Container Intrusion Detection.

SK Saqlain Mustaq
Center for Cloud Computing And Big Data.
PES University.
Bengaluru, India
sksaqlain25@gmail.com

Sumair Ahmed Shariff
Center for Cloud Computing And Big Data.
PES University.
Bengaluru, India
sumairshariff98@gmail.com

Suraj K A
Center for Cloud Computing And Big Data.
PES University.
Bengaluru, India
kasuraj98@gmail.com

Dr Sanchika Gupta
Center for Cloud Computing And Big Data.
PES University.
Bengaluru, India
dr.sanchikagupta@gmail.com

***Abstract*—Linux Containers is an OS level virtualization in which each container shares the same OS kernel and isolate each application running in the container from each other. They run their own init processes independent of each other, filesystems etc. which are virtualized using the root OS. In comparison to a Virtual Machine, Linux containers split the use of a single kernel by utilizing namespaces that run on a central host OS. Linux containers are gaining popularity in both individual and industrial use, many of which are integrated with real-time applications that may contain sensitive data that cannot be revealed, thus resulting in a real-time intrusion detection that indicates whether any sort of intrusion that is being conducted over a specific Linux container. In this paper we introduce a new model for Linux container-based intrusion detection system. The basic idea is to use frequency of system calls followed by an auto-encoder which is trained over only positive data that is non-intrusive in nature.**

***Index Terms*—IDS, SOM, Auto-encoder, Frequency of system calls, Host based intrusion detection system.**

I. Introduction

Intrusion detection is an important and active area of research in cyber security. In general, the techniques for container intrusion detection fall into two major categories depending on the modeling methods used: misuse detection and anomaly detection. Misuse detection is usually conducted by signature matching of known set of attacks, whereas anomaly detection is done by tracing system calls in case of Host Intrusion Detection System (HIDS) or tracing of network packets in case of Network Intrusion Detection System (NIDS). There are multiple sources of attacks that can originate from multiple locations in one case the attack originates from outside the host attacking the host kernel and/or the guest container, another source of attack comes from another container residing on the same host but attacking other container on the same host and the other class of attacks when the container attacks the host kernel.

II. Related Work

A. Previous work of the previous papers

When it comes to intrusion detection the way data is collected and processed matter over all the other factors. In case of HIDS utilizing system call traces for anomaly detection has been previously applied at the process level [1] and has shown good results. There are two basic approaches to anomaly detection: sequence-based approach and frequency-based approach. The sequence-based approach keeps track of system call sequences in a database of normal behavior. The frequency based approach drops the order of the system calls while keeping the frequency of occurrence of each distinct system call in a particular sequence length. By not storing order information of the system call sequence, frequency-based techniques require much less storage space while providing better performance and accuracy [2].

A number of intrusion detection system used sequence of system calls to train a Hidden Markov Model (HMM) classifier [5]. The system differs in technique used to raising anomaly signal. [5], for example, indicate that the pattern is anomalous when the probability of the whole sequence is below certain specific threshold. On the other hand, some system is trained for declaring a sequence as anomalous when the probability of one system call within a sequence is below the threshold.

One of the places where frequency of system calls has been used is in BoSC(bags of systems calls) [2], Particular advantages associated with the use of bags of systems calls, as opposed to sequences of system calls, are that it is computationally manageable [3]. One of the techniques that is used for HIDS is [4] tracing of system calls, a system employs a background service running on the host kernel to monitor system call between any Linux containers and the host kernel on starting a new container on the host kernel triggers the service, which uses the Linux strace tool to trace all system calls issued by the container to the host kernel. The strace tool reports system calls with their originating process

ID, arguments, and return values [4]. The processed system calls are then updated using a sliding window of size 10 to generate frequency of system calls, in order to generate bag of system calls the sliding window is checked whether the same frequency of system calls pattern occurs if so then the value associated to that system call pattern is incremented by one thus resulting in a database of BoSC [4]. For detection purpose a sliding window of the same size is passed over the system call trace if the current set of BoSC is not present in the database of BoSC which was trained for normal behavior only then the trace is declared as anomalous [4].

Typically, system intrusion is detected by examining the data trail left by user and searching for abnormal user behavior. Data Mining has become a very useful technique to reduce information overload and improve decision making by extracting and useful relationships and patterns from the extensive data. The extracted information is used to predict, classify, model, and summarize the data being mined. In the paper [7], a supervised learning algorithm called K-nn was implemented, training the model with labeled data in order to predict whether the trace is anomalous or non-anomalous in nature. In the paper [8], the author has implemented a model called Linear Discriminant Analysis and Linear regression for binary classification. Linear Discriminant Analysis is a dimensionality reduction algorithm and Linear regression is a classifier Where the model was trained with labeled data then used to predict whether the input data was anomalous or not. And some other techniques that are implemented are using SVM with swarm intelligence [9], using k-means and Naive Bayes classification [10] trained over KDD Cup '99 benchmark dataset.

B. Summary of previous semester work(2017-2018)

Implemented both NIDS and HIDS. The setup included a docker container with a minimal ubuntu base image with MySQL installed in it, a small web application was deployed in the container which used MySQL as a backend. The web application used flask to communicate between front end and the backend.

To detect anomaly SOM(self-organizing maps) a neural network was trained.SOM is an unsupervised learning algorithm which is used for cluster and classification. A SOM consists of nodes or neurons each of which are initialized with random weights. During the training phase, when an input vector is given to the SOM, the distance of this vector from all the neurons are calculated and the vector is mapped to the closest one. With a suitable neighborhood function, the positions of the remaining neurons are adjusted based on certain parameters. Classification of a test vector is based on whether the distance of a test vector from a neuron exceeds that of the threshold (maximum distance between a training vector and the corresponding neuron) or not. The process of intrusion detection was done in two phases: System Call Analysis, Network Packet Analysis.

Simulation of normal behavior was done with a simple python script which sends multiple post requests to mimic the working of the web app. Simulation of abnormal behavior was done with w3af, a web application attack framework, an open source web vulnerability scanner and the python script. Normal behavior was simulated and w3af was used to attack the web app (attacks such as sql injection, buffer overflow, xss etc.).System calls for the web server and the database server were traced separately using the built-in Linux tool, strace, from the host machine. Once the datasets were ready, the SOM was trained with the normal behavior only. Abnormal behavior was said to be detected if an input vector was flagged as malicious within the same time frame as that of the attack. Network packets for the web server and the database server were traced using socket programming in python to capture TCP headers of the incoming TCP packets, only important features of the TCP header were considered and the remaining features were neglected the resultant values of the input vector were normalized between the values 0-1 in order to train SOM similar to system call analysis. The trained SOM model's accuracy was around 99% with the detection of anomalous behavior within 10 seconds.

III. Proposed work

In order to implement Host intrusion detection system a docker container was set up with a mysql:5.7 image obtained from the docker hub, then a database (employees database) was install inside the MySQL container. In order to generate data sets the container was tested with mysqlslap and sqlmap. Mysqlslap is a diagnostic program designed to emulate client load for a MySQL server The Mysqlslap runs queries similar to normal behavior exhibited in a database whereas sqlmap is an automated SQL injection and database takeover tool which is used to exhibit abnormal behaviors. whenever sqlmap or mysqlslap was run the system calls of the MySQL daemon of the container was traced by using strace tool a Linux system trace tool, once the sequence of system calls were collected the data was processed using a python script where a sequence of length of 50 each was taken and frequency of each unique system call was taken within that particular sequence ,a total of 40 unique system calls were registered and given an index value in the index list with an additional index for some miscellaneous system call. In order to check whether the data processing algorithm works, the dataset was divided into training and testing data, an ML model SVM and k-nn was trained with the training data and on validating with the testing data an accuracy of 99% was obtained. Since there are multiple number of anomalies which are unknown or cannot be simulated by sqlmap thus using an supervised learning algorithm for anomaly detection is not advisable. Thus to overcome this drawback we used Autoencoders. Autoencoders are artificial neural networks capable of learning efficient representations of the input data, called codings, without any supervision that is the training set is unlabeled. These codings typically have a much lower dimensionality than the input data, making autoencoders useful for dimensionality reduction. More importantly. Lastly, they are capable of randomly generating new data that looks very similar to the training

data, thus this principle is used for classification [6] whether the give input vector is anomalous or not.

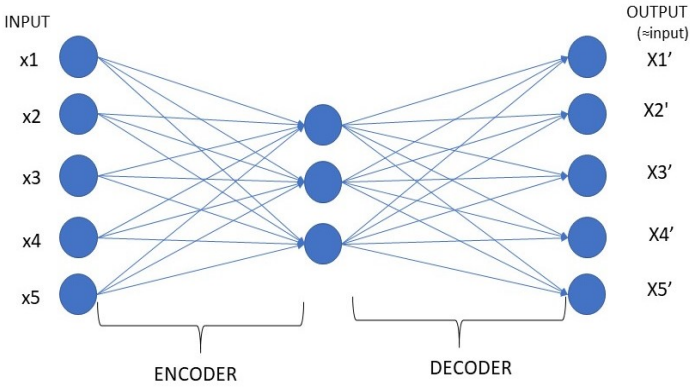

Fig. 1.

The Autoencoder consists of a encoder part and a decoder part as described in the fig.1 . An autoencoder has the same architecture as the Multi-layer Perceptron, except that the number of neurons in the outer layer must be equal to the number of inputs. The inner layers are always less than the outer layer as they do the job of data compression and feature recognition the outputs are often called the reconstructions since the autoencoder tries to reconstruct the inputs, based on the how the outputs are different from the inputs the weights of the inner layer are adjusted using an optimization algorithm.

For classifying a given vector as anomalous or non-anomalous the autoencoder was trained with just normal datasets, the autoencoder learns to reconstruct positive data, but does not learn to reconstruct negative data. Since negative data have a different pattern associated to it that are different from those of the positive data, reconstruction of negative data might result in the output which is entire different from the input. At testing time, therefore, reconstruction of positive data will succeed, whereas reconstruction of negative data will fail, and this success or failure is the criterion used in classifying whether the given input vector is anomalous not but before that a particular threshold was set for the rms error between the input vector and the output which was taken to be as the max of the rms error when the autoencoder was tested with the positive data. On testing with the negative dataset that is anomalous the algorithm worked with an accuracy of nearly 95%.

IV. Conclusion

The idea of using an autoencoder for predicting whether the given input vector is anomalous or not is new in the fields of container intrusion detection. Even though many ML algorithms have been implemented but most of it is supervised learning that is they require labeled datasets for learning which is expensive as to get as well as to train and the number of different kinds of attacks changes over time in that case the algorithm has to be trained with new set of data where as using an unsupervised learning algorithm the model can be trained with normal datasets only anything that is predicted with the least accuracy becomes an abnormal behavior. Our future work includes making the model dynamic and real time in nature and work on improving the accuracy of the system and also working on online intrusion detection system.

References

[1] Forrest, Hofmeyr, Somayaji, Longsta: A sense of self for unix processes.IEEE Symposium on Security and Privacy. pp. 120–128.
[2] Fuller, D., Honavar, V.: Learning classiers for misuse and anomaly detection using a bag of system calls representation. In: Proceedings of the Sixth Annual IEEE Systems, Man and Cybernetics (SMC) Information Assurance Workshop. pp. 118– 125. IEEE (2005)
[3] Alari, S., Wolthusen, S.: Detecting anomalies in IaaS environments through virtual machine host system call analysis. In: International Conference for Internet Technology And Secured Transactions. pp. 211–218. IEEE (2012)
[4] Intrusion Detection System for Applications using Linux Containers.Amr S. Abed, Charles Clancy, and David S. Levy .
[5] Wang, W., Guan, X.H., Zhang, X.L.: Modeling program behaviors by hidden markov models for intrusion detection. In: Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on. vol. 5, pp. 2830–2835. IEEE (2004).
[6] A Novelty Detection Approach to Classification;Nathalie Japkowicz, Catherine Myers and Mark Gluck.
[7] Intrusion Detection Using k-NearestNeighbor .M.Govindarajan, R M.Chandrasekaran
[8] Intrusion Detection Systems using Linear Discriminant Analysis and Logistic Regression.Basant Subba, Santosh Biswas, Sushanta Karmakar.
[9] Intrusions Detection Based On Support Vector Machine Optimized with Swarm Intelligence,Adriana-Cristina Enache, Victor Valeriu Patriciu.
[10] Intrusion Detection based on K-Means Clustering and Naïve BayesClassification;Z. Muda, W. Yassin, M.N. Sulaiman, N.I. Udzir