Detecção de Intrusão em Clusters de Orquestração de Contêineres

Proposta de um framework baseado na análise em tempo real de system calls com machine learning para detecção de anomalias

Intrusion Detection in Container Orchestration Clusters

A framework proposal based on real-time system call analysis with machine learning for anomaly detection

Sávio Levy Rocha[1], Georges Daniel Amvame Nze[2] e
Fábio Lucio Lopes de Mendonça[3]
Programa de Pós-Graduação Profissional em Engenharia Elétrica (PPEE), Departamento de Engenharia Elétrica,
Universidade de Brasília (UnB)
Brasília-DF, Brasil
[1]slevyr@gmail.com, [2]georges@unb.br,
[3]fabio.mendonca@redes.unb.br

Resumo — **Apesar dos benefícios trazidos pelo uso de contêineres, ameaças e riscos de ataques voltados a essa tecnologia têm crescido em igual proporção à sua adoção. Sistemas de Detecção de Intrusão (IDS) têm sido empregados visando garantir a segurança de ambientes de nuvem e contêineres. Contudo, as características inerentes a esses ambientes têm apresentado novos desafios para garantir um nível de segurança adequado. Neste trabalho, um *framework* é proposto para implementação de um Sistema de Detecção de Intrusão baseado em *Host* (HIDS), através da análise de *system calls* com *machine learning* em um *cluster* de orquestração de contêineres Kubernetes. O framework apresentado possibilita a desoneração dos nós do *cluster* do processamento voltado para a detecção de intrusão através de uma arquitetura distribuída e escalável. Alertas gerados na ocorrência de anomalias detectadas poderão ser utilizados como fonte de informação complementar para a tomada de decisão e atuação da equipe do Centro de Operações de Segurança (SOC) para tratar um eventual incidente de segurança. A arquitetura proposta foi implementada no *software* GNS3, emulando um ambiente de rede corporativo para demonstrar a viabilidade de implementação do *framework* em um ambiente real.**

Palavras Chave – detecção de intrusão; HIDS; system calls; contêineres; SOC.

Abstract — **Despite the benefits containerization brings, threats and risks of attacks against containerized technology have grown in equal proportion to its adoption. Intrusion Detection Systems (IDS) have been employed to secure cloud and container environments. However, the inherent characteristics of these environments have presented new challenges to ensuring an adequate level of security. In this paper, a framework is proposed for implementing a Host-based Intrusion Detection System (HIDS) by analyzing system calls with machine learning on a Kubernetes container orchestration cluster. The presented framework prevents the overhead of the cluster nodes from processing focused on intrusion detection through a distributed and scalable architecture. Alerts generated in the occurrence of detected anomalies can be used as a complementary source of information for decision making and action by the Security Operations Center (SOC) team to deal with an eventual security incident. The proposed architecture was implemented in the GNS3 software, emulating a corporate network environment to demonstrate the feasibility of implementing the framework in a real environment.**

Keywords – intrusion detection; HIDS; system calls; containers; SOC.

I. INTRODUÇÃO

Tecnologias de contêineres modernas surgiram em grande parte juntamente com a adoção de práticas de Desenvolvimento de *software* e Operações e TI (DevOps), que procuram aumentar a integração entre a construção e a execução de aplicações [1]. Ferramentas conhecidas como orquestradores de contêiner permitem às equipes de DevOps ou automações construídas para tal finalidade, fazer o *download* de imagens de registros, colocar essas imagens em contêineres, e gerenciar os contêineres em execução [1]. Infraestruturas de computação em nuvem baseadas em contêiner têm emergido devido às suas características, provendo aos usuários serviços sob demanda e com elasticidade, acessíveis de qualquer lugar [2] [3]. Uma vez que os ambientes de nuvem são ambientes *multi-tenant*,

2022 17th Iberian Conference on Information Systems and Technologies (CISTI)
22 – 25 June 2022, Madrid, Spain
ISBN: 978-989-33-3436-2

Authorized licensed use limited to: UNIVERSIDADE DE BRASILIA. Downloaded on January 25,2023 at 18:04:07 UTC from IEEE Xplore. Restrictions apply.

apoiando-se em tecnologias como *cgroups* e *namespaces* para proporcionar o isolamento dos contêineres, a superfície de ataque das aplicações implantadas, aumentou substancialmente nesses ambientes [3]. Por conseguinte, existe uma grande necessidade de implantar contramedidas centradas em contêineres nessas infraestruturas [3].

Um dos recursos mais utilizados visando garantir um nível de segurança adequado aos ambientes computacionais modernos é a utilização de Sistemas de Detecção de Intrusão (IDS, do termo em inglês). A detecção de intrusão é o processo de monitoramento dos eventos que ocorrem em um sistema de computador ou rede, analisando-os em busca de indícios de possíveis incidentes [4]. Um HIDS (*Host-based* IDS) analisa diversos recursos em um *host* como tráfego de rede, atividades no sistema de arquivos, *logs* de aplicação e *system calls* [4]. Em Sistemas Operacionais (SO) baseados em Unix, todas as aplicações que necessitam de recursos do SO devem fazer uso de *system calls*, por esse motivo HIDS baseados em *system calls* conseguem obter a melhor granularidade de dados [5]. Da mesma forma, contêineres Linux usualmente comunicam com o *kernel* do Sistema Operacional do *host* através de *system calls* [8].

HIDS baseados em anomalia inicialmente constroem um perfil normal de *system calls* utilizadas e processos que não se enquadram no perfil normal construído são considerados como intrusivos [6]. Sendo assim, a maioria dos HIDS propostos na literatura são baseados em anomalia [6]. HIDS baseados em *system calls* têm ganhado atenção nos últimos vinte anos devido ao aumento crescente de ataques voltados para servidores Linux e têm sido desenvolvidos para detecção de intrusão em *hosts* virtuais e sistemas embarcados [7].

Neste trabalho, será realizada a proposição de um *framework* que possibilita a implementação de um HIDS baseado em anomalias de *system calls*, especificamente em ambientes que fazem uso de plataformas para execução de contêineres. Através desse *framework*, espera-se contribuir com a resolução de problemas existentes na área como: a necessidade de aprimoramento da segurança em ambientes de contêineres, a sobrecarga causada nas plataformas de contêineres pelos HIDS, a ausência de *datasets* atuais, e a pouca ênfase dada à detecção de intrusão em tempo real.

O restante deste trabalho está organizado da seguinte forma: considerações sobre trabalhos relacionados são realizadas na seção II. Na seção III, o *framework* proposto é apresentado. Por fim, conclusões e indicações de trabalhos futuros são realizadas na seção IV.

II. Trabalhos Relacionados

De acordo com [3] e [9], apesar dos avanços em termos de IDS para máquinas virtuais, as contribuições para abordagens de sistemas de detecção voltados para contêiner são limitadas e poucos esforços têm sido direcionados para a área de segurança de contêineres em nuvem.

Entre os trabalhos mais recentes sobre HIDS baseados em anomalias em *system calls*, muitas são as abordagens utilizadas para identificação de intrusões. A detecção de intrusão através de sequências de *system calls* foi apresentada em 1996 e ainda é utilizada em conjunto com técnicas de análise probabilísticas em trabalhos como [10], bem como redes neurais em [12]. Outros trabalhos como [5], [6], [8] e [13] fazem uso de abordagens baseadas na análise de frequência de *system calls*, e a utilização conjunta com redes neurais também é possível, conforme [14]. De acordo com [21], diversos algoritmos e técnicas de *machine learning* têm sido utilizados para detecção de anomalias. Ademais, abordagens de detecção baseadas em Redes Neurais Artificiais (ANN) têm apresentado resultados promissores, e portanto, têm fomentado muitos estudos recentemente.

Conforme [8], devido ao rápido desenvolvimento de técnicas e instalações de *data centers*, recentemente, os HIDS têm sofrido do bem conhecido desafio de *Big Data*. Dessa forma, métodos de mineração e sistemas de gerenciamento de banco de dados tradicionais em um *host* único podem não ser capazes de lidar com a quantidade massiva de *system calls* de maneira eficiente [8]. Ainda, a maioria dos HIDS tradicionais executam a análise de intrusão em um *host* independente com um *software* de detecção *standalone* instalado e não há interação entre os HIDS instalados nos diferentes nós de um *cluster* [8].

Um problema recorrente e citado em grande número de trabalhos como [5], [6], [7], [10], [11] e [12], diz respeito à sobrecarga de processamento ligada à detecção de anomalias através de *system calls*. O grande volume de dados e a complexidade das técnicas envolvidas no processo de análise, muitas vezes acarreta em uma penalidade de desempenho dos sistemas monitorados ou compromete a eficácia da detecção.

Outra limitação existente na área envolve a defasagem dos datasets de *system calls* disponíveis atualmente. De acordo com [11], [12] e [9], os principais *datasets* conhecidos apresentam inúmeras deficiências tais como: obsolescência por antiguidade, falta de volume, ausência de dados complementares, além de a maioria não ser voltada para *system calls* de contêineres. No intuito de sanar essa lacuna, [9] propõe uma metodologia para geração de novos *datasets* que podem ser úteis no contexto da detecção de intrusão.

No que tange a cenários onde plataformas de orquestração de contêineres são utilizadas como ambientes de produção, poucos são os estudos de implementação de HIDS baseados em anomalias de *system calls* desenvolvidos até o momento. No trabalho de [14], um *framework* para aprendizado distribuído foi desenvolvido visando a construção de modelos de detecção por aplicação através de redes neurais. Contudo, é sabido que o sistema implementado em cada nó da plataforma de contêineres gera uma sobrecarga que compete com a carga de trabalho real de aplicações e essa sobrecarga não foi considerada. Em [15], foi proposto um HIDS voltado para um *cluster* Kubernetes com detecção de anomalias através de redes neurais com aprendizado supervisionado e quatro categorias de *system calls*. Apesar do sistema ser capaz de monitorar os diversos nós do *cluster* e realizar a detecção em componente externo, as regras de filtragem a partir de um conjunto limitado de *system calls* podem restringir o escopo de detecção de ataques. Outro aspecto diz respeito à limitação acerca da falta de análise das anomalias reportadas e a necessidade de desenvolvimento de subcomponentes próprios como um portal

2022 17th Iberian Conference on Information Systems and Technologies (CISTI)
22 – 25 June 2022, Madrid, Spain
ISBN: 978-989-33-3436-2

Authorized licensed use limited to: UNIVERSIDADE DE BRASILIA. Downloaded on January 25,2023 at 18:04:07 UTC from IEEE Xplore. Restrictions apply.

web e um serviço de API (*Application Programming Interface*) *Restful* para implementação do sistema de detecção de intrusão.

III. Framework Proposto

Diversos estudos têm sido conduzidos visando o aprimoramento dos HIDS baseados em anomalias em *system calls*. Contudo, muitos desafios ainda se apresentam sem uma solução definitiva e permanecem não completamente explorados. Nesse sentido, o *framework* proposto aborda alguns dos desafios expostos pelos trabalhos relacionados e visa o aprimoramento dos sistemas de detecção de intrusão em plataformas de contêineres.

Para implementação do *framework*, o *software* emulador de redes GNS3 [16] foi utilizado na versão 2.2.29. Uma topologia de um ambiente corporativo foi então construída contendo um *cluster* de orquestração de contêineres Kubernetes [17] na versão 1.18, diversas *vlans*, um *firewall*, roteadores e *switches*.

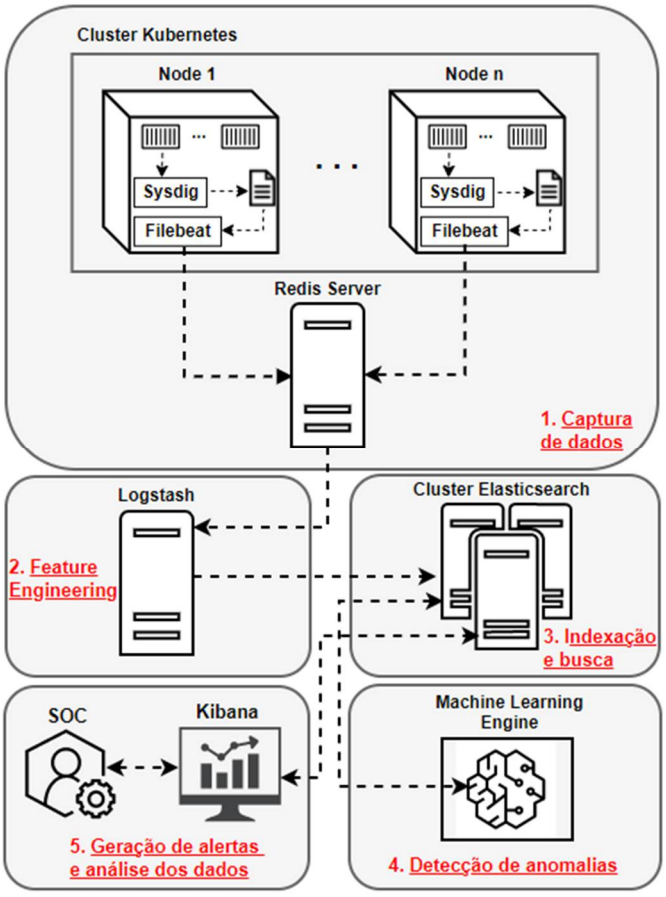

Figura 1: Arquitetura proposta com cinco camadas e ferramentas.

A arquitetura proposta apresenta um HIDS com componentes distribuídos, viabilizando a elasticidade do sistema conforme a necessidade. O fluxo de dados do sistema de detecção compreende cinco camadas, onde ferramentas são utilizadas para propósitos específicos, conforme a Fig. 1. O fluxo tem início através de um agente localizado em cada nó do *cluster* Kubernetes. Esse agente faz a coleta das *system calls* dos contêineres desejados através da ferramenta Sysdig [18] e as grava em um arquivo localmente, evitando a dependência de serviços externos na camada de captura de dados. À medida que as *system calls* são gravadas em arquivo, o serviço Filebeat [19] faz a leitura de cada nova linha adicionada em arquivo e as envia para o Redis [20], um banco de dados em memória externo ao *cluster*. O Redis age como um *cache*, dando resiliência e alto desempenho ao fluxo de *system calls* capturadas em tempo real dos contêineres em execução. O Logstash [19] é responsável por consumir os dados do Redis e fazer a filtragem e tratamento dos dados brutos a partir das *system calls* (*feature engineering*). Depois do tratamento, as *system calls* são indexadas no Elasticsearch [19] em índices que podem ser criados dinamicamente conforme o número de documentos indexados ou em intervalos de tempo pré-definidos. Os índices no Elasticsearch contém os *datasets* com as *system calls* e as suas *features* já codificadas em formato numérico, próprio para o processamento feito pelos algoritmos de *machine learning*.

O Elasticsearch é um motor de indexação e busca de alto desempenho [19]. Sua utilização como banco de dados para armazenamento das *system calls* permite que haja alta disponibilidade dos dados, além de garantir a consulta através de meta-dados úteis para filtragens, agregações, ordenações, etc. Além de conter os índices criados pelo Logstash representando os *datasets*, o Elasticsearch integra com o módulo de *machine learning* e também é utilizado para armazenar os resultados das análises de anomalias. Para cada índice de *dataset* existente, após sua análise, um novo índice é criado no Elasticsearch com o escore de anomalia correspondente às janelas de *system calls*. As janelas de *system calls* consistem em uma estrutura da dados que armazenam sequências de *system calls* de múltiplos tamanhos e é uma abordagem comum para análise de *system calls*.

Na camada de detecção de anomalias, o módulo de *machine learning* representado na arquitetura pode executar diversos algoritmos do estado da arte para detecção de anomalias. O requisito existente é que haja uma forma de integração entre a ferramenta utilizada e o Elasticsearch para leitura e gravação de dados. Para ambientes baseados em Python, já existe uma biblioteca que faz a integração com o Elasticsearch. Desse modo, é possível que vários algoritmos e ferramentas de *machine learning* sejam utilizados no *framework* visando obter o melhor resultado na detecção de anomalias. Na última camada do fluxo dos dados, o Kibana [19] é uma importante ferramenta de auxílio à equipe do SOC (Centro de Operações de Segurança) no acompanhamento de eventos de segurança. O Kibana tem integração nativa com o Elasticsearch e fornece uma interface gráfica para análise tanto dos índices de *datasets* quanto de anomalias gerados pelo HIDS. Esse recurso permite que a análise de janelas de *system calls* com escore anômalo sejam investigadas e assinaturas de ataques através de sequências de *system calls* sejam identificadas. Por fim, mas não de maneira exaustiva, através do Kibana é possível que *dashboards* sejam criados com métricas e alertas customizados na ocorrência de anomalias, tendo seu lugar no painel de monitoramento do SOC para correlação com outros eventos de segurança gerados por outras ferramentas.

Devido à impossibilidade de incluir todas as imagens neste documento, o repositório [22] disponível no Github, contém imagens que demonstram a topologia implementada no GNS3,

ISBN: 978-989-33-3436-2

Authorized licensed use limited to: UNIVERSIDADE DE BRASILIA. Downloaded on January 25,2023 at 18:04:07 UTC from IEEE Xplore. Restrictions apply.

além de um exemplo de funcionamento do *pipeline* e integração entre as camadas do *framework* no monitoramento de um contêiner de teste.

IV. CONCLUSÕES

O *framework* proposto apresenta um HIDS baseado em anomalias de *system calls* para plataformas de orquestração contêineres. De maneira resumida, foi apresentada uma arquitetura que implementa um *pipeline* composto de cinco camadas endereçando algumas das limitações existentes na área como: a necessidade de aprimoramento da segurança em ambientes de contêineres, a sobrecarga causada nas plataformas de contêineres pelos HIDS; a ausência de datasets atuais; e a pouca ênfase dada à detecção de intrusão em tempo real. Demonstrando sua viabilidade de implementação e funcionamento, o *framework* foi construído em um cenário equivalente aos ambientes corporativos encontrados *on-premises* ou em nuvem. Como principais contribuições esperadas do *framework*, destaca-se:

- Realização da coleta remota de *system calls* em diferentes nós de um *cluster* de orquestração de contêineres, possibilitando o aprendizado de anomalias de maneira colaborativa;
- Redução da sobrecarga de processamento na plataforma de contêineres através de uma arquitetura de HIDS com componentes distribuídos;
- Arquitetura escalável implementada com ferramentas gratuitas em um ambiente corporativo emulado.
- Construção de *dataset* próprio e possibilidade de compartilhamento com a comunidade;
- Geração de alertas de detecção para apoio ao SOC através da análise de *system calls* em tempo real;
- Análise dos dados através de interface *web* contendo *datasets* e anomalias indexados;
- Possibilidade de implementação de diferentes algoritmos de *machine learning* e abordagens para detecção de anomalias em *system calls* (análise de frequência, sequência, argumentos e outros dados);
- Capacidade de integração do *framework* com outras ferramentas.

Em trabalho futuro, serão fornecidos detalhes de implementação da topologia construída no GNS3, além de configurações das ferramentas utilizadas em cada camada da arquitetura apresentada. Resultados de detecção para alguns dos ataques atuais também serão fornecidos. Além disso, será disponibilizado um *dataset* de *system calls* gerado através do *framework* para uma aplicação em ambiente de produção, com o objetivo de auxiliar outras pesquisas.

AGRADECIMENTOS

Os autores agradecem o suporte da ABIN TED 08/2019.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] M. Souppaya, J. Morello, and K. Scarfone, “Application container security guide,” National Institute of Standards & Technology, SP 800-190, 2017.

[2] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," National Institute of Standards & Technology, SP 800-145, 2011.

[3] J. Flora, P. Gonçalves and N. Antunes, "Using Attack Injection to Evaluate Intrusion Detection Effectiveness in Container-based Systems," 2020 IEEE 25th (PRDC), 2020, pp. 60-69.

[4] K. Scarfone and P. Mell, "Guide to Intrusion Detection and Prevention Systems (IDPS)," National Institute of Standard & Technology, SP 800-94, 2007.

[5] X. Zhang, Q. Niyaz, F. Jahan and W. Sun, "Early Detection of Host-based Intrusions in Linux Environment," 2020 IEEE (EIT), 2020.

[6] B. Subba, S. Biswas, and S. Karmakar, “Host based intrusion detection system using frequency analysis of n-gram terms,” in TENCON, 2017.

[7] M. Liu, Z. Xue, X. Xu, C. Zhong, and J. Chen, “Host-based intrusion detection system with system calls: Review and future trends,” ACM, Nov. 2018.

[8] M. Kashkoush, C. Clancy, A. Abed, and M. Azab, “Resilient intrusion detection system for cloud containers,” International Journal of Communication Networks and Distributed Systems, vol. 24, p. 1, 2020.

[9] M. M. Rohling, M. Grimmer, D. Kreubel, J. Hoffmann, and B. Franczyk, “Standardized container virtualization approach for collecting host intrusion detection data,” in Proceedings of the 2019 Federated Conference on Computer Science and Information Systems, 2019.

[10] S. Srinivasan, A. Kumar, M. Mahajan, D. Sitaram, and S. Gupta, “Probabilistic real-time intrusion detection system for docker containers,” in Communications in Computer and Information Science, 2019, vol. 969, pp. 336–347.

[11] P. Cui and D. Umphress, “Towards unsupervised introspection of containerized application,” in ACM nternational Conference Proceeding Series, 2020.

[12] J. Byrnes, T. Hoang, N. N. Mehta, and Y. Cheng, “A Modern Implementation of System Call Sequence Based Host-based Intrusion Detection Systems,” in Proceedings - 2020 2nd IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications, 2020, pp. 218–225.

[13] W. Haider, J. Hu, and N. Moustafa, “Designing anomaly detection system for cloud servers by frequency domain features of system call identifiers and machine learning,” in Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, 2018, vol. 235.

[14] Y. Lin, O. Tunde-Onadele, and X. Gu, “CDL: Classified Distributed Learning for Detecting Security Attacks in Containerized Applications,” in PervasiveHealth: Pervasive Computing Tec. for Healthcare, 2020.

[15] C. Tien, T. Huang, C. Tien, T. Huang, and S. Kuo, “KubAnomaly: Anomaly detection for the Docker orchestration platform with neural network approaches,” Eng. Reports, vol. 1, no. 5, Dec. 2019.

[16] GNS3, "The software that empowers network professionals". [Online] Disponível em: https://gns3.com. [Acessado em 24/01/2022].

[17] Kubernetes, "Production-Grade Container Orchestration". [Online] Disponível em: https://kubernetes.io. [Acessado em 24/01/2022].

[18] Sysdig, "Security Tools for Containers, Kubernetes & Cloud". [Online] Disponível em: https://sysdig.com. [Acessado em 27/01/2022].

[19] Elastic, "Produtos da Elastic: Busca, analítica, logging e segurança". [Online] Disponível em: https://elastic.com/pt/products. [Acessado em 27/01/2022].

[20] Redis, "Redis". [Online] Disponível em: https://redis.io. [Acessado em 27/01/2022].

[21] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, “Survey of intrusion detection systems: techniques, datasets and challenges,” Cybersecurity, vol. 2, no. 1, Dec. 2019.

[22] Github, "rcids-public/CISTI/figuras at main · s4vio/rcids-public". [Online] Disponível em: https://github.com/s4vio/rcids-public/tree/main/CISTI/figuras. [Acessado em 01/04/2022].

ISBN: 978-989-33-3436-2

Authorized licensed use limited to: UNIVERSIDADE DE BRASILIA. Downloaded on January 25,2023 at 18:04:07 UTC from IEEE Xplore. Restrictions apply.