DeepL
Assine o DeepL Pro para traduzir arquivos maiores.
Mais informações em www.DeepL.com/pro.

2019 15ª Conferência Européia de Computação de Confiança (EDCC)

Estudo da aplicabilidade da detecção de intrusão em ambientes de contentores multitenant

Jose' Flora, Nuno Antunes
CISUC, Departamento de Engenharia Informática da Universidade de Coimbra
Pólo II, 3030-290 Coimbra - Portugal
jeflora@dei.uc.pt, nmsa@dei.uc.pt

Resumo **- *O* uso de recipientes em aplicações baseadas em nuvens permite implantações rápidas e escalonáveis. Os contêineres são leves e atraentes para serem usados mesmo em sistemas críticos para os negócios, mas seu uso implica em grandes preocupações de segurança, que são exacerbadas em ambientes multi-tenant. Para mitigar estas preocupações, técnicas como a detecção de intrusão são uma necessidade, como - sempre, no contexto dos contêineres, tem recebido atenção limitada. Assim, é necessário definir uma abordagem melhorada da detecção de intrusão no nível de contêineres para ambientes multitenant. Neste documento, fazemos uma análise preliminar de viabilidade da detecção de intrusão a nível de contêineres baseados em host. Para isso, estamos atualmente nos concentrando em alcançar uma definição estável do perfil do recipiente e os resultados obtidos mostram que estamos seguindo o caminho correto.**

Termos de índice **- *Contêineres*, Segurança, Detecção de intrusão, Multi- inquilinato, Avaliação experimental**

I. Introdução

Tem havido um crescimento no uso de contêineres em implantações de nuvens [1]. Esta tecnologia permite compartilhar uma infra-estrutura física sobre vários clientes de forma leve e rápida. Entretanto, isto só é possível porque os containers compartilham o núcleo do Sistema Operacional (SO) subjacente. Como os ambientes em nuvem são ambientes multi-tenant, re-localizados em tecnologias como *cgroups* e *namespaces* para pro- isolamento de contêineres de vídeo, a superfície de ataque das aplicações lá implantadas, que são freqüentemente críticas para os negócios, aumentou substancialmente. Portanto, há uma grande necessidade de implantar contra-medidas focadas em contêineres nestas infra-estruturas.

A detecção de intrusão tem sido amplamente utilizada em múltiplos contextos, no entanto, sua aplicação em recipientes ainda é esparsa. Este remédio, corretamente aplicado, é *imprescindível para que as* intrusões sejam detectadas e medidas de mitigação ou remediação sejam postas em prática. Como resultado, pretendemos desenvolver um Sistema de Detecção de Intrusão (IDS) funcional e baseado em anomalias, no qual os recipientes são perfilados durante seus períodos normais de execução e, em seguida, monitorados para eventos anômalos, tais como ataques *de dia zero* ou ataques conhecidos.

Até onde sabemos, foi feito um trabalho limitado no que diz respeito à detecção de intrusão a nível de contêineres. Em 2015, Abed *et. al* [2] propuseram um IDS para nível de contêineres, que utilizava Bags of System Call (BoSC) para representar as entradas normais do banco de dados de comportamento. Neste trabalho, foi proposto um IDS baseado na detecção de anomalias, capaz de monitorar um container e detectar possíveis tentativas de intrusão em tempo real. Durante a análise experimental, os autores afirmaram ter obtido

978-1-7281-3929-6/19/$41.00 ©2019 IEEE
DOI 10.1109/EDCC.2019.00033

Uso autorizado e licenciado limitado a: UNIVERSIDADE FEDERAL DO PARANAENSE. Download em 22.2023 de abril às 02:42:11 UTC do IEEE Xplore. As

uma Taxa Positiva Verdadeira (TPR) de 100% com uma Taxa Positiva

Falsa (FPR) de 2%. Em 2019, Srinivasan *et. al* [1] propuseram um IDS em tempo real que usa n-gramas de chamadas de sistema e a probabilidade de sua ocorrência. Os autores utilizaram o conjunto de dados da Universidade do Novo México (UNM) para validar a abordagem e afirmam ter obtido uma precisão que varia de 87-97%.

Neste artigo, estamos focados em entender se as técnicas de detecção de intrusão são aplicáveis a ambientes de recipientes multitenant, usando algoritmos de detecção de anomalias, tais como Sequence Time-Delaying Embedding (STIDE) [3] ou BoSC. Por isso, procuramos definir perfis estáveis para os contêineres sob monitoramento usando as chamadas do sistema, por eles executadas, para treinar os referidos algoritmos. Conseguimos observar a aplicabilidade de ambos STIDE e BoSC no contexto da detecção de intrusão, e concluímos que ambos os algoritmos têm uma capacidade promissora de definir perfis estáveis para os recipientes, seja Docker ou LXC.

A estrutura do papel é a seguinte. O Sec. II apresenta os antecedentes e a motivação, o Sec. III focaliza o escopo, a arquitetura pré-liminária, e apresenta os resultados e a discussão, e o Sec. IV delineia os próximos passos e conclui.

II. Antecedentes e Motivação

A detecção de intrusão baseada em nuvens tem visto alguns desenvolvimentos, em termos de modo de implantação e monitoramento, por exemplo, o uso de IDSs distribuídos. Apesar dos grandes avanços na detecção de intrusões para ambientes baseados em máquinas virtuais, as abordagens a nível de contêineres foram negligenciadas e as melhorias neste contexto ainda são escassas.

A detecção de intrusão é o processo de monitoramento dos eventos que ocorrem em um sistema ou rede de computadores e sua análise para sinais de possíveis incidentes [4], e é realizada pelo IDSes. Enquanto os IDSs baseados em rede processam mais dados e são normalmente colocados no perímetro da rede, monitorando assim uma área mais ampla, os IDSs baseados em host são colocados em máquinas, presentes em uma rede, monitorando-os e, portanto, tendo uma visão mais local dos eventos.

Além disso, o IDSes pode seguir duas abordagens diferentes para atingir seu objetivo. A abordagem **baseada na assinatura** consiste na identificação de padrões maliciosos conhecidos ao analisar novos eventos, tais como tráfego de rede ou dados de aplicação [5]. Esta abordagem resulta em uma baixa taxa de falsos positivos, mas é incapaz de detectar ataques recém-criados. Enquanto que a **base da anomalia** implica

Uso autorizado e licenciado limitado a: UNIVERSIDADE FEDERAL DO PARANAENSE. Download em 22.2023 de abril às 02:42:11 UTC do IEEE Xplore. As

duas fases, uma **fase de treinamento**, para construir o perfil, e uma **fase de detecção**, onde novos eventos são avaliados com base no perfil definido [6]. Portanto, esta abordagem é capaz de detectar novos ataques, embora possa produzir uma maior taxa de falsos positivos.

Isto normalmente implica o uso de algoritmos que vão desde métodos estatísticos até aprendizagem de máquinas e abordagens de mineração de dados, cujo uso tem provado ser útil no campo da detecção de intrusão. Redes Neurais Artificiais (ANNs) é uma metodologia que tem a capacidade de generalizar dados, sendo assim
capazes de detectar intrusões a partir de dados de treinamento incompletos [7]. Os Modelos Markov Escondidos (HMM) são usados para descrever o sistema sendo modelado através de um conjunto de conexões ponderadas entre cada par de estados, que representam a probabilidade de transição de um para o outro [8]. A aplicação do K- Nearest Neighbour (KNN) à detecção de intrusão também teve alguns resultados positivos, classificando os eventos com base na heurística de distância [9]. O método STIDE é baseado em uma janela que desliza sobre uma seqüência de chamadas do sistema [3]. Support Vector Machines (SVM) é um método que em sua essência divide os eventos em duas classes diferentes [7]. Normalmente, na detecção de anomalias, é utilizada uma variação SVM chamada One-Class Support Vector Machines (OCSVM), que classifica os dados em apenas uma classe, eliminando assim a necessidade de fornecer dados anômalos para o classificador quando em fase de treinamento.

Atualmente, o principal desafio é o fato de que **vários continers, com diferentes proprietários, coabitam na mesma máquina**, o que levanta a possibilidade de que alguns inquilinos, com intenções maliciosas, possam tentar comprometer a execução normal, seja dos vizinhos ou da máquina hospedeira. Assim, assumimos que os containers com intenções maliciosas, *inicialmente adquirem recursos de forma lícita e não abusiva*. Ou seja, somente após obter acesso ao contêiner, seu proprietário utiliza seus recursos para realizar atividades maliciosas, tais como tentar comprometer a execução de outros inquilinos. O fato de que os serviços de cloud computing assumem que seus consumidores são confiáveis, permite que um atacante possa alocar recursos, como qualquer outro cliente, e utilizá-los. Portanto, neste trabalho, assumimos que um agente obtém acesso aos recursos computacionais dentro de um serviço de cloud computing legalmente e, somente depois, os utiliza para realizar esforços ilegais e maliciosos.

Portanto, nossas preocupações se baseiam nas ameaças apresentadas por esses recursos legalmente alocados, que originalmente são considerados confiáveis apesar da possibilidade de ter intenções mal-intencionadas em relação a outros inquilinos ou à infra-estrutura de serviços na nuvem. O que de fato representa uma grande ameaça para as camadas subjacentes da pilha de serviços das nuvens e para a execução normal de outros contêineres honestamente obtidos sem motivações perversas.

III. Detecção de intrusão em contêineres

Meio

Ambiente

Este trabalho surge como resultado de ameaças iminentes a implantações de nuvens baseadas em contêineres, que são representadas na Fig. 1, considerada o modelo de ameaça neste trabalho.

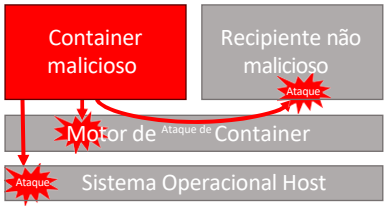


Fig. 1. Modelo de ameaça de implantação de nuvens baseadas em contêineres.

Nesta figura, é possível observar um recipiente malicioso, compartilhando a infra-estrutura com um não malicioso, e o

Uso autorizado e licenciado limitado a: UNIVERSIDADE FEDERAL DO PARANAENSE. Download em 22.2023 de abril às 02:42:11 UTC do IEEE Xplore. As

locais de ataque que se pode tomar como uma forma de comprometer a infra-estrutura ou outros recipientes. Geralmente, todo ataque visa comprometer a infra-estrutura ou seus componentes e obter algum tipo de recompensa, como mais recursos ou acesso não autorizado à informação.

Com relação ao ataque à máquina host, um recipiente malicioso pode tentar explorar as vulnerabilidades presentes na pilha da nuvem a fim de obter acesso ou obter controle de recursos privilegiados. Tipicamente, estes ataques são motivados pela ambição de coletar informações ou controlar a maneira pela qual os recursos são alocados.

Os contêineres são normalmente gerenciados por um middleware de orquestração, chamado de motor de contêineres. Isto também atua como um local de ataque para atingir a camada de SO, assim, este software também pode ser comprometido em implementações de nuvens, por contêineres desonestos, assumindo um papel de caminho para atingir ambos os ataques, ou seja, para a máquina hospedeira e para outros inquilinos. Assim, um contêiner malicioso precisaria comprometer o motor do contêiner para fazer com que os danos chegassem a outros contêineres ou ao próprio host. Os ataques ao motor são uma grande preocupação devido aos danos que podem vir de seu comprometimento.

Portanto, o foco principal deste trabalho é detectar tentativas de intrusão contra contêineres, que são implantados em um ambiente de múltiplos arrendamentos, ficando assim sujeitos aos riscos inerentes de compartilhar recursos físicos e não ter o controle dos mesmos. Este trabalho visa atuar como uma melhoria do estado da arte na detecção de intrusões no nível de contêineres, devido ao crescimento do uso em serviços de nuvem e também para auxiliar com garantias de isolamento.

A. Arquitetura Preliminar para Detecção de Intrusão

Nesta seção, fornecemos uma visão geral da arquitetura preliminar para a abordagem de detecção de intrusão, descrita na Fig. 2, que consiste em três componentes principais: o sistema sob monitoramento, uma ferramenta de coleta de dados e o analisador IDS.

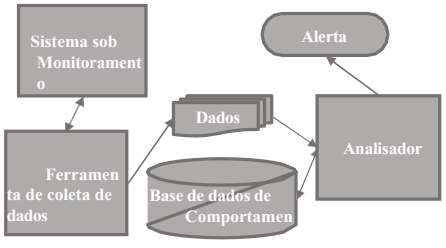


Fig. 2. Arquitetura preliminar proposta para a abordagem de detecção de intrusão.

Uso autorizado e licenciado limitado a: UNIVERSIDADE FEDERAL DO PARANAENSE. Download em 22.2023 de abril às 02:42:11 UTC do IEEE Xplore. As

Com relação ao sistema sob monitoramento, nosso foco se concentra novamente nos contêineres, ou seja, o detector de intrusão deve construir um perfil de um contêiner com base nas chamadas do sistema emitidas por ele para o sistema operacional subjacente.

Existem várias tecnologias de contentorização. Por enquanto, estamos nos concentrando no Docker e no LXC, no entanto, nosso objetivo é ter uma plataforma que seja capaz de trabalhar de forma agnóstica com a tecnologia de contêineres.

Para cumprir o componente de coleta de dados, selecionamos duas ferramentas. Em primeiro lugar, desencadeamos o *strace* [10], que é capaz de anexar aos processos em execução em uma máquina, monitorando e coletando as chamadas do sistema por eles invocadas juntamente com os argumentos que lhe foram passados. Em segundo lugar, selecionamos o *sysdig* [11]
uma vez que se trata de um container de monitoramento e auditoria spa-
recipientes de diálogo. Ela permite a coleta de traços de

contendo dados tais como chamadas ao sistema e outros eventos do sistema operacional.

Em suma, ambas as ferramentas são capazes de coletar chamadas de sistema e outros eventos do sistema operacional, no entanto, *o strace* é orientado para o processo enquanto o *sysdig* é orientado para o contêiner, o que significa que em nosso caso *o sysdig* pode ser a melhor escolha.

O componente analisador é a principal unidade do IDS que pretendemos construir. Consequentemente, queremos selecionar os algoritmos com melhor desempenho para o contexto do nível do contêiner, que podem ser usados independentemente ou combinados como um conjunto. Para isso, coletamos ou implementamos algoritmos utilizados para a detecção de intrusão. Neste artigo, focalizamos os algoritmos que implementamos, STIDE e BoSC, devido a sua alta adoção e eficácia comprovada [3] [12].

Além disso, nosso objetivo é avaliar o estado dos algoritmos de detecção de intrusão, utilizando múltiplos cenários realistas, no contexto da detecção de intrusão a nível de contêineres. Estes expertises ainda estão em seus primórdios, pois não há conjuntos de dados disponíveis para a detecção de intrusão de contêineres. Portanto, precisamos produzir um conjunto de dados representativo e significativo para realizar um conjunto válido de experimentos.

B. Resultados Preliminares

Nesta seção, apresentamos os resultados preliminares relativos ao treinamento do algoritmo STIDE utilizado para o desinteresse da técnica de intrusão. Primeiramente, usando sua representação de dados habitual, uma janela extraída de uma seqüência de chamadas do sistema, e usando a representação BoSC baseada em freqüência.

Coletamos traços benignos de con- tainers Docker e LXC rodando aplicativo servidor MySQL que recebeu a carga de trabalho produzida por uma implementação do TPCC On-Line Transaction Processing Benchmark [13], configurado com 100 armazéns e utilizando 50 clientes durante a execução da carga de trabalho. Realizamos duas execuções de coletas por 10 e 24 horas de período de coleta.

Após a conclusão desta coleta, pudemos analisar os traços produzidos tanto para LXC quanto para Docker container. A observação da Tabela I permite concluir que os containers LXC produzem traços mais longos com base em um conjunto

TABELA I
ANÁLISE DOS TRAÇOS COLETADOS.

Treinamento	Plataforma	Corre 1		Corre 2	
Tempo		Único	Total	Único	Total
10H	Docker	37	1,303,036,648	29	684,693,069
	LXC	116	3,279,400,309	127	2,962,761,415
24H	Docker	37	1,907,494,529	37	3,533,706,363
	LXC	129	9,617,350,813	129	10,305,731,175

em cada um deles. Calculamos o valor da inclinação da curva de crescimento, que, neste contexto, representa a taxa de novas combinações (janelas) de chamadas de sistema adicionadas ao banco de dados de comportamento normal dos classificadores durante um período de tempo.

Para isso, utilizamos as fórmulas $0 \leq \frac{Sts2 - Sts1}{t_{s2} - t_{s1}} \leq \sigma$ [14]
maior de chamadas de sistema exclusivas *por* conjunto de dados.

Em seguida, treinamos classificadores dos métodos STIDE e BoSC, com janelas que variam de 3 a 6 chamadas de sistema.

Uso autorizado e licenciado limitado a: UNIVERSIDADE FEDERAL DO PARANAENSE. Download em 22.2023 de abril às 02:42:11 UTC do IEEE Xplore. As

para decidir se um intervalo está em um estado estável de aprendizagem. Além disso, consideramos que o estado estacionário do procedimento de aprendizagem é alcançado quando a desigualdade acima é satisfeita 5 vezes seguidas para $\sigma = 0{,}15$, com base em experiências anteriores realizadas em [14]. Os resultados destes experimentos são apresentados em Tab. II, Docker à esquerda e LXC à direita.

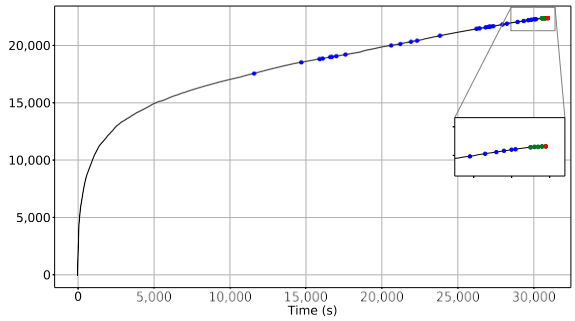
A Fig. 3 apresenta um display visual para o procedimento executado. Nesta figura, os pontos azuis representam os momentos em que a inclinação está abaixo de σ, os pontos verdes representam os quatro intervalos de estado estacionário anteriores ao intervalo em que os classificadores atingem o estado estacionário de aprendizagem, representado pelo ponto vermelho.

Fig. 3. Procedimento de treinamento para STIDE com janela 4 de corrida 1 de coleta 24h para container Docker.

A análise dos resultados produzidos, permite concluir que, no caso do STIDE, conseguimos alcançar um estado estável de aprendizagem usando o tamanho de janela 3 e 4, com STIDE, e 3 a 6 quando usamos BoSC. Isto significa que a melhor configuração, neste caso, permanece no uso de janelas de tamanho 3 e 4. Além disso, o fato de que o STIDE não alcançou um estado estável com janelas de tamanho 5 e 6, na maioria dos casos, deve-se ao fato de que é um método baseado em seqüência. Portanto, o número de combinações possíveis para diferentes janelas é maior do que para BoSC, o que significa que nosso valor limite pode exigir algum ajuste para este caso

IV. Lições Aprendidas e Pesquisas Futuras

Nossos passos seguintes residem na avaliação dos algoritmos de última geração para detecção de intrusão. Para isso, estamos focados em eliciar ataques e explorações, que serão utilizados



Uso autorizado e licenciado limitado a: UNIVERSIDADE FEDERAL DO PARANAENSE. Download em 22.2023 de abril às 02:42:11 UTC do IEEE Xplore. As

QUADRO II

RESULTADOS DO DOCKER & LXC PARA ALCANÇAR O ESTADO ESTÁVEL DE APRENDIZAGEM.

Algoritmo de Hora Treinamento		Janela ar	Ro	1 Tamanho DB	Roda T_{max}(s)	2 Tamanho DB	Algoritmo de Hora Treinamento		Janela ar	Ro	1 Tamanho DB	Roda T_{max}(s)	2 Tamanho DB
10H	STIDE	4	20,700	21,952	24,100	18,625	10H	STIDE	4	10,800	15,692	9,600	15,106
		5	-	-	-	-			5	-	-	-	-
		6	-	-	-	-			6	-	-	-	-
	BoSC	3	1,400	1,486	1,900	1,406		BoSC	3	3,900	2,376	6,800	2,487
		4	6,800	6,091	6,400	5,280			4	5,500	5,554	6,900	5,526
		5	20,700	19,066	20,800	16,273			5	13,800	13,979	19,300	19,783
		6	-	-	-	-			6	-	-	28,200	39,513
24H	STIDE	3	3,000	2,674	2,700	2,428	24H	STIDE	3	6,800	4,271	4,600	3,490
		4	30,900	22,317	29,800	22,091			4	12,100	19,563	24,300	26,312
		5	-	-	-	-			5	77,700	152,366	-	-
		6	-	-	-	-			6	-	-	-	-
	BoSC	3	1,700	1,649	1,500	1,460		BoSC	3	4,800	2,857	4,600	2,644
		4	5,800	5,898	6,800	5,801			4	6,800	6,970	17,400	10,728
		5	26,400	19,214	23,300	18,544			5	12,100	16,867	24,300	24,037
		6	34,200	43,942	54,900	49,151			6	35,600	42,830	25,200	42,019

T_{max}(s) representa quando o aprendizado em estado estável é alcançado, DB Size é o número de entradas do banco de dados normal naquele momento.

para construir os conjuntos de dados de avaliação. Tais ataques podem seguir diferentes locais de ataque, alguns dos especificados na Fig. 1. Portanto, avaliando os algoritmos de detecção de anomalias por formalidade no contexto de detecção de intrusão a nível de contêineres.

Além disso, para continuar o projeto de uma abordagem de detecção de intrusão, devemos produzir uma arquitetura mais detalhada do que a apresentada na Fig. 2. Esta arquitetura conterá especificações de seus componentes, bem como de suas interações, que servirão como base para sua implementação em um IDS funcional. Com relação a esta implementação, ela será objeto de avaliação para que possamos extrair métricas significativas e comparar seu desempenho com outros IDSs utilizados no mesmo contexto.

A realização deste trabalho permitirá avançar o estado atual da arte da detecção de intrusão em implantações de nuvens baseadas em contêineres. Além disso, a importância deste trabalho é alta como resultado da crescente presença do serviço de nuvem no cotidiano da população. Além disso, o aumento do uso da nuvem para usuários finais, bem como para empresas, comumente para sistemas críticos para os negócios, que utilizam infra-estruturas de terceiros para hospedar suas aplicações, torna-o atraente para usuários maliciosos. Estes agentes malévolos podem roubar informações confidenciais e causar danos às empresas, simplesmente impedindo que elas sejam alcançadas por seus clientes. Portanto, o aumento da qualidade das contramedidas disponíveis para estes sistemas é muito bem-vindo, assim como qualquer avanço neste contexto de segurança da informação é de grande importância.

AGRADECIMENTOS

Este trabalho foi parcialmente apoiado pelo projeto **ATMOSPHERE** (atmosfere-eubrazil.eu), financiado pelo Ministério da Ciência, Tecnologia e Inovação do Brasil (Projeto 51119
- MCTI/RNP 4ª Chamada Coordenada) e pela Comissão Européia no âmbito do Programa de Cooperação, acordo de subvenção Horizon 2020 no 777154. Também é parcialmente apoiado pelo projeto **METRICS**, financiado pela Foun- dation for Science and Technology (FCT) - acordo no POCI-01-0145-FEDER-032504.

REFERÊNCIAS

[1] S. Srinivasan, A. Kumar, M. Mahajan, D. Sitaram, e S. Gupta, "Probabilistic Real-Time Intrusion Detection System for Docker Containers", em *Segurança em Informática e Comunicações*. Springer Singapore, 2019, pp. 336-347.

[2] A. S. Abed, C. Clancy, e D. S. Levy, "Intrusion Detection System for Applications using Linux Containers", *arXiv:1611.03056 [cs]*, vol. 9331, pp. 123-135, 2015.

[3] S. Forrest, S. A. Hofmeyr, A. Somayaji, e T. A. Longstaff, "A sense of self for unix processes", em *In Proceedings of the 1996 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 1996, pp. 120-128.

[4] K. A. Scarfone e P. M. Mell, "Guide to Intrusion Detection and Prevention Systems (IDPS)", National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep. NIST SP 800-94, 2007.

[5] A. Patcha e J.-M. Park, "Uma visão geral da técnica de detecção de anomalias - niques: Existing solutions and latest technological trends", *Computer Networks*, vol. 51, no. 12, pp. 3448-3470, ago. 2007.

[6] S. A. Hofmeyr, S. Forrest, e A. Somayaji, "Intrusion detection using sequences of system calls", *Journal of Computer Security*, vol. 6, no. 3, pp. 151-180, Jul. 1998.

[7] W.-H. Chen, S.-H. Hsu, e H.-P. Shen, "Application of SVM and ANN for intrusion detection", *Computers & Operations Research*, vol. 32, no. 10, pp. 2617-2634, Out. 2005.

[8] C. Warrender, S. Forrest, e B. Pearlmutter, "Detecting intrusions using system calls: alternative data models", em *Proceedings of the 1999 IEEE Symposium on Security and Privacy (Cat. No.99CB36344)*. IEEE Comput. Soc., 1999.

[9] L. Li, H. Zhang, H. Peng, e Y. Yang, "Nearest neighbors based density paks approach to intrusion detection", *Chaos, Solitons & Fractals*, vol. 110, pp. 33-40, maio de 2018.

[10] Michael Kerrisk, "strace - trace system calls and signals", http: //man7.org/linux/man-pages/man1/strace.1.html, Oct. 2018, [Acesso: 2019-01-24]. [Online]. Disponível: http://man7.org/linux/man-pages/man1/strace.1.html

[11] Sysdig, Inc, "sysdig," https://sysdig.com/, 2019, [Acesso: 2019-01- 24]. [Online]. Disponível: https://sysdig.com/

[12] A. S. Abed, T. C. Clancy, e D. S. Levy, "Applying Bag of System Calls for Anomalous Behavior Detection of Applications in Linux Containers", em *2015 Workshops IEEE Globecom (GC Wkshps)*. IEEE, 2015, pp. 1-5.

[13] "Tpc-c benchmark", http://www.tpc.org/tpcc/, acessado em: 2019-04-10.

[14] A. Milenkoski, B. D. Payne, N. Antunes, M. Vieira, S. Kounev, A. Avritzer, e M. Luft, "Avaliação de sistemas de detecção de intrusão em ambientes virtualizados usando injeção de ataque", no *Simpósio Internacional sobre Avanços Recentes na Detecção de Intrusão*. Springer, 2015, pp. 471–492.

Uso autorizado e licenciado limitado a: UNIVERSIDADE FEDERAL DO PARANAENSE. Download em 22.2023 de abril às 02:42:11 UTC do IEEE Xplore. As

Uso autorizado e licenciado limitado a: UNIVERSIDADE FEDERAL DO PARANAENSE. Download em 22.2023 de abril às 02:42:11 UTC do IEEE Xplore. As