Sistema Probabilístico de Detecção de Intrusão em Tempo Real para Docker Containers

Siddharth Srinivasan, Akshay Kumar(✉), Manik Mahajan, Dinkar Sitaram, e Sanchika Gupta

Universidade PES, Bangalore, Índia
akshayma@gmail.com

Abstrato. O uso de contêineres se tornou comum e onipresente em ambientes de nuvens. Um recipiente é uma forma de abstrair processos e sistemas de arquivo em uma única unidade separada do núcleo. Eles fornecem um ambiente virtual leve que agrupa e isola um conjunto de processos e recursos como memória, CPU, disco, etc., do host e de quaisquer outros containers. O Docker é um exemplo de tecnologias baseadas em contêineres para recipientes de aplicação. Entretanto, existem problemas de segurança que afetam o uso difundido e confiante da plataforma de contêineres. Este documento propõe um modelo para um sistema de detecção de intrusão em tempo real (IDS) que pode ser usado para detectar aplicações maliciosas executadas em contêineres Docker. Nosso IDS utiliza n-gramas de chamadas de sistema e a probabilidade de ocorrência deste n-grama é então calculada. Além disso, o rastreamento é processado
usando o Maximum Likelihood Estimator (MLE) e o Simple Good Turing (SGT) para fornecer uma melhor estimativa dos valores invisíveis das seqüências de chamada do sistema. O conjunto de dados UNM tem sido usado para validar a abordagem e uma comparação dos resultados obtidos usando MLE e SGT tem sido feita. Temos uma precisão que varia de 87-97% para diferentes conjuntos de dados UNM.

Palavras-chave: Detecção de intrusão - Contêineres - Computação em nuvem

1 Introdução

A virtualização pode ser implementada em dois tipos, máquinas virtuais (convencionais) e containers (novos). A forma convencional de emular um sistema e seus recursos é através de máquinas virtuais rodando em cima do hipervisor, que roda em cima do sistema operacional hospedeiro. Estamos trabalhando com containers, que são processos que rodam em cima do sistema operacional hospedeiro, ao invés do hipervisor. Existem certos problemas com a virtualização de máquinas virtuais convencionais: Há um aumento da sobrecarga de rodar um sistema operacional totalmente
instalado sistema operacional convidado. Há uma sobrecarga significativa das chamadas para o hipervisor de um sistema operacional convidado. As máquinas

virtuais mostram uma incapacidade de alocar livremente recursos aos processos. As principais empresas de nuvem como Google (Google Drive), Amazon (AWS) e Microsoft (Azure) estão, ao invés disso, utilizando containers, para executar seus processos em
servidores como um IaaS. Os containers fornecem um mecanismo para isolar um conjunto de processos e recursos (Processador, Memória, disco, etc.) do host e de quaisquer outros containers. Este isolamento garante que qualquer processo dentro do contêiner não possa ver outros processos ou recursos fora do contêiner. Os contêineres estão sendo usados atualmente como um

S. M. Thampi et al. (Eds.): SSCC 2018, CCIS 969, pp. 336-347, 2019.
https://doi.org/10.1007/978-981-13-5826-5_26

sistema operacional ou como um mecanismo de embalagem de aplicação. Os contêineres OS são ambientes virtuais que compartilham o núcleo do sistema operacional hospedeiro, mas fornecem isolamento do espaço do usuário. Os recipientes de aplicação são projetados para embalar e executar um único serviço. A maioria das empresas (como Amazon, Google e Microsoft) usa containers Docker para executar suas aplicações isoladas do kernel do host no servidor. Ao invés de virtualizar o hardware (que requer imagens completas do sistema operacional virtualizado para cada convidado), os containers Docker virtualizam o próprio SO, compartilhando o kernel do SO host e seus recursos com o host e outros containers. Resumidamente, os containers Docker apenas abstraem o kernel do sistema operacional ao invés de todo o dispositivo. Os contêineres Docker compartilham recursos do kernel através de recursos como namespaces, chroot e cgroups. Estes permitem aos con- tainers isolar processos, gerenciar completamente os recursos e garantir a segurança adequada.

Os contêineres Docker estão provando ser altamente leves e, portanto, rápidos em sua execução e com bom desempenho. Entretanto, sua segurança tem sido a questão chave, levantada em todas as conferências de virtualização da Docker. Os contêineres Docker são vulneráveis, quando se trata de ataques como fuga de contêineres e negação de serviço (DoS). Os contêineres Docker são agora onipresentes e uma solução predominante quando se trata de virtualização em servidores Linux e, portanto, a análise de segurança através da detecção de intrusão é vital e crucial para garantir o funcionamento seguro das aplicações. Como a comunidade Docker está sempre trabalhando para retificar e documentar estas vulnerabilidades 24 horas por dia, esta área nos deu uma boa margem para descobrir maneiras de detectar a possível ocorrência de
vulnerabilidades, ao invés das vulnerabilidades per se. Esta detecção é feita através de desvios dos padrões observados nas imagens oficiais, testadas e, portanto, seguras do Docker (Um container é uma instância em funcionamento de uma imagem. Todas as instruções para colocar o contêiner em funcionamento estão nas imagens). Primeiro, o IDS é executado para conjuntos de dados seguros e seu comportamento é registrado. Mais tarde, ao executar o IDS em conjuntos de dados maliciosos, seremos capazes de invadir os contêineres e criar as anomalias que detectamos, provando assim que estamos certos de que essas intrusões criam problemas que são detectados. Nós
testaram nossa abordagem nos conjuntos de dados UNM [9] para comparar o desempenho. É uma forma holística de rever como um contêiner pode ser afetado e que ele não é verdadeiramente seguro e protegido.

O papel é dividido em 5 seções. As seções 2, 3 e 4 falam sobre Pesquisa de Literatura, Proposta de Abordagem e Avaliação, respectivamente. A seção 5 conclui o trabalho seguido por Referências.

2 Pesquisa de Literatura

A detecção de anomalias no sistema baseado na seqüência já existe há muito tempo. Abed et al. [3] usaram a abordagem baseada em freqüência de bolsas de chamadas de sistema (BoSC) para rastrear chamadas de sistema em vez de seqüenciais. Eles foram capazes de detectar anomalias no comportamento de con- tainer e analisá-las. Se o número de desvios de freqüência de ocorrências de chamadas de sistema em um recipiente de teste exceder o das seqüências seguras das oficiais, então uma anomalia é detectada. Ela alegou ter um desempenho melhor e mais rápido para...
mance do que o método convencional. No entanto, sentimos que as freqüências

seriam menos precisas. As seqüências ofereceriam uma lista branca mais concreta de chamadas do sistema. Isto porque, uma chamada de sistema poderia ocorrer em qualquer ordem com a mesma freqüência. Se essa chamada,

digamos, apaga um arquivo antes que outra chamada leia que o arquivo apagado, a freqüência de ocorrências - as ocorrências das chamadas do sistema não mudariam, mas por causa da seqüência podemos capturar o comportamento anômalo. Assim, sentimos que as seqüências dariam uma visão detalhada dos traços e detectariam melhor potenciais e possíveis anomalias.

[5] falou sobre o uso de N-Grams para modelagem de linguagem. Dada a história das palavras vistas, eles previram a próxima palavra usando o Modelo Markov de N-Grams. Eles calcularam e depois suavizaram as probabilidades das seqüências de duas palavras (bigrams), pois os bigrams podem representar todas as palavras históricas daquela época. Usando todas essas probabilidades, eles foram capazes de completar uma máquina e até mesmo produzir uma frase em inglês. Naseer et al. [8] propuseram um classificador para seqüências arbitrariamente longas de chamadas de sistema utilizando uma abordagem de classificação Bayes naïve. As probabilidades de classe condicional de longo
Seqüências de chamadas de sistema são mapeadas para um Modelo de Cadeia de Markov Observável para classificar as seqüências de entrada de forma eficiente. A técnica foi comparada com as técnicas de classificação líderes e foi descoberto que para ter um bom desempenho contra técnicas como a ingênua Bayes multinomial, Support Vector Machine (SVM) e Logistic Regres-
sion. Além disso, permite um melhor comprometimento da taxa de detecção em relação à precisão. Mas o problema com sua abordagem foi que eles não se preocupam com os parâmetros de chamada do sistema.

[4, 7] alavancar argumentos de chamada de sistema, informações contextuais e conhecimento de nível de domínio para produzir clusters para cada chamada de sistema individual. Estes clusters são então usados para reescrever seqüências de processo de chamadas de sistema obtidas a partir dos logs do kernel. Estas novas seqüências são então alimentadas por um classificador Bayes ingênuo que constrói habilidades de sondagem condicional de classe a partir da modelagem de Markov de seqüências de chamadas de sistema. Os resultados foram então testados no conjunto de dados de 1999 da DARPA e se constatou que apresentavam melhorias significativas de desempenho em termos de taxa de falsos positivos, mantendo uma alta taxa de detecção quando comparados com outros classificadores. O problema com este trabalho foi que a identificação
o melhor subconjunto de chamadas do sistema para cluster foi feito manualmente, o que é ineficiente e o
A classificação dos clusters de chamadas de sistema demorou muito tempo.

Stolfo et al. [6] publicaram um artigo, "PAYL: Anomalous Payload-based Network Intrusion Detection" com relação à análise da carga útil da rede. Os dados de treinamento são um perfil que consiste nas freqüências relativas das cargas úteis rastreadas. Eles classificam cada uma das cargas úteis como 1 grama. A freqüência relativa de cada 1 grama é o número de
ocorrências do 1 grama dividido pelo número total de 1 grama. Seu desvio padrão é então computado por porta. Isto completa os dados do treinamento. Este processo de cálculo é repetido para cada conjunto de teste rastreado. A distância Mahalanobis é computada entre o conjunto de treinamento e cada conjunto de teste. Se a distância exceder um determinado limite, eles levantam um alerta (de anomalia). Esta abordagem propõe a detecção de intrusão para sistemas tradicionais utilizando cargas úteis de rede.

Nosso objetivo é desenvolver um Sistema de Detecção de Intrusão para aplicações em Docker Containers, ou seja, dada uma aplicação em um container, nosso sistema deve ser capaz de determinar se essa aplicação é maliciosa ou não. Planejamos desenvolver um Sistema de Detecção de Intrusão (HIDS) para monitorar aplicações

que rodam em uma única máquina. O HIDS proposto pode realizar o seguinte:
(a) Monitora a aplicação em execução dentro do container Docker em tempo real (b) Utiliza uma abordagem baseada em chamadas do sistema para detecção de anomalias e, portanto, reporta intrusões quando elas ocorrem.

3 Abordagem proposta

Este documento propõe uma abordagem de n-grama para detecção de intrusão usando chamadas de sistema para detectar aplicações maliciosas em ambiente de contêineres. Ao contrário da abordagem baseada na freqüência proposta por Abed et al. [3], cada seqüência de chamadas de sistema é mantida como um n-grama, em vez disso, para contabilizar a proporção de ocorrências de chamadas de sistema, tendo em mente a ordem em que as chamadas de sistema ocorrem também. A abordagem adotada pode alcançar maior precisão na detecção de ataques como ataques de Trojan, DoS, DDoS e SQL Injection que ocorrem em aplicações rodando em containers Docker.

Tabela 1. Exemplo de probabilidades de bigram (após suavização)

Seqüência de bigramas	Probabilidade
{alarme, sigprocmask}	0.5
{alarme, sigsuspend}	0.5
{brk, brk}	0.25
{brk, sigprocmask}	0.25
{chdir, abrir}	1.0

Tabela 2. Exemplo de probabilidades de trigrama (após suavização)

Seqüência de trigramas	Probabilidade
{alarme, sigprocmask, select}	0.5
{alarme, sigsuspend, sigreturn}	0.5
{brk, brk, abrir}	0.25
{brk, fstat, mmap}	0.5
{brk, aberto, stat}	0.25

A configuração de nosso sistema de detecção de intrusão para um container Docker é como mostrado na Fig. 1. Um ponto comum de montagem é feito entre o container e o sistema hospedeiro usando uma pasta compartilhada. O serviço Web executado dentro do container é rastreado usando o utilitário strace usando todos os identificadores de processo associados ao serviço, e o traço das chamadas de sistema que são obtidas em tempo real é passado para o IDS. Isto fornece apenas um mecanismo para o IDS ler a seqüência de chamadas de sistema geradas dentro do recipiente em tempo real.

Cada seqüência é passada para o IDS onde gera n-gramas de chamadas de sistema e continua calculando as probabilidades de ocorrências desses n-gramas. Estas probabilidades calculadas são usadas para acumular as probabilidades relativas totais de n-gramas para aquela sessão de monitoramento do recipiente (Tabelas de Ref. 1 e 2).

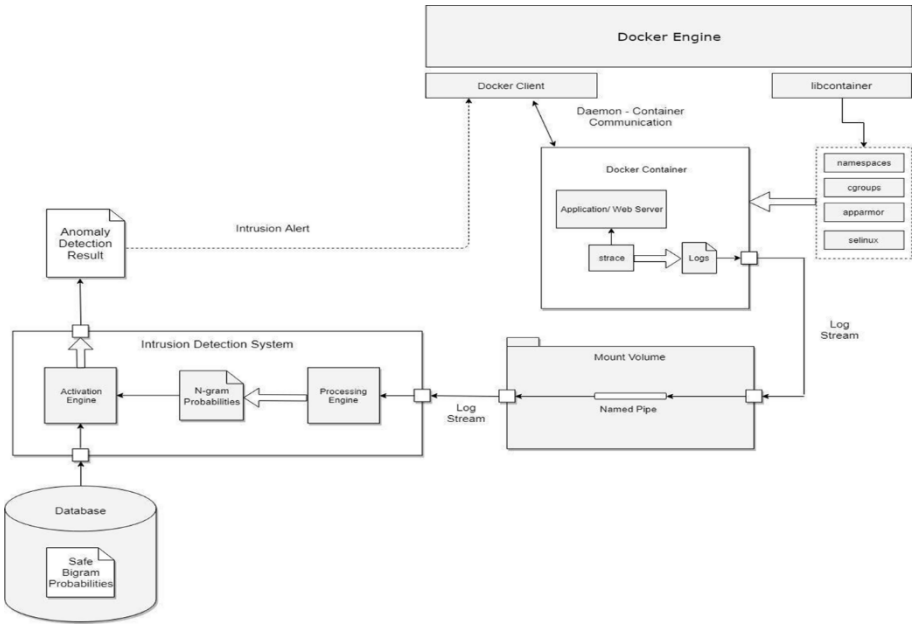


Fig. 1. Estrutura do sistema de detecção de intrusão em tempo real proposto

O HIDS funciona em dois modos: Normal e Detecção. No modo normal, o strace traça a aplicação e as probabilidades de bigram destas seqüências rastreadas de chamadas ao sistema são armazenadas em um banco de dados. Estas seqüências são denominadas "seqüências seguras". No modo de detecção, o sqlmap é usado para injetar cargas de trabalho maliciosas periodicamente à medida que e quando o strace estiver funcionando no modo normal. As probabilidades de N-grama são então calculadas para estas seqüências denominadas seqüências 'inseguras'.

A detecção de uma intrusão é baseada em um mecanismo de limiar. O IDS mantém verificando cada probabilidade de n-grama com as probabilidades armazenadas no banco de dados. Se o n-grama não existir no banco de dados, ou se a diferença entre as probabilidades do n-grama observado e armazenado estiver além de um certo limite, então marque o comportamento como possivelmente anômalo. Se o número de bandeiras atingir um determinado limite, então o IDS afirma que a atividade é maliciosa e dá a opção de continuar rodando o contêiner ou de encerrar completamente a operação. Os valores limiares são obtidos
observando as probabilidades de n-gramas tanto no modo normal como no modo de detecção e anotando a maior diferença de probabilidade entre ambos os modos.

Esta abordagem resolve o maior problema enfrentado em [4, 7], já que as seqüências de chamadas de sistema que são anômalas como denotadas por sua representação de n-grama são automaticamente sinalizadas, em contraste com o agrupamento manual de subconjuntos de chamadas de sistema, e as seqüências de chamadas de sistema que constituem uma intrusão podem ser computadas de forma muito mais eficiente.

Em vez de trabalharmos com probabilidades de N-grama bruto, processamos o traço e operados com os Estimadores de Máxima Probabilidade (MLE) [13] e, em seu lugar, com o SGT (Simple Good Turing).

3.1 Estimador de Máxima Probabilidade

Por exemplo, para calcular uma probabilidade de bigram particular de chamada de sistema y dada a chamada de sistema x, vamos calcular a contagem de bigram C(xy) e normalizar a soma de todos os bigrams que compartilham a primeira chamada de sistema x

$$P(Wn/Wn - 1) = \frac{C(Wn - 1Wn)}{C(Wn - 1)} \quad (1)$$

Da mesma forma, usando MLE para calcular as probabilidades de n-grama

$$P(Wn/W1 \ldots .Wn - 1) = \frac{C(W1 \ldots .Wn)}{C(W1 \ldots .Wn - 1)} \quad (2)$$

A principal vantagem desta abordagem é a simplicidade, que é útil especialmente quando desejamos isolar o funcionamento do próprio IDS, ou seja, executar o IDS separadamente em um container. Neste caso, podemos fazer uso de uma rede "ponte" dentro da qual podemos permitir que dois contêineres Docker transfiram informações de forma segura do contêiner para o serviço de monitoramento do outro contêiner. A maior desvantagem desta abordagem, entretanto, é que o processamento e o uso de memória aumentam em comparação
para executar o IDS no sistema hospedeiro.

3.2 Boa Turing simples

O Simples Bom-Termo [14] baseia-se na intuição de que podemos estimar a probabilidade de eventos que não ocorrem com base no número de N-gramas que só ocorrem uma vez. De acordo com as regras da Boa Turing suavização, definimos o que é conhecido como "quências livres de freqüências" para uma determinada contagem. Em outras palavras, se um N-grama ocorre 'r' vezes, denota-se Nr como o número de vezes que a contagem 'r' ocorre em toda a distribuição de freqüência do conjunto de dados. A probabilidade de observar qualquer classe que tenha ocorrido 'r' vezes no futuro é igual a

$$Pr_1 = \frac{(r + 1) \times Nr +}{Nr} \quad (3)$$

Há uma grande desvantagem com o Good-Turing Smoothing, que é a de que, para grandes contagens de ocorrência, no entanto, a freqüência correspondente dos valores de freqüência tenderia a Zero mesmo que eles existam. Para resolver este problema, foi introduzida uma variante de Good-Turing, conhecida como Linear Good-Turing (LGT) Smoothing. Eles solucionam o problema
com freqüência de freqüência Nr para o grande 'r', calculando a média do grande Nr com o vizinho
valores que são zero na ocorrência. Assim, outro estimador é introduzido. Se q, r e t representam os índices consecutivos, então outro termo Zr é introduzido de tal forma que,

$$Zr = \frac{Nr}{0{,}5 \times (t - q)}$$

(4)

E uma vez feito isso, os valores resultantes são suavizados usando a regressão linear logarítmica para minimizar a variação que de outra forma seria grande usando apenas a suavização da Boa Turing. A proposta feita para o SGT (Simple Good-Turing) Smoothing foi que as estimativas normais do Good-Turing devem ser usadas se os valores forem consideravelmente diferentes das estimativas do LGT (Linear Good-Turing), e uma vez atingido o ponto de saturação, usar as estimativas do LGT em seu lugar.

4 Avaliação

4.1 Configuração ambiental

Para testar nossa abordagem, executamos uma aplicação web em contêineres, que é conhecida por ser vulnerável. Em nosso caso, estávamos executando a aplicação dvwa em um único container, que é usado por profissionais de segurança para testar suas habilidades em um ambiente legal. O sistema host é baseado no Ubuntu 16.04, e o host executa um script python contendo a lógica para nosso Sistema de Detecção de Intrusão. Usamos o MongoDB para armazenar as probabilidades de bigram para seqüências seguras de chamadas ao sistema. Uma nova funcionalidade é adicionada durante a construção do IDS. Inicialmente o Motor de Ativação aceitou uma pequena fração como uma entrada extra que denotou o limiar da comparação entre as probabilidades de N-grama atualmente calculadas no instante dado, bem como as probabilidades de N-grama armazenadas no Banco de Dados denotando seqüência segura. Isto significa apenas que este valor limite teve que ser passado manualmente e foi geralmente determinado pela observação dos desvios mínimos que ocorrem com as probabilidades de N-grama entre duas ou mais seqüências de chamadas de sistema que denotaram o modo normal de operação do recipiente. Em vez de passá-lo manualmente, agora usamos um Modelo de Regressão Linear Múltipla que toma as duas diferentes Probabilidades de N-grama como entrada e retorna a probabilidade-limiar atualizado que deve ser usado para detectar intrusões. Como o modelo de previsão de limiar faz uso de um modelo de regressão linear, a suposição que fazemos durante o modo normal (fase de treinamento) é que as probabilidades de N-grama (tanto as atuais como as seguras) são linearmente dependentes do novo valor de limiar.

A abordagem proposta foi executada em diferentes sistemas de computação de especificações diferentes, e foi observado que embora o número de n-gramas diminua com um aumento no valor de 'n', existe uma defasagem perceptível ao executar o sistema para grandes valores de 'n'. Este efeito só pode ser atribuído ao processamento adicional necessário para calcular os n-gramas presentes nos traços para um grande valor de 'n'. Para sistemas modernos, o melhor valor de n é encontrado girando em três e quatro
respectivamente.

4.2 Validação da abordagem

Para testar nossa abordagem, fizemos uso de conjuntos de dados UNM [9] que são amplamente utilizados com o único propósito de validar diferentes abordagens para a detecção de anomalias. Estes conjuntos de dados contêm chamadas de sistema geradas para diferentes tipos de programas, e diferentes intrusões, tais como ataques de Trojan, estouro de buffer, etc. Alguns dos traços normais fornecidos são

"sintéticos", referindo-se a traços coletados através da execução de um roteiro preparado, e alguns traços

são "ao vivo", referentes a vestígios coletados durante o uso normal de um sistema computador de produção. Para nossa conveniência, agrupamos a lista de chamadas de sistema geradas em cada traço pelos PIDs dos processos que os geram antes de criar os
n-gramas e probabilidades correspondentes. Para o conjunto de dados do sendmail, nós pré-processamos principalmente o daemon e os traços de log, pois no caso de uma intrusão, sabe-se que a maioria das mudanças são observadas nestes traços. As tabelas 3 e 4 descrevem o número de chamadas de sistema seguras e intrusivas, bigramas e trigramas considerados para cada conjunto de dados para testar nosso modelo, respectivamente.

Um significado importante em nossa análise é que, enquanto testamos nossa abordagem com os traços normais, a precisão tem uma variação muito baixa e intervalos como diferenças decimais entre 99 e 100%. A maior variação é observada apenas testando o modelo com
dados intrusivos.

Tabela 3. Número de arquivos seguros avaliados para cada conjunto de dados UNM

Nome do conjunto de dados UNM	Número de chamadas ao sistema	Número de bigramas	Número de trigramas
Sendmail sintético	1800582	1800232	1799882
Sendmail Cert	1576086	1575792	1575498
Syn_lpr	2398	2389	2380
Inetd	541	538	535
Stide	2013755	2013739	2013723

Tabela 4. Número de arquivos intrusivos avaliados para cada conjunto de dados UNM

Nome do conjunto de dados UNM	Número de chamadas ao sistema	Número de bigramas	Número de trigramas
Sendmail sintético	1119	1116	1113
Sendmail Cert	6504	6481	6458
Syn_lpr	164232	163231	162230
Inetd	8371	8340	8309
Stide	205935	205830	205725

4.3 Resultados

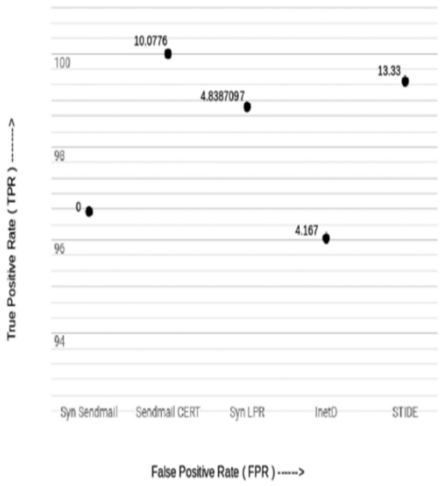
O desempenho do Sistema de Detecção de Intrusão proposto para recipientes Docker é expresso por uma matriz de confusão que contém duas linhas e duas colunas e informa o número de falsos positivos, falsos negativos, verdadeiros positivos e verdadeiros negativos, respectivamente. A matriz de confusão permite ainda uma análise mais detalhada através da estimativa dos valores de precisão, sensibilidade e especificidade. As medidas acima foram calculadas para o IDS usando estimadores MLE e SGT a fim de determinar o desempenho.

A Figura 2 e a Tabela 5 mostram o gráfico de dispersão e as métricas de desempenho do uso do MLE sobre o conjunto de dados UNM de nosso sistema.

Como pode ser visto pelos resultados obtidos em nosso sistema
apresenta altos valores de sensibilidade para todos os conjuntos de dados testados na faixa de 96-100%, e a Falsa Taxa Positiva é muito baixa, variando de 0-14%, o que significa que a taxa

de falsos alarmes que ocorrem é menor. Com relação à detecção de uma intrusão, observou-se que nosso sistema teve o melhor desempenho tanto com o sendmail sintético quanto com os conjuntos de dados STIDE e isto poderia ser atribuído ao grande tamanho de traços encontrados nestes conjuntos de dados.

Em comparação com os valores de precisão calculados ao usar a técnica de EML para ocorrências de N-grama, os valores de precisão SGT correspondentes (Tabela de Ref. 6 e Fig. 3) foram comparativamente menores. Isto porque a maioria dos conjuntos de dados forneceu apenas um ou dois traços para seqüências seguras para chamadas de sistema, bem como um único traço intrusivo também. O alisamento simples de Boa Turing funciona melhor para um vasto número de traços disponíveis para o modo normal de operação, de modo que pode fazer uma melhor estimativa sobre se um traço de entrada é seguro ou malicioso. STIDE foi o único conjunto de dados que forneceu um grande número de traços seguros de seqüências de chamadas do sistema .



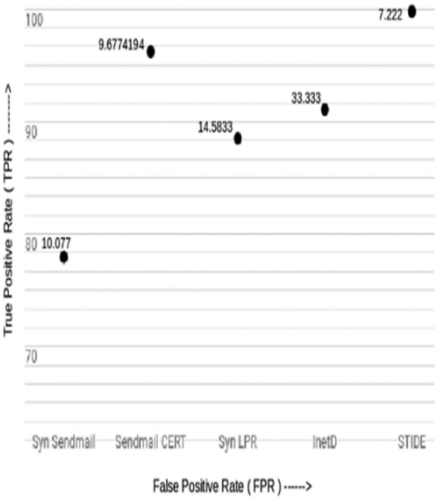


Tabela 5. Métricas de desempenho do IDS usando MLE sobre o conjunto de dados UNM

UNM conjunto de dados nome	Verdadeiro Positivo (TP)	Verdadeiro Negativo (TN)	Falsos Positivos (PF)	Falso Negativo (FN)	Precisão %	Sensibilidade	Especificidade
Sendmail sintético	57	123	0	2	98.9	96.61	100
Sendmail cert	53	118	6	0	96.61	100	95.15
Syn_lpr	87	46	2	1	97.79	98.86	95.83
Inetd	97	3	0	4	96.15	96.03	100
Stide	1012	156	24	6	97.49	99.41	86.67

Tabela 6. Métricas de desempenho do IDS usando o SGT sobre o conjunto de dados UNM

UNM conjunto de dados nome	Verdadeiro Positivo (TP)	Verdadeiro Negativo (TN)	Falsos Positivos (PF)	Falso Negativo (FN)	Precisão %	Sensibilidade	Especificidade
Sendmail sintético	46	116	7	13	89.01	77.96	89.92
Sendmail cert	51	112	12	2	92.09	96.22	90.32
Syn_lpr	77	41	7	10	87.40	88.50	85.41
Inetd	92	2	1	9	90.38	91.08	66.67
Stide	1016	167	13	2	98.74	99.80	92.77

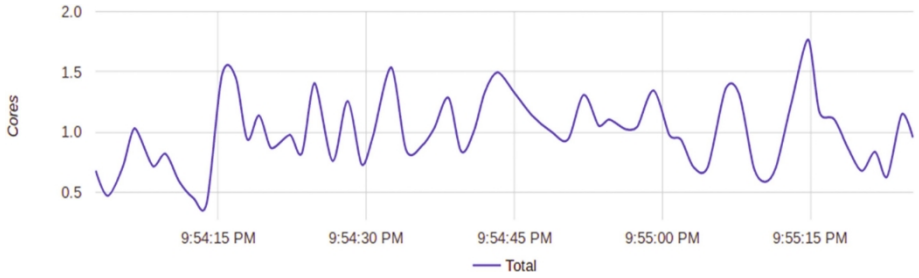


Fig. 4. Utilização da CPU de rodar o IDS no sistema host

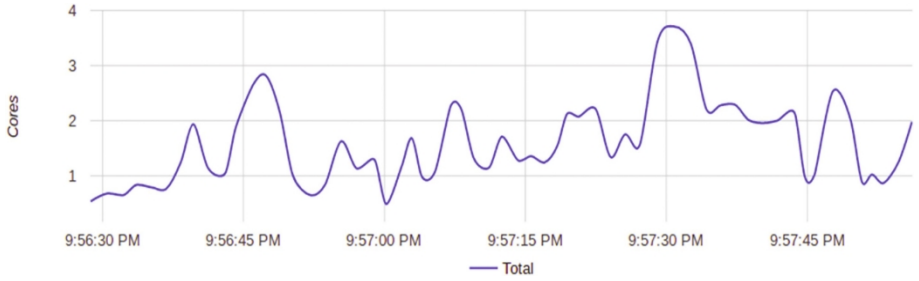


Fig. 5. Utilização de CPU de rodar IDS em container

5 Conclusão

Extraído dos resultados obtidos acima para nossa abordagem proposta, é evidente que o uso do estimador MLE para denotar as ocorrências é recomendado apenas quando um pequeno número de vestígios está disponível para o modo de treinamento, e o SGT é preferido de outra forma. Também observamos a utilização da CPU durante a execução do IDS em um recipiente separado em comparação com a execução no próprio sistema hospedeiro e descobrimos que a utilização da CPU é mais no primeiro caso do que no segundo (Figs. 4 e 5).

Uma grande vantagem de rodá-lo no sistema host é que podemos rodar vários containers Docker e um único IDS monitorando esses containers, portanto essa abordagem é muito mais escalável, e é considerada como parte de um trabalho futuro. Uma grande melhoria em nossa abordagem seria tornar possível a detecção de outros ataques aos contêineres Docker, tais como Container Breakout (que envolve o escalonamento automático dos privilégios do usuário sem qualquer solicitação de permissão), Cross-site Request Forgery (CSRF), injeção de XSS e detecção de malware no contêiner também. O trabalho futuro também inclui a extensão do mecanismo Probabilístico de n-grama para a detecção de intrusão para uma implementação de NIDS (Network-Based Intrusion Detection System), que significa analisar a carga útil dos pacotes de rede da mesma maneira probabilística, bem como mapear a implementação para uma técnica padrão de Machine Learning, especialmente um Modelo Bayesiano ou um Modelo Markov Escondido (HMM).

Referências

1. Gupta, S., Kumar, P..: Sistema cum esquema de detecção de execução de programas maliciosos leves para nuvens. Inf. Secur. J. Global Perspect. 23(3), 86–99 (2014)
2. Gupta, S., Kumar, P..: Uma abordagem imediata baseada na seqüência de chamadas do sistema para detectar execuções maliciosas de programas em ambiente de nuvem. Wireless Pers. Commun. 81(1), 405– 425 (2015)
3. Abed, A.S., Clancy, C., Levy, D.S.: Sistema de detecção de intrusão para aplicações utilizando recipientes linux . In: Foresti, S. (ed.) STM 2015. LNCS, vol. 9331, pp. 123-135. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24858-5_8
4. Koucham, O., Rachidi, T., Assem, N.: Detecção de intrusão no host usando argumentação de chamada de sistema - baseado em clustering combinado com a classificação Bayesiana. Em: 2015 SAI Intelligent Systems Conference (IntelliSys), Londres, pp. 1010-1016 (2015)
5. Jurafsky, D., Martin, J.H.: "Language Modeling with Ngrams" Discurso e linguagem Processamento, Cap. 4 (2016)
6. Wang, K., Stolfo, S.J.: Detecção de intrusão na rede baseada em carga útil anômala. In: Jonsson, E., Valdes, A., Almgren, M. (eds.) RAID 2004. LNCS, vol. 3224, pp. 203-222. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30143-1_11
7. Rachidi, T., Koucham, O., Assem, N.: Detecção de intrusão de host de dados e fluxo de execução combinados usando o aprendizado de máquina. In: Bi, Y., Kapoor, S., Bhatia, R. (eds.) Intelligent Systems e Applications. SCI, vol. 650, pp. 427-450. Springer, Cham (2016). https://doi.org/10. 1007/978-3-319-33386-1_21
8. Assem, N., Rachidi, T., El Graini, M.T.: Detecção de intrusão usando o classificador Bayesian para seqüências de chamadas de sistema arbitrariamente longas. IADIS Int. J. Comput. Sci. Inf. Syst. 9, 71–81 (2014)
9. Departamento de Informática, Centro de Engenharia Farris. Computer Immune Systems Data Sets (1998) http://cs.unm.edu/*immsec/data/synth-sm.html. Acesso em 21 de abril de 2013
10. Chiba, Z., Abghour, N., Moussaid, K., El Omri, A., Rida, M.: Uma pesquisa de intrusão sistemas de detecção para ambiente de computação em nuvem. In: 2016 International Conference on Engineering & MIS (ICEMIS), Agadir, pp. 1-13 (2016)
11. Sukhanov, A.V., Kovalev, S.M., Stýskala, V.: Aprendizagem avançada de diferenças temporais para detecção de intrusão. IFAC-PapersOnLine 48, 43-48 (2015). https://doi.org/10.1016/j.ifacol. 2015.07.005. Este trabalho foi apoiado pela Fundação Russa para Pesquisa Básica (Subsídios No. 13-07-00183 A, 13-08-12151 ofi_m_RZHD, 13-07-00226 A, 14-01-00259 A e 13-07-13109 ofi_m_RZHD) e parcialmente apoiado pela subvenção da SGS nº SP2015/151,

VŠB - Universidade Técnica de Ostrava, República Tcheca

12. Hubballi, N., Biswas, S., Nandi, S.: Sequencegram: n-grama modelagem de chamadas de sistema para detecção de anomalias baseadas em programa . Em: 2011 Third International Conference on Communi- cation Systems and Networks (COMSNETS 2011), Bangalore, pp. 1-10 (2011)
13. Jurafsky, D., Martin, J.H.: Processamento da fala e da linguagem. Direitos autorais © 14. Todos os direitos
reservado. Esboço de 1º de setembro de 2014 (2014)
14. Gale, W.A., Sampson, G.: Estimativa de freqüência sem rasgos. J. Quant. Linguista. 2(3), 217–237 (1995)