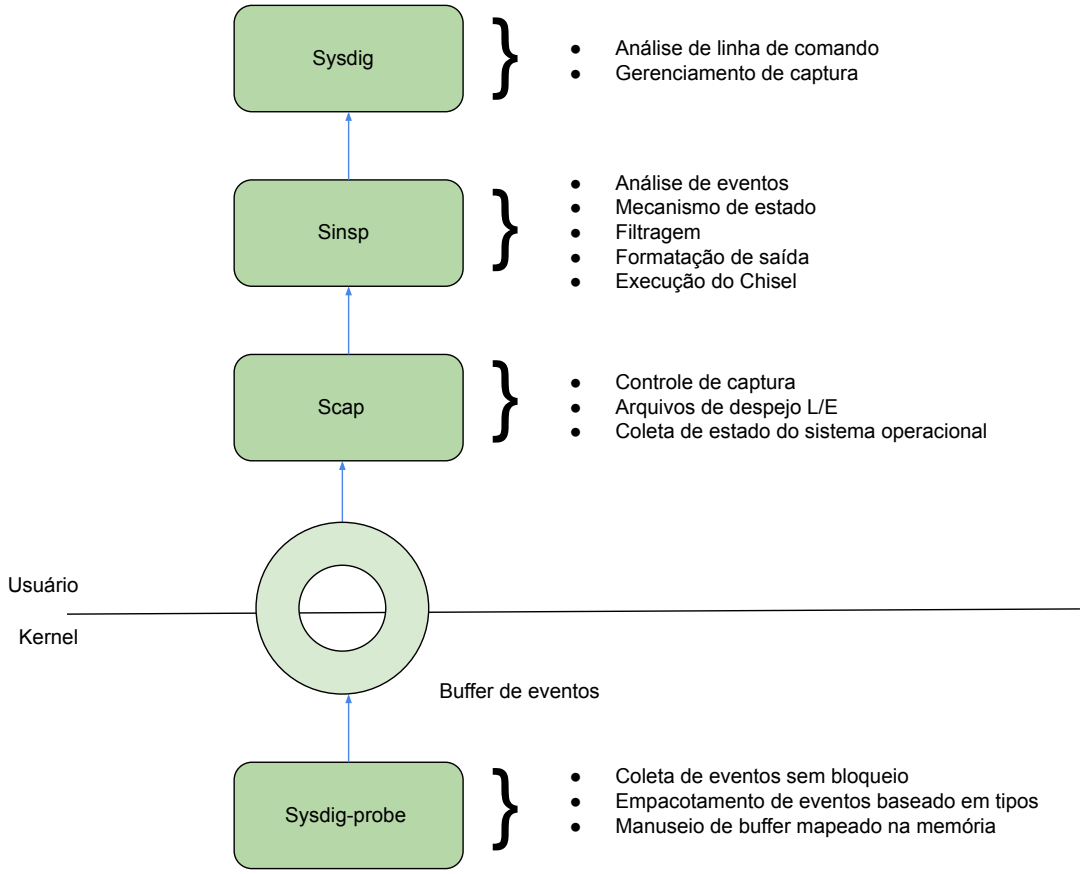

Figura 4.2: Interação do Sysdig, com base em (Sysdig, 2014).

O Strace é conhecido por ter um overhead significativo, pois sua abordagem envolve a interceptação de todas as chamadas de sistema de um processo e a exibição dos detalhes para fins de depuração. Isso pode resultar em uma degradação perceptível do desempenho do sistema, especialmente quando usado em processos intensivos em I/O (entrada/saída) (Sysdig, 2014). Por outro lado, o Sysdig utiliza uma abordagem diferente para monitorar chamadas de sistema. Ele faz uso de uma funcionalidade do kernel Linux chamada *extended Berkeley Packet Filter* (eBPF) (Sysdig, 2019) para coletar dados em tempo real sobre as atividades do sistema. Essa abordagem é mais eficiente e geralmente tem um impacto menor no desempenho.

4.2 ESTRATÉGIA

A proposta deste trabalho é desenvolver um HIDS especializado para a detecção de anomalias e intrusões em contêineres Docker. Através do uso de chamadas de sistemas e características internas do contêiner, o isolamento da aplicação será levado em consideração e a coleta das informações será realizada em um nível próximo da aplicação para que não seja captura informações que não pertencem ao contêiner. A abordagem levará em consideração as características específicas dos contêineres Docker, como o isolamento de recursos e o uso de *namespaces*, para garantir uma detecção precisa e eficiente.

A solução proposta terá as seguintes etapas principais:

1. Coleta de dados: será realizada a coleta das chamadas de sistema realizadas dentro do contêiner Docker, capturando informações relevantes sobre as atividades em execução.

2. Pré-processamento: Os dados coletados serão pré-processados para extrair características relevantes e reduzir o ruído, garantindo que apenas informações significativas sejam consideradas no processo de detecção.

3. Treinamento: Utilizando técnicas de aprendizado de máquina, o HIDS será treinado em um conjunto de dados anotados, onde serão identificados os padrões normais de comportamento do contêiner.

4. Detecção de anomalias: Com base no modelo treinado, o HIDS será capaz de detectar anomalias e intrusões no comportamento do contêiner em tempo real. Qualquer desvio significativo em relação aos padrões estabelecidos será considerado uma possível ameaça.

4.2.1 Coleta de Dados

Nesta subseção, é descrita a abordagem utilizada para coletar os dados relevantes. Para essa finalidade, foram empregadas as seguintes ferramentas:

Elasticsearch/Logstash: O Elasticsearch é um mecanismo de busca e análise de dados em tempo real, enquanto o Logstash é um pipeline de processamento de dados que permite a ingestão, transformação e envio desses dados para o Elasticsearch (Elastic, 2023b,a).

O Logstash é usado para extrair os dados brutos de diversas fontes, como registros de eventos, *logs* de sistemas e métricas de desempenho. Esses dados são então normalizados e enriquecidos, a fim de garantir sua integridade e consistência. Em seguida, os dados processados são armazenados no Elasticsearch para permitir uma consulta eficiente e flexível durante as etapas de análise subsequentes.

Fluentd: É um sistema de agregação de registros de código aberto que oferece uma arquitetura flexível e escalável para a coleta de logs em tempo real.

Com o Fluentd, é possível coletar dados de várias fontes distribuídas em diferentes componentes do sistema. Ele facilita a integração com diferentes formatos de *logs* e protocolos de comunicação, permitindo obter dados de maneira uniforme e centralizada, (Fluentd, 2023).

SysDig: é uma ferramenta de monitoramento de contêineres e sistemas operacionais que fornece uma visão detalhada do desempenho e do comportamento do sistema, (Sysdig, 2017).

Através do SysDig, pode-se capturar métricas essenciais do sistema, como utilização de CPU, memória e E/S de disco, além de informações relacionadas a eventos e chamadas do kernel.

4.2.2 Métodos de Filtragem

Será aplicada uma filtragem por análise de padrões envolvendo a identificação de padrões nos dados obtidos das chamadas de sistema e a seleção dos registros que se enquadram nesses padrões. Serão aplicadas técnicas de mineração de dados, como algoritmos de aprendizado de máquina, para identificar padrões relevantes nos dados e filtrar aqueles que não se encaixam nesses padrões. Isso auxiliará na detecção de comportamentos anômalos ou indesejados nas chamadas de sistema.

4.3 MÉTODOS DE AVALIAÇÃO

Com o intuito de avaliar uma técnica viável para detectar anomalias com base em chamadas de sistema, a abordagem proposta concentra-se em uma aplicação em execução em um contêiner. O objetivo é investigar o uso de técnicas de aprendizado de máquina para identificar ataques e compreender o comportamento dos sistemas de detecção de intrusão nessa perspectiva.

4.4 CRONOGRAMA

A seguir estão é apresentado as tarefas que serão realizadas. A Tabela 4.1 apresenta o cronograma completo no calendário.

1. Definição do tópico de pesquisa;
2. Estudo de ferramentas para a extração de características de contêineres;
3. Levantamento da literatura relacionado à detecção de anomalias no ambiente de contêiner;
4. Escrita do TCC1 para apresentação da proposta;

5. Definição do conjunto de dados que será utilizado para a avaliação;
6. Coleta e Treinamento dos modelos de aprendizado de máquina que serão utilizados para a detecção de anomalia;
7. Avaliação dos experimentos realizados; e
8. Escrita do TCC2.

Fases	**2023**									
	M	A	M	J	J	A	S	O	N	D
1	**X**	**X**	**X**	**X**						
2	**X**	**X**	**X**	**X**						
3			**X**	**X**	**X**					
4				**X**	**X**	**X**				
5					**X**	**X**	**X**			
6							**X**	**X**		
7								**X**	**X**	
8								**X**	**X**	**X**

Tabela 4.1: Cronograma da pesquisa

REFERÊNCIAS

Abed, A. S., Clancy, C. e Levy, D. S. (2015a). Intrusion detection system for applications using linux containers. Em *Security and Trust Management: 11th International Workshop, STM 2015, Vienna, Austria, September 21-22, 2015, Proceedings 11*, páginas 123–135. Springer.

Abed, A. S., Clancy, T. C. e Levy, D. S. (2015b). Applying bag of system calls for anomalous behavior detection of applications in linux containers. Em *2015 IEEE globecom workshops (GC Wkshps)*, páginas 1–5. IEEE.

Ahmed, M., Mahmood, A. N. e Hu, J. (2016). A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60:19–31.

Amazon (2023). Integridade de instâncias do contêiner. `https://docs.aws.amazon.com/pt_br/AmazonECS/latest/developerguide/container-instance-health.html`.

Bridges, R. A., Glass-Vanderlan, T. R., Iannacone, M. D., Vincent, M. S. e Chen, Q. (2019). A survey of intrusion detection systems leveraging host data. *ACM Computing Surveys (CSUR)*, 52(6):1–35.

Carbonell, J. G., Michalski, R. S. e Mitchell, T. M. (1983). An overview of machine learning. *Machine learning*, páginas 3–23.

Castanhel, G. R., Heinrich, T., Ceschin, F. e Maziero, C. (2021). Taking a peek: An evaluation of anomaly detection using system calls for containers. Em *2021 IEEE Symposium on Computers and Communications (ISCC)*, páginas 1–6. IEEE.

Castanhel, G. R., Heinrich, T., Ceschin, F. e Maziero, C. A. (2020). Sliding window: The impact of trace size in anomaly detection system for containers through machine learning. Em *Anais da XVIII Escola Regional de Redes de Computadores*, páginas 141–146. SBC.

Ceschin, F., Gomes, H. M., Botacin, M., Bifet, A., Pfahringer, B., Oliveira, L. S. e Grégio, A. (2020). Machine learning (in) security: A stream of problems. *arXiv preprint arXiv:2010.16045*.

Chandola, V., Banerjee, A. e Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58.

Docker (2023). Use containers to build, share and run your applications. `https://www.docker.com/resources/what-container/`.

Du, Q., Xie, T. e He, Y. (2018). Anomaly detection and diagnosis for container-based microservices with performance monitoring. Em *Algorithms and Architectures for Parallel Processing: 18th International Conference, ICA3PP 2018, Guangzhou, China, November 15-17, 2018, Proceedings, Part IV 18*, páginas 560–572. Springer.

Eder, M. (2016). Hypervisor-vs. container-based virtualization. *Future Internet (FI) and Innovative Internet Technologies and Mobile Communications (IITM)*, 1.

Elastic (2023a). Centralize, transforme e oculte seus dados. `https://www.elastic.co/pt/logstash/`.

Elastic (2023b). O que é o elasticsearch? `https://www.elastic.co/pt/what-is/elasticsearch`.

Eskin, E. (2000). Anomaly detection over noisy data using learned probability distributions.

Flora, J. e Antunes, N. (2019). Studying the applicability of intrusion detection to multi-tenant container environments. Em *2019 15th European Dependable Computing Conference (EDCC)*, páginas 133–136. IEEE.

Fluentd (2023). O que é fluentd? `https://www.fluentd.org/architecture`.

Hodge, V. e Austin, J. (2004). A survey of outlier detection methodologies. *Artificial intelligence review*, 22:85–126.

Joy, A. M. (2015). Performance comparison between linux containers and virtual machines. Em *2015 international conference on advances in computer engineering and applications*, páginas 342–346. IEEE.

Kubernetes (2023). Kubernetes components. `https://kubernetes.io/docs/concepts/overview/components/`.

Lakhina, A., Crovella, M. e Diot, C. (2004). Diagnosing network-wide traffic anomalies. *ACM SIGCOMM computer communication review*, 34(4):219–230.

Laureano, M. A. P., Maziero, C. A. e Jamhour, E. (2003). Detecção de intrusão em máquinas virtuais. *5º Simpósio de Segurança em Informática–SSI. São José dos Campos*, páginas 1–7.

Liao, H.-J., Lin, C.-H. R., Lin, Y.-C. e Tung, K.-Y. (2013). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24.

Lin, X., Lei, L., Wang, Y., Jing, J., Sun, K. e Zhou, Q. (2018). A measurement study on linux container security: Attacks and countermeasures. Em *Proceedings of the 34th Annual Computer Security Applications Conference*, páginas 418–429.

Liu, M., Xue, Z., Xu, X., Zhong, C. e Chen, J. (2018). Host-based intrusion detection system with system calls: Review and future trends. *ACM Computing Surveys (CSUR)*, 51(5):1–36.

Lopez, M., Figueiredo, U., Lobato, A. e Duarte, O. C. (2014). Broflow: Um sistema eficiente de detecção e prevenção de intrusão em redes definidas por software. Em *Anais do XIII Workshop em Desempenho de Sistemas Computacionais e de Comunicação*, páginas 108–121. SBC.

Martínez-Magdaleno, S., Morales-Rocha, V. e Parra, R. (2021). A review of security risks and countermeasures in containers. *International Journal of Security and Networks*, 16(3):183–190.

Merkel, D. et al. (2014). Docker: lightweight linux containers for consistent development and deployment. *Linux j*, 239(2):2.

Mohri, M., Rostamizadeh, A. e Talwalkar, A. (2018). *Foundations of machine learning*. MIT press.

Nasteski, V. (2017). An overview of the supervised machine learning methods. *Horizons. b*, 4:51–62.

RedHat (2023a). Docker: desenvolvimento de aplicações em containers. `https://www.redhat.com/pt-br/topics/containers/what-is-docker`.

RedHat (2023b). Kubernetes. `https://www.redhat.com/pt-br/topics/containers/what-is-kubernetes`.

Rocha, S. L., Nze, G. D. A. e de Mendonça, F. L. L. (2022). Intrusion detection in container orchestration clusters: A framework proposal based on real-time system call analysis with machine learning for anomaly detection. Em *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*, páginas 1–4. IEEE.

Röhling, M. M., Grimmer, M., Kreubel, D., Hoffmann, J. e Franczyk, B. (2019). Standardized container virtualization approach for collecting host intrusion detection data. Em *2019 Federated Conference on Computer Science and Information Systems (FedCSIS)*, páginas 459–463. IEEE.

Shen, J., Zeng, F., Zhang, W., Tao, Y. e Tao, S. (2022). A clustered learning framework for host based intrusion detection in container environment. Em *2022 IEEE International Conference on Communications Workshops (ICC Workshops)*, páginas 409–414. IEEE.

Srinivasan, S., Kumar, A., Mahajan, M., Sitaram, D. e Gupta, S. (2019). Probabilistic real-time intrusion detection system for docker containers. Em *Security in Computing and Communications: 6th International Symposium, SSCC 2018, Bangalore, India, September 19–22, 2018, Revised Selected Papers 6*, páginas 336–347. Springer.

Sturm, R., Pollard, C. e Craig, J. (2017). *Application performance management (APM) in the digital enterprise: managing applications for cloud, mobile, iot and eBusiness*. Morgan Kaufmann.

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

Sysdig (2014). Sysdig vs dtrace vs strace: A technical discussion. `https://sysdig.com/blog/sysdig-vs-dtrace-vs-strace-a-technical-discussion/`.

Sysdig (2017). Visão geral do sysdig. `https://github.com/draios/sysdig/wiki/Sysdig-Overview`.

Sysdig (2019). Sysdig e falco agora alimentados por ebpf. `https://sysdig.com/blog/sysdig-and-falco-now-powered-by-ebpf/`.

Thottan, M. e Ji, C. (2003). Anomaly detection in ip networks. *IEEE Transactions on signal processing*, 51(8):2191–2204.

Vinchurkar, D. P. e Reshamwala, A. (2012). A review of intrusion detection system using neural network and machine learning. *J. Eng. Sci. Innov. Technol*, 1:54–63.

Xavier, M. G., Neves, M. V., Rossi, F. D., Ferreto, T. C., Lange, T. e De Rose, C. A. (2013). Performance evaluation of container-based virtualization for high performance computing environments. Em *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, páginas 233–240. IEEE.

Zou, Z., Xie, Y., Huang, K., Xu, G., Feng, D. e Long, D. (2019). A docker container anomaly monitoring system based on optimized isolation forest. *IEEE Transactions on Cloud Computing*, 10(1):134–145.