2022 IEEE International Conference on Communications Workshops (ICC Workshops) | 978-1-6654-2671-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICCWORKSHOPS53468.2022.9814620

Uma estrutura de aprendizagem agrupada para a detecção de intrusão em ambiente de contentores com base no hospedeiro

Jingfei Shen*‡ , Fanping Zeng†§ , Weikang Zhang*‡ , Yufan Tao*‡ , Shengkun Tao*‡
* *Escola de Informática e Tecnologia, Universidade de Ciência e Tecnologia da China, Anhui, China*
†Key Laboratório de Comunicação Óptica Wireless-Optical, Universidade de Ciência e Tecnologia da China
‡{*ericjeff*, buttman, tyf1999, *nonestack*}*@mail*.ustc.edu.cn
§billzeng@ustc.edu.cn

A tecnologia de Contentores-Abstractos tem sido amplamente utilizada em ambientes de computação de ponta. Utilizando o isolamento e gestão de recursos a nível de SO, pode alcançar uma eficiência elevada incomparável, em contraste com a abordagem tradicional da máquina virtual. Contudo, estudos recentes demonstraram que o ambiente de contentores é vulnerável a vários ataques de segurança. Além disso, a natureza altamente personalizável e dinâmica da mudança do contentor exacerbou a sua vulnerabilidade. Neste documento, propomos uma nova estrutura de detecção de anomalias combinando algoritmo de cluster para melhorar a eficiência da detecção de anomalias no ambiente de computação de bordo que contém um grande número de contentores. Utiliza o algoritmo de cluster para identificar automaticamente recipientes que executam a mesma aplicação, e constrói um modelo de detecção de anomalias para cada categoria separadamente. Investigámos 8 vulnerabilidades do mundo real de várias aplicações frequentemente utilizadas e avaliámos a nossa estrutura sobre as mesmas. Os resultados da experiência mostram que a nossa estrutura proposta pode identificar eficazmente recipientes construídos na mesma imagem de aplicação sem qualquer rotulagem manual, reduzir a FPR da detecção de anomalias de 0,61% para 0,09%, e aumentar a TPR de 90,3% para 96,2% em comparação com o método tradicional.

Termos de índice **- Segurança de *Contentores*, Ambiente de Computação de Bordos, Detecção de Intrusão, Cluster, Aprendizagem Máquina**

I. Introdução

A tecnologia de contentores tem sido amplamente adoptada no ambiente informático actual devido à sua escalabilidade, alta eficiência, e baixa sobrecarga de isolamento. A tecnologia de contentor é uma tecnologia de virtualização de nível de sistema operacional (SO) que tem múltiplos blocos de construção dentro do kernel Linux, incluindo isolamento de recursos, técnicas de controlo (por exemplo, namespace e cgroup) e mecanismos de segurança (por exemplo, Capacidades, SELinux [1], AppArmor [2], e Seccomp [3]). Ao evitar a sobrecarga de camadas de abstracção adicionais, os contentores podem atingir um desempenho quase nativo e superar os sistemas baseados em máquinas virtuais em quase todos os aspectos [4]. Além disso, o advento de sistemas de gestão e orquestração de contentores, tais como Docker e Kubernetes, alterou grandemente o ecossistema de construção, expedição e implantação de aplicações distribuídas no ambiente de computação de bordo.

A investigação actual sobre segurança de contentores inclui a análise estática de imagens de contentores e a detecção de anomalias de um contentor em funcionamento. O primeiro analisa os problemas de segurança detectando ficheiros binários, scripts, bibliotecas, e outros ficheiros na imagem. O segundo analisa a utilização de recursos e dados de registo do contentor para detectar o comportamento anómalo do contentor.

Neste documento, concentramo-nos nesta última abordagem, estudando o método de detecção de comportamento anómalo de contentores causados por ataques de segurança, ou seja, a detecção de intrusão baseada em anomalias. Um sistema de detecção de intrusão (IDS) é um dispositivo ou aplicação de software que monitoriza uma rede ou sistema em relação a actividades maliciosas ou violações de políticas [5]. Os métodos geralmente utilizados para a detecção de intrusões incluem a detecção baseada em assinaturas e anomalias. O método baseado na assinatura detecta anomalias através da correspondência de padrões, tais como sequências de bytes de tráfego de rede e sequências de instruções conhecidas de malware. Pode detectar eficazmente ataques maliciosos com padrões conhecidos, mas não consegue identificar ataques maliciosos desconhecidos [6]. O método baseado em anomalias é utilizado principalmente para detectar ataques desconhecidos. Constrói um modelo de comportamentos normais de confiança através de abordagens de aprendizagem automática e utiliza o modelo construído para detectar anomalias.

A tecnologia de contentores trouxe novos desafios à detecção de intrusão. Devido à leveza e à eficiência das características de actuação, uma máquina hospedeira contém frequentemente múltiplas instâncias de contentores, pelo que o sistema de detecção de intrusão da máquina hospedeira não consegue detectar eficazmente os comportamentos anómalos de cada contentor. Além disso, devido à grande variedade e à natureza altamente personalizável do recipiente, é sempre difícil generalizar o modelo de detecção entre diferentes recipientes, enquanto que a construção de um modelo para cada recipiente implicará uma grande sobrecarga de desempenho. Por conseguinte, são urgentemente necessários novos métodos para melhorar a detecção de anomalias nos contentores.

Neste documento, propusemos um quadro para melhorar a detecção de comportamentos de anomalias de con- tiner utilizando a agregação. A estrutura pró-agrupamento utiliza algoritmo de agrupamento para identificar os que contêm a mesma aplicação sem qualquer etiquetagem manual, construir um modelo para cada aplicação e efectuar a detecção de anomalias para cada uma delas. As principais contribuições deste trabalho são as seguintes:

- Um novo quadro de detecção de anomalias combinado com uma classificação não supervisionada (isto é, cluster) é proposto para melhorar a eficiência da detecção de intrusão no ambiente do contentor.
- Apresentamos uma classificação eficiente de contentores e detecção de anomalias através da integração de algoritmos de agrupamento com métodos leves de aprendizagem de máquinas.
- Este trabalho mostra como as chamadas de sistema podem ser utilizadas para inspeccionar o comportamento de um Contentor e para efectuar a classificação e detecção de anomalias.

Este artigo está organizado da seguinte forma: A Secção II apresenta um trabalho relacionado, que investiga a detecção de anomalias com base em chamadas de sistema. A Secção III apresenta a nossa proposta de abordagem da detecção de anomalias, que é depois avaliada na Secção IV. O documento conclui na Secção V.

II. Trabalho relacionado

Segurança dos contentores. As questões de segurança dos contentores têm atraído cada vez mais atenção nos últimos anos. Shu et al. [8] investigaram a vulnerabilidade das imagens nas plataformas de alojamento de contentores. Analisaram 356.218 imagens e descobriram que tanto as imagens oficiais como as comunitárias contêm uma média de 180 vulnerabilidades, e as vulnerabilidades nas imagens são passadas da imagem de base para a imagem derivada com dependência. Combe et al. [9] analisaram os possíveis canais de fuga de informação entre contentores e anfitrião e expuseram os potenciais riscos de segurança, tais como a inferência de dados privados e a mesma verificação do anfitrião, etc. Gao et al. [10] explicaram a fonte dos principais ataques do contentor Docker, compararam as características de segurança do contentor e do kernel Linux, e mencionaram o problema de segurança da utilização da tecnologia do contentor para comunicar directamente com e atacar o kernel do anfitrião. Para além da vulnerabilidade, há também muita investigação sobre a protecção da segurança do contentor. Yuqiong et al. [11] implementaram o espaço de nomes de Segurança no kernel Linux, o que permite aos contentores personalizar as políticas de segurança de forma independente e aplicá-las a processos específicos. Sergei et al. [12] propuseram SCONE, uma salvaguarda contra a SGX Trusted Execution Technology, utilizando processadores Intel. Este método pode encriptar/descriptar os dados de E/S a baixo custo, garantir que os dados do contentor não sejam utilizados pelo atacante, e aumentar a segurança do contentor.

Detecção de anomalias baseadas em chamadas de sistema. A ideia básica da detecção de anomalias é criar um modelo para operações de confiança através do método de aprendizagem de máquinas, e depois utilizar o modelo para detectar comportamentos anormais. Este método utiliza principalmente sequências de chamadas de sistema do processo como dados. Kang et al. [13] propuseram um método de representação denominado saco de chamadas de sistema (BoSC) para a detecção de anomalias. Este método divide a sequência de chamadas de sistema em segmentos, conta a frequência das diferentes chamadas de sistema em cada segmento, depois utiliza a sequência estatística para a detecção de anomalias. Tomaram a detecção como um problema de classificação e utilizaram a aprendizagem da máquina para detectar anomalias. Os resultados mostraram que a precisão e a taxa de falsos positivos foram significativamente melhoradas em comparação com os métodos anteriores.

A maior parte da detecção da anomalia baseada na chamada do sistema usa apenas o nome da chamada do sistema como características, mas para métodos de ataque deliberadamente construídos de forma errónea, tais como o ataque mímico, muitas vezes não consegue encontrar a tempo um comportamento anormal. Parampalli et al. [14] propôs um método de ataque mímico interactivo, no qual o atacante poderia modificar constantemente os dados e inserir código no script malicioso com base nos resultados do ataque, contornando assim o sistema de detecção de intrusão. A este respeito, Larson et al. [15] propuseram que a utilização de informações adicionais na chamada do sistema, tais como os parâmetros da chamada do sistema, o valor de retorno, a identificação do utilizador da pessoa que telefona, pode efectivamente melhorar a taxa de sucesso da detecção de anomalias.

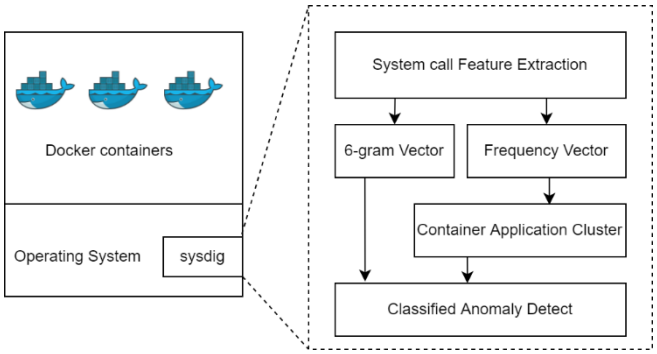

Fig. 1. Visão geral do quadro

Detecção de anomalias em contentores. Para um contentor iniciado, é essencialmente um conjunto de processos dentro do sistema hospedeiro. Assim, a detecção da anomalia baseada na chamada do sistema também é aplicável aos contentores. Abed et al. [17] propuseram aplicar o saco de chamadas de sistema à detecção de anomalias em contentores. Armazenam as chamadas de sistema normais numa base de dados de comportamento. Quando a detecção, se a sequência de chamadas de sistema actual não aparecer na base de dados, é marcada como uma anomalia. Se a anomalia marcada atingir um determinado limiar, o comportamento é considerado como uma anomalia. Karn et al. [18] propuseram um método de detecção para resolver o problema de que os recipientes são utilizados para realizar operações de criptografia em computação de alto desempenho. os seus resultados experimentais mostram que a precisão da árvore de decisão é de 97,1%, o que é significativamente superior a outra máquina
métodos de aprendizagem. Métodos de detecção de corrente para recipientes
concentram-se principalmente em campos específicos e não se podem aplicar ao ambiente de computação de bordo com vários tipos e uma enorme quantidade de contentores. Por conseguinte, é necessário um quadro de detecção de anomalias - trabalho que pode ser generalizado a diferentes aplicações de contentores.

III. Concepção da estrutura proposta

A. Visão geral do quadro

O nosso quadro proposto consiste em três principais componentes: colector de dados de chamada de sistema, classificador de contentores, e detector de anomalias. A figura 1 mostra uma visão geral do quadro proposto. Uma ferramenta open-source, out-of-the-box, sysdig [7], é utilizada para obter dados de chamada de sistema gerados por contentores para conseguir a detecção de anomalias não intrusivas e de baixo custo. Para a classificação de contentores, o algoritmo de cluster DBSCAN [19] é utilizado para classificar os contentores de uma forma não supervisionada. Durante a detecção, a estrutura constrói um classificador RandomForest [20] para cada categoria de aplicação, para detectar anomalias nas chamadas de sistema observadas.

B. Colector de dados da chamada de sistema

Monitorizamos os dados de chamada do sistema gerados pelo contentor utilizando a ferramenta sysdig, que fornece suporte nativo para contentores Linux incluindo Docker, LXC, etc. A chamada de sistema contém muita informação, como nome da chamada de sistema, parâmetros, e valor de retorno. Durante a experiência, encontrámos o nome

QUADRO I
VECTOR DE FREQUÊNCIA DE HTTPD

	frequência de chamada do sistema				
Timestamp	**sondagem**	**mmap**	**ler**	**fechar**	**writev**
1618900524618	0.0151	0.0640	0.2213	0.0987	0.0764
1618900524718	0.0156	0.0635	0.2309	0.1013	0.0804
1618900524818	0.0173	0.0643	0.2284	0.0960	0.0779
1618900524918	0.0272	0.0549	0.2546	0.0926	0.0883
1618900524019	0.0122	0.0721	0.2099	0.1026	0.0708
1618900524128	0.0274	0.0602	0.2250	0.0904	0.0848
1618900524229	0.0230	0.0588	0.2561	0.0964	0.0793
1618900524329	0.0285	0.0560	0.2445	0.0859	0.0896
1618900524430	0.0301	0.0644	0.2306	0.0964	0.0692

QUADRO II
VECTOR DE FREQUÊNCIA DO MYSQL

	frequência de chamada do sistema				
Timestamp	**sondagem**	**mmap**	**ler**	**fechar**	**writev**
1618900524618	0.0068	0.0000	0.2666	0.0034	0.0000
1618900524718	0.0057	0.0000	0.2859	0.0029	0.0000
1618900524818	0.0010	0.0000	0.3190	0.0000	0.0000
1618900524918	0.0068	0.0000	0.2754	0.0045	0.0000
1618900524019	0.0022	0.0000	0.2968	0.0022	0.0000
1618900524128	0.0059	0.0000	0.2831	0.0022	0.0000
1618900524229	0.0033	0.0000	0.3012	0.0022	0.0000
1618900524329	0.0019	0.0000	0.3151	0.0010	0.0000
1618900524430	0.0020	0.0000	0.3104	0.0007	0.0000

de chamada do sistema é suficiente para a classificação de contentores e detecção de anomalias. Assim, especificamos os parâmetros necessários para que o sysdig filtre outras informações e obtenha a sequência de nomes de chamada do sistema de um contentor em funcionamento.

A sequência de nomes de chamada do sistema recolhida é essencialmente um tipo de dados de série temporal. O sistema Linux contém aproximadamente
420 chamadas de sistema [21], que podem ser consideradas como eventos discretos do sistema. Para agrupamento, a sequência de nomes recolhidos é continuamente amostrada num intervalo de tempo específico e processada em muitos vectores de frequência semelhantes ao método de codificação One-Hot [22]. A dimensão do vector de frequência é o número de diferentes chamadas do sistema (ou seja, 420). Tabela I e Tabela
II mostram dois exemplos de vectores de frequência gerados por um contentor *Httpd* e um contentor *MySQL*. Cada carimbo de tempo corresponde a um vector de frequência $V = [f_1, f_2, ..., f_k]$, onde f_k representa a razão de frequência que a chamada sys-tem correspondente foi executada pelo contentor dentro do intervalo de amostragem.

C. Classificação de contentores

O objectivo da classificação é identificar os recipientes construídos pela mesma imagem de aplicação a partir da perspectiva do seu comportamento. Devido às suas características altamente personalizáveis, os contentores no ambiente de computação de bordo muitas vezes não têm informação detalhada da categoria de aplicação, e a grande quantidade dos mesmos torna difícil a etiquetagem manual. Além disso, os comportamentos de diferentes aplicações divergem muito uns dos outros, causando o modelo de detecção de anomalias que funciona bem para uma aplicação não adequada para a outra. O quadro II mostra um exemplo de um vector de frequência parcial gerado pelo contentor MySQL. Em contraste com o Quadro I, pode ser visto que o sistema chama

tais como *mmap*, *writev*, e *fstat*, que foram executados várias vezes pelo contentor Httpd, nunca foram executados no contentor MySQL. Portanto, a classificação não supervisionada (ou seja, cluster) é utilizada para alcançar uma classificação eficaz em dados de chamada de sistema não etiquetados.

Em termos do método de agrupamento, DBSCAN [19] é escolhido para o agrupamento de contentores. Este método conduz a classificação sem supervisão de acordo com a densidade. Em comparação com os meios K [23], poderia encontrar aglomerados de qualquer forma em vez de se limitar a uma forma convexa [24]. Durante o agrupamento, cada vector de frequência gerado por um recipiente participará no processo de agrupamento, e os seus correspondentes resultados de agrupamento são obtidos de acordo com o algoritmo DBSCAN. Depois disso, a categoria com o maior número de resultados de agregação de todos os vectores de frequência será considerada como a categoria do contentor.

D. Detecção de anomalias

Quanto à detecção de anomalias, os dados de chamada do sistema recolhidos de um contentor são primeiro mapeados em índices de palavras, e depois em termos de n-grama, que serão finalmente introduzidos num classificador RandomForest para detectar anomalias nos dados. O modelo N-grama é um método comummente utilizado no Processamento de linguagem natural, é uma sequência contígua de n itens de uma dada amostra de texto ou discurso. Existem outras abordagens padrão como a análise de componentes principais (PCA) utilizada em [25] para extrair características da sequência de chamada do sistema. A PCA é conhecida por ser computacionalmente intensiva e, por isso, imprópria para a detecção em tempo real. Amostra de chamadas de sistema em termos contíguos de 6 gramas. Como exemplo, considere a sequência de chamadas do sistema {1, 4, 2, 6, 12, 34, 12, ...}, com a
método de 6 gramas, será amostrado em estruturas de tal forma que
$frame_1$ = {1, 4, 2, 6, 12, *34}*, $frame_2$ = {4, 2, 6, 12, 34, *12}*,
$quadro_3$ = {2, 6, 12, 34, 12, *...}* e assim por diante. Os termos n-grama são divididos em formação e validação definidos como a convenção padrão de aprendizagem mecânica (proporção de 70:30).

Como descrito acima, o nosso quadro proposto constrói um modelo de detecção de anomalias para cada categoria de aplicação de acordo com os resultados do agrupamento. Para a detecção por categoria, é utilizado um classificador RandomForest para detectar anomalias. Ran- domForest é um método de aprendizagem em conjunto para classificação, regressão, e outras tarefas que funciona através da construção de uma multiplicidade de árvores de decisão em tempo de treino para melhorar a precisão ao mesmo tempo que se combate o excesso de ajuste. A figura 2 mostra o diagrama simplificado de uma floresta de decisão aleatória.

Cada árvore de decisão toma decisões locais óptimas de divisão entre um subconjunto aleatório das características ao dividir os nós. Como resultado, as árvores de decisão treinadas são normalmente bastante diferentes umas das outras. A saída do classificador florestal aleatório utiliza então o resultado da votação maioritária entre as árvores de decisão individuais. Muitos outros métodos de aprendizagem mecânica podem ser utilizados para efectuar a detecção de anomalias, tais como o K-NearestNeighbor al- gorithm (KNN) [26], Support Vector Machine(SVC) [27] e Neural Network. Contudo, ao avaliá-los no conjunto de dados ADFA- LD [28], descobrimos que o RandomForest supera outros métodos e é, portanto, seleccionado como detectores de anomalias.

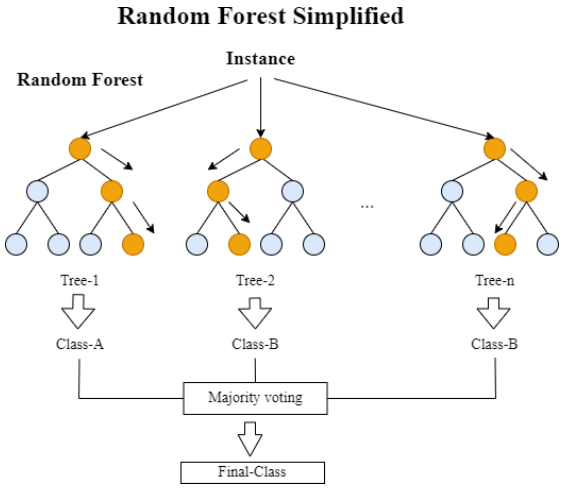

Fig. 2. Diagrama de uma floresta aleatória

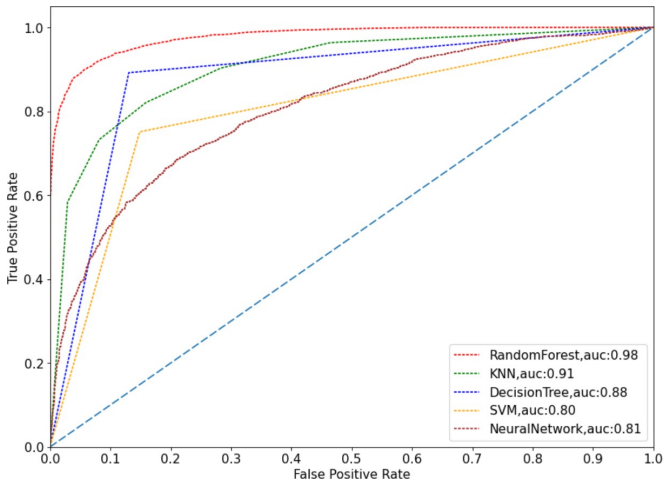

Fig. 3. Curva ROC de diferentes algoritmos no conjunto de dados ADFA-LD

A figura 3 mostra a curva ROC-AUC dos resultados da avaliação de diferentes algoritmos. A curva ROC é um gráfico que mostra o desempenho de um modelo de classificação em todos os limiares possíveis e a CUA é toda a área abaixo desta curva ROC. Da curva, podemos ver que RandomForest atinge a melhor pontuação AUC de 0,98, que é muito melhor do que outros algoritmos.

IV. Instalação e Avaliação de Experiências

Nesta secção, apresentamos a nossa configuração de ambiente experimental e análise de resultados. Implementamos a nossa estrutura de detecção e avaliamo-la num servidor Dell PowerEdge T440 com 40 CPUs de 2,20GHz e 64GB de memória a correr Ubuntu 20.04.1 sistema LTS.

A. Configuração da Experiência

1) Dados normais e anómalos: Seleccionámos 8 vulnerabilidades de 4 software de código aberto da Common Vulnerability Exposures(CVE). Estas vulnerabilidades cobrem questões de segurança relacionadas com a injecção de SQL, injecção de comando remoto, arbitrária escrita de ficheiros, etc. O quadro III mostra informação sobre cada vul- nerabilidade. Vulhub [29] foi utilizado para criar o nosso ambiente experimental. Fornece um ambiente vulnerável leve baseado em contentores Docker.

QUADRO III
Lista de Vulnerabilidades

Aplicação	Vulnerabilidade	Tipo
Django	CVE-2017-12794	Ataque XSS
Django	CVE-2019-14234	Injecção SQL
Django	CVE-2021-35042	Injecção SQL
Httpd	CVE-2017-15715	Política de Segurança Bypass
Httpd	bash-shellshock	Injecção por comando remoto
MySQL	CVE-2012-2122	Bypass de Identidade
Tomcat	CVE-2017-12615	Escrita de ficheiro arbitrário
Tomcat	CVE-2020-1938	Ficheiro arbitrário lido

As chamadas de sistema geradas por aplicações ociosas são normalmente chamadas de sistema relacionadas com o interruptor de espaço do núcleo do utilizador, que são muito homogéneas mesmo para diferentes aplicações. Por conseguinte, utilizamos JMeter [30] para simular a carga de trabalho da aplicação. Pode ser utilizado como uma ferramenta de teste de unidade para ligações a bases de dados JDBC, serviços web, HTTP, etc. Neste trabalho, utilizamos o amostrador HTTP do JMeter para enviar pedidos a uma aplicação que fornece serviço web, e o seu amostrador JDBC para enviar pedidos de leitura/escrita de base de dados a uma aplicação de base de dados. Quanto aos dados de anomalia, executamos o script de vulnerabilidade enquanto o contentor está a funcionar e registamos o carimbo da hora e os nomes das chamadas do sistema. Utilizamos o sysdig para extrair os dados das chamadas de sistema dos contentores. Especificamente, simulamos primeiro a carga de trabalho durante 30 segundos, recolhendo os nomes das chamadas de sistema e os seus correspondentes carimbos temporais, que são amostrados em vectores de frequência para agrupamento. Depois disso, utilizamos o script de exploração da vulnerabilidade para atacar o contentor, obtendo chamadas de sistema contendo anomalias. Estes dados são então amostrados em 6-gramas juntamente com os normais e utilizados para avaliar a eficiência do detector de anomalias.

2) Aglomeração: Os vectores de frequência são obtidos através de chamadas de sistema de amostragem a partir de contentores num intervalo de 10ms. O valor no vector de frequência reflecte a frequência com que a chamada de sistema tem correspondente é invocada no intervalo. Através da observação, descobrimos que aplicações sob carga normal de trabalho geram mais de 200 chamadas de sistema num intervalo de 10ms, assim, vectores de frequência contendo menos de 20 chamadas de sistema são descartados. Após a amostragem de todas as chamadas de sistema geradas, utilizamos o scikit- learn implementation do algoritmo DBSCAN para agrupar os vectores de frequência de todos os contentores. A distância Euclidiana é utilizada como métrica da distância entre dois vectores de frequência do algoritmo DBSCAN. Além disso, dois parâmetros *eps* e *minPts* são definidos para 0,2 e 5 respectivamente para alcançar resultados razoáveis de agregação. Finalmente, a categoria de um recipiente é a etiqueta que tem o número máximo de chamadas do sistema nos resultados de agrupamento.

3) Classificador RandomForest: O implementa- mento de scikit-learn de floresta aleatória é utilizado para efectuar a detecção de anomalias. Internamente, o RandomForest cria múltiplas Árvores de Decisão para classificar a mesma observação. A árvore de decisão criada depende de

sobre um conjunto de regras derivadas do processo de formação para dividir os dados em grupos que sejam o mais homogéneos possível. No nosso trabalho, é construído um RandomForest com 100 estimadores para avaliar a eficiência da detecção.

4) Abordagens alternativas de detecção: Avaliamos a nossa abordagem pro-posta em relação a dois métodos de detecção comummente utilizados no ambiente de computação de ponta.

Um detector para todos os contentores. Este método agrega os dados de chamada do sistema de todos os contentores para detecção de anomalias sem distinguir a categoria de aplicação do con- tainer. Devido à grande variedade de contentores no ambiente de computação de bordo e às diferenças nas sequências de chamadas de sistema geradas por diferentes aplicações, este método pode ser menos eficaz para a detecção.

Um detector para cada contentor. Este método constrói um modelo para cada contentor para detectar anomalias e pode alcançar resultados consideráveis num único contentor. Para um grande número de contentores, contudo, esta abordagem implica um custo de desempenho que não pode ser ignorado, o que dificulta a sua realização.

5) Métricas de avaliação: Taxa Positiva Verdadeira (TPR, também conhecida como sensibilidade) e Taxa Positiva Falsa (FPR, também conhecida como rácio de falso alarme) são utilizadas para comparar os resultados de detecção com diferentes abordagens. O cálculo para estas métricas é dado pelas seguintes equações onde TP denota o número de amostras que o detector identifica correctamente enquanto FP denota o número de amostras que o detector identifica falsamente e TN denota o número de amostras que o detector rejeita correctamente enquanto FN denota o número de amostras que o detector rejeita falsamente.

$$TPR = \frac{TP}{TP + FN} \quad (1)$$

$$FPR = \frac{FP}{PF + TN} \quad (2)$$

B. Análise de resultados

1) Aglomerado de contentores: O objectivo do agrupamento é identificar eficazmente os contentores construídos a partir da mesma imagem de aplicação. Comparamos os resultados de agrupamento do algoritmo DBSCAN utilizado pela estrutura proposta com o algoritmo clássico KMeans. Fig. 4 e Fig. 5 mostram os resultados de agrupamento de DBSCAN e KMeans. Apresenta a proporção de chamadas de sistema que são rotuladas como categorias diferentes nos resultados de agrupamento para cada contentor. Pode-se ver que para o DBSCAN, a maioria das chamadas de sistema para a mesma aplicação são etiquetadas como a mesma classe, indicando que podem efectivamente identificar recipientes que executam a mesma aplicação. Por exemplo, três contentores Django rotulados como *label* 0, dois contentores a executar o Apache Tomcat
aplicação são rotuladas como *rótulo* 3. Para KMeans, no entanto
ou seja, um detector para cada categoria de aplicação, atinge o FPR mais pequeno de 0,10% no MySQL e 0,016% no Tomcat. Enquanto nas aplicações Django e Httpd, a proposta ap-roach alcança um FPR semelhante mais baixo de 0,09% em comparação com o detector para cada aproximação de contentor. No entanto, o único detector para a aproximação de todos os contentores atinge o FPR mais alto de 0,6% devido à sua incapacidade de modelar um grande número de chamadas de sistema de várias aplicações. A figura 7 mostra o TPR das três aproximações em todas as categorias de aplicação. Pode-se ver que a estrutura proposta aumenta o TPR para 95,9% de 91,2% na aplicação Apache Tomcat em comparação com um detector para todas as abordagens, enquanto nas outras três categorias de aplicação, atinge quase o mesmo TPR de 97% que os outros dois métodos.

A partir dos resultados experimentais, pode-se descobrir que, devido à capacidade de encaixe limitada de um único modelo, a abordagem de um detector para todos os contentores executa o pior, o único detector para cada contentor atinge os melhores resultados globais mas necessita de estabelecer um modelo de detecção para cada contentor. A estrutura que propusemos pode alcançar uma eficácia de detecção comparável ou mesmo melhor do que o método de um detector para cada contentor, ao passo que só precisa de construir um modelo de detecção para cada categoria de aplicação.

V. Conclusão

Neste documento, propusemos um quadro para melhorar a detecção de anomalias em ambientes de contentores, agrupando os contentores em diferentes categorias de aplicação e construindo um modelo de detecção para cada categoria. Mostramos que, com o algoritmo DBSCAN, os dados de chamada de sistema podem ser utilizados para classificar contentores
sem qualquer etiquetagem manual, e identificam eficazmente os recipientes que contêm a mesma aplicação. Ao construir um modelo de detecção de anomalias para cada categoria de aplicação, a nossa estrutura proposta pode reduzir a taxa de falsos positivos de 0,6% para 0,09% em comparação com o detector único para todas as abordagens, e alcançar melhores resultados de detecção em comparação com o detector tradicional
não conseguiu identificar correctamente o mesmo contentor e erroneamente
rotular recipientes de diferentes aplicações como sendo da mesma categoria,
por exemplo, dois contentores Httpd rotulados como categorias diferentes.

C. Detecção de aglomerados

A figura 6 mostra a FPR das três abordagens em todas as categorias de ap- plicação. Pode-se ver que a abordagem proposta,

abordagens de detecção de anomalias.

AGRADECIMENTOS

Este trabalho é apoiado em parte pelo Programa Nacional Chave de I&D da China 2018YFB2100300, 2018YFB0803400, e pela National Natural Science Foundation of China (NSFC) ao abrigo da subvenção 61772487.

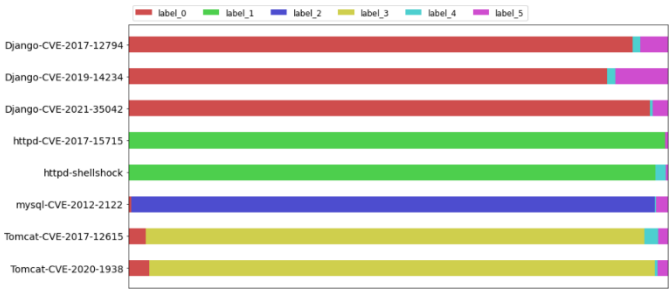

Fig. 4. Resultados do agrupamento de DBSCAN

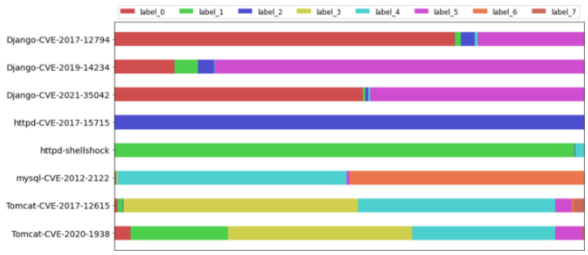

Fig. 5. Resultado do aglomerado de kmeans

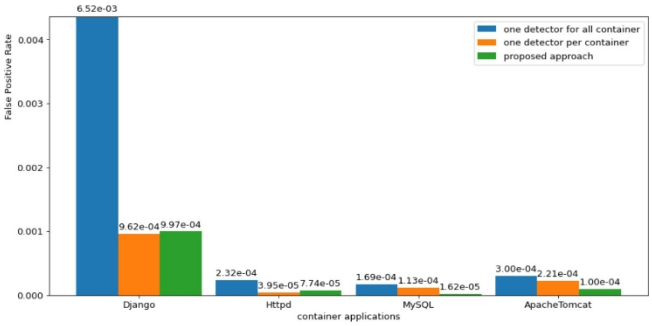

Fig. 6. Detecção FPR entre diferentes abordagens

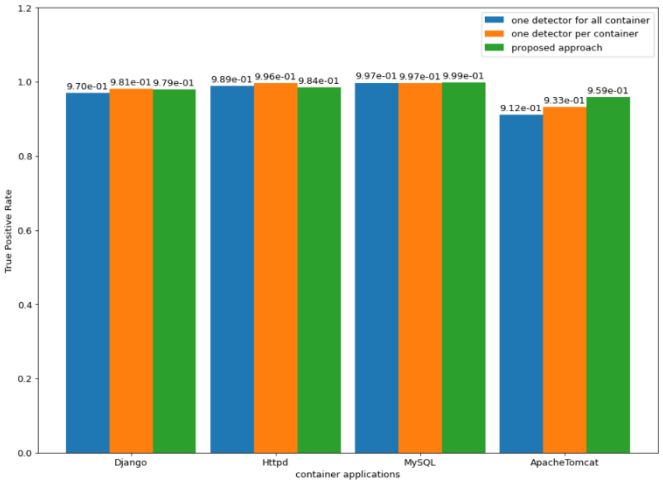

Fig. 7. Detecção de TPR entre diferentes abordagens

REFERÊNCIAS

[1] 2021. Segurança-Aumentada Linux. https://en.wikipedia.org/wiki/Linux Avançado em Segurança.

[2] 2021. AppArmor. https://en.wikipedia.org/wiki/AppArmor.

[3] 2021. Seccomp. https://en.wikipedia.org/wiki/Seccomp.

[4] Containers Not Virtual Machines Are the Future Cloud. http://www.linuxjournal.com/conteúdo/content/containers.

[5] 2021. Sistema de detecção de intrusão. https://en.wikipedia.org/wiki/Sistema de detecção de intrusão.

[6] Douligeris, Christos; Serpanos, Dimitrios N. (2007-02-09). Segurança de Redes: Estado Actual e Direcções Futuras. John Wiley & Sons. ISBN 9780470099735

[7] 2021. Secure DevOps for Containers, Kubernetes, and Cloud. https://docs.sysdig.com.

[8] Shu R, Gu X, Enck W. Um estudo das vulnerabilidades de segurança no centro portuário[C]//Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy. 2017: 269-280.

[9] Combe T, Martin A, Di Pietro R. Para atracar ou não para atracar: Uma perspectiva de segurança[J]. IEEE Cloud Computing, 2016, 3(5): 54-62.

[10] Gao X, Steenkamer B, Gu Z, et al. Um estudo sobre as implicações de segurança de fugas de informação em nuvens de contentores[J]. IEEE Transactions on Dependable and Secure Computing, 2018.

[11] Sun Y, Safford D, Zohar M, et al. Espaço de nomes de segurança: disponibilização de estruturas de segurança linux para contentores[C]//27º USENIX Security Symposium (USENIX Security 18). 2018: 1423-1439.

[12] Arnautov S, Trach B, Gregor F, et al. SCONE: Contentores de linux seguros com informações SGX[C]//12º Simpósio USENIX sobre Concepção e Implementação de Sistemas Operativos (OSDI 16). 2016: 689-703.

[13] Kang D K, Fuller D, Honavar V. Classificadores de aprendizagem para a detecção de erros de utilização e anomalias utilizando um saco de chamadas do sistema representa- tion[C]// Procedimentos do Sexto Workshop Anual de Garantia de Informação SMC IEEE. IEEE, 2005: 118-125.

[14] Parampalli C, Sekar R, Johnson R. Um ataque prático de mímica contra poderosos monitores de chamada de sistema[C]//Processos do ACM sym- posium 2008 sobre segurança da informação, computador e comunicações. 2008: 156-167.

[15] Larson U E, Nilsson D K, Jonsson E, et al. Utilizando informações de chamada de sistema para revelar manifestações de ataques ocultos[C]//2009 Actas do 1º Workshop Internacional sobre Segurança e Redes de Comunicação. IEEE, 2009: 1-8.

[16] Sakurada M, Yairi T. Anomaly detection using autoencoders with nonlinear dimensionality reduction[C]//Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis. 2014: 4-11.

[17] Abed A S, Clancy T C, Levy D S. A aplicação de saco de sistema exige a detecção de comportamentos anómalos de aplicações em linux contém- e r s [C]//2015 IEEE Globecom Workshops (GC Wkshps). IEEE, 2015: 1-5.

[18] Karn R R, Kudva P, Huang H, et al. Detecção de Criptominas em Nuvens de Con- tainer Utilizando Chamadas de Sistema e Aprendizagem de Máquina Explicável[J]. IEEE Transactions on Parallel and Distributed Systems, 2020, 32(3): 674-691.

[19] Ester, Martin; Kriegel, Hans-Peter; Sander, Jo¨rg; Xu, Xiaowei (1996). Simoudis, Evangelos; Han, Jiawei; Fayyad, Usama M. (eds.). Um algoritmo baseado na densidade para a descoberta de clusters em grandes bases de dados espaciais com ruído. Actas da Segunda Conferência Internacional sobre Descoberta de Conhecimento e Exploração de Dados (KDD-96). AAAI Press. pp. 226-231.

[20] Breiman, "Random Forests", Machine Learning, 45(1), 5-32, 2001.

[21] 2021. syscalls(2) - página de manual do Linux. https://man7.org/linux/man-pages/man2/syscalls.2.html.

[22] Harris, David e Harris, Sarah (2012-08-07). Digital design and computer architecture (2ª ed.). São Francisco, Califórnia: Morgan Kaufmann.
p. 129. ISBN 978-0-12-394424-5

[23] S. Lloyd, "Least squares quantization in PCM," in IEEE Transactions on Information Theory, vol. 28, no. 2, pp. 129-137, March 1982, doi: 10.1109/TIT.1982.1056489.

[24] Scikit-learn: Aprendizagem mecânica em Python, Pedregosa et al., JMLR 12,
pp. 2825-2830, 2011.

[25] J. Zhang, K. Zhang, Z. Qin, H. Yin, e Q. Wu, "Sensitive system calls based packed malware variants detection using main component initialized multilayers neural networks," Cybersecurity, vol. 1, no. 1, 2018, Art. no. 10.

[26] 2021. Algoritmo k-nearest Neighbors. https://en.wikipedia.org/wiki/Algoritmo K-nearest Neighbors.

[27] Platt, John (1999). "Probabilistic outputs for support vector machines and comparison to regularizedelihood methods".

[28] G. Creech. Desenvolver um Sistema de Detecção de Trusão Baseado em Hospedeiras de alta precisão, capaz de detectar de forma fiável ataques de dia zero, 2014.

[29] 2021. Ambientes Vulneráveis Pré-Construídos Baseados no Docker-Compose. https://github.com/vulhub/vulhub.

[30] 2021. Apache JMeter. https://jmeter.apache.org/.