An Adaptive Real-Time Architecture for Zero-Day Threat Detection

Antonio Gonzalez Pastana Lobato*, Martin Andreoni Lopez*†, Igor Jochem Sanz*,
Alvaro A. Cardenas‡, Otto Carlos M. B. Duarte*, and Guy Pujolle†

*Universidade Federal do Rio de Janeiro - GTA/COPPE/UFRJ - Rio de Janeiro, Brazil
† Sorbonne Universités, UPMC Univ Paris 06, Laboratoire d'Informatique de Paris 6 - Paris, France
‡ University of Texas at Dallas, Cy-Phy Security Lab - Texas, USA

Emails: {antonio, martin, sanz, otto}@gta.ufrj.br, alvaro.cardenas@utdallas.edu, guy.pujolle@lip6.fr

***Abstract*—Attackers create new threats and constantly change their behavior to mislead security systems. In this paper, we propose an adaptive threat detection architecture that trains its detection models in real time. The major contributions of the proposed architecture are: i) gather data about zero-day attacks and attacker behavior using honeypots in the network; ii) process data in real time and achieve high processing throughput through detection schemes implemented with stream processing technology; iii) use of two real datasets to evaluate our detection schemes, the first from a major network operator in Brazil and the other created in our lab; iv) design and development of adaptive detection schemes including both online trained supervised classification schemes that update their parameters in real time and learn zero-day threats from the honeypots, and online trained unsupervised anomaly detection schemes that model legitimate user behavior and adapt to changes. The performance evaluation results show that proposed architecture maintains an excellent trade-off between threat detection and false positive rates and achieves high classification accuracy of more than 90%, even with legitimate behavior changes and zero-day threats.**

I. Introduction

Attackers keep changing their behavior and searching unknown vulnerabilities to bypass traditional security mechanisms. Therefore, zero-day attacks are increasing by more than 125% each year, and we are expecting one new zero-day attack per day by the year 2021 [1]. This kind of attack is hard to identify, because information becomes available, on average, 312 days after the attack occurs [2]. Current security systems, such as Security Information and Event Management (SIEM) are not effective, since 85% of network intrusions are detected weeks after they occurred [3]. Therefore, a long threat detection time makes any kind of defense unfeasible. Honeypots complement traditional security mechanisms by performing two main tasks: confuse attackers in order to protect the infrastructure; and collect information about attacker behavior and attack patterns [4]. Hence, honeypots are able to spot zero-day attacks and give insights on attacker actions and behavior.

In this paper, we propose an adaptive real-time threat detection architecture composed by the use of distributed stream processors with honeypot collected data to train in real-time machine-learning algorithms. We consider all honeypot traffic as malicious, since there are no useful resources [5], [6], and we use the collected data to update the threat detection algorithms. Our architecture adapts itself to attacker behavior changes and learns zero-day attacks. The proposed architecture features both online supervised threat classification and unsupervised anomaly detection to better protect the network. Moreover, we propose distributed stream processing for the detection schemes to achieve the minimum detection time, to gain scalability while processing huge amounts of data, and to extract security analytics [7]. We design, develop, and evaluate the performance of four detection schemes using real datasets, one created in our lab and the other containing data from one of the major network operators in Brazil to prove the efficiency of our architecture. The results show that the online classification schemes are capable to learn unknown threats from the honeypot data and to maintain high accuracy rates, higher than 90%, as traffic streams arrive. Furthermore, the implemented anomaly detection schemes present an exceptional trade-off between false positive and threat detection rates, proving that the architecture can model well legitimate user behavior, even with its ever-changing nature.

The rest of the paper is organized as follows. Section II discusses related work. The proposed Threat Detection Architecture is presented in Section III, along with the developed threat detection schemes. Section IV discusses the results. Finally, Section V concludes the work.

II. Related Work

Big data threat detection architectures are mainly based in batch processing schemes. Bostani and Sheikan use the popular implementation of MapReduce, Hadoop, to implement an unsupervised anomaly detection algorithm [8]. The work uses Optimum-Path Forest (OPF) algorithm to project clustering models and detect anomalous behaviors. Nevertheless, the paper only focuses on two specific Internet of Things (IoT) attacks, sinkhole and selective-forwarding, disregarding unknown threats or zero-day attacks. A similar architecture is proposed by Singh *et al*. The authors also use MapReduce to perform big data analytics. In this case, big data analytics aim

to detect Peer-to-Peer botnets [9]. A limitation of this approach is that the analysis is executed offline. Rathore *et al.* [10] propose a Hadoop based real-time Intrusion Detection System (IDS) for detecting threats in high-speed networks. Unlike our proposal that trains the models and classifies threats in real-time, they use the term real-time referring to stream processing classification with offline training. Moreover, their proposal and many other conventional traffic classification papers are based on artificial and outdated datasets, such as the KDD99 dataset. In this paper, we use two real-world traffic datasets collected recently.

Many zero-day attacks detection approaches use unsupervised algorithms, such as clustering and outliers detection. Owerzarski *et al.* [5] propose an unsupervised clustering algorithm for anomaly characterization of honeypot traffic. Their approach assumes all honeypot traffic is malicious and classifies it into different anomalies. Wang *et al*. propose a seed expanding (SE) cluster technique for earlier attack detection [11]. The unsupervised algorithm clusters 14 features outperforming traditional KMeans algorithm. However, the SE algorithm is unable to adapt in real-time to zero-days threats. In contrast, our work performs both supervised and unsupervised detection, detecting threats among legitimate network traffic. Saied *et al.* propose the use of Artificial Neural Networks (ANN) to detect Distribute Denial of Service (DDoS) attacks [12]. The approach presents up to 98% of accuracy, when compared with traditional methods. The training process, however, is realized offline and cannot adapt to changes and zero-day threats.

Unlike the previously cited papers, we propose a specific architecture that allows hybrid real-time supervised threat classification and unsupervised anomaly detection. Hence, our architecture learns and adapts to new threats captured by the honeypots and monitors legitimate user behavior to detect anomalies. Moreover, we implement the schemes using stream processing. Stream processing enable us to detect threats in real-time and to scale using distributed processing.

III. Proposed Threat Detection Architecture

We propose an architecture that uses live feeds of honeypot data to train real-time threat detection schemes. Moreover, our proposal considers the use of stream processing technology to speed up detection and enable counter measures. Figure 1 shows the proposed architecture which is divided in three modules: the data capture module, the distributed processing module, and the visualization module. The data capture module deals with the incoming data from different data sources in real-time. The processing module uses real-time stream processing to detect threats, train online detection algorithms, and uses offline processing to tune the threat detection mathematical models improving the online detection. Finally, the visualization module provides threat detection alerts in real time.

The input to the data capture module is heterogeneous, coming from several traffic analyzers and distributed honeypots. In order to capture data from different sources, we deploy the traffic analyzer at different network locations. Network flows are characterized by synthesizing packet information and grouping them in time windows. Several honeypots compose the second data source. We collect honeypot data to train online classification schemes that adapt to new threats. In this work, we use the low-interaction open-source honeypot `Dionaea` to emulate decoy services and we employ the packet analyzer `tcpdump` to capture the incoming data traffic. Hence, we perform several attacks on the honeypot and use the collected data to adapt our detection schemes in real-time. We consider the honeypot threats as zero-day threats since the classification algorithms have no prior knowledge of their signatures when the training process initializes. To avoid data overload, we place a buffer that adapts different generation and processing rates using the publish/subscribe model.

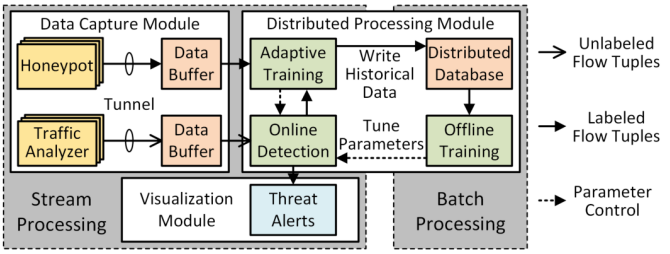


Figure 1: Proposed architecture for real-time stream processing and historical data batch processing. Honeypot data feed and train the detection models in real-time, adapting to zero-day threats.

The distributed processing module is the core for our real-time threat detection architecture. In our previous work [13], we compare the three most famous open-source tools for stream processing. We chose `Storm` due to the higher processing throughput and the possibility to process samples in parallel. `Storm` also offers a distributed fault tolerant stream processing framework and processes data in memory, ensuring low latency. In `Storm` we implement the online detection algorithms and the adaptive training, this two modules exchange parameters to detect threats.

Results are stored in a dynamic database for posterior analysis. The stored data contain the information gathered during the threat detection and can be processed offline to tune parameters for real-time models. At this point, there is a feedback, since the parameters calculated offline from historical data improve our adaptive real-time processing schemes.

A. Proposed Adaptive Threat Detection Schemes

We propose four adaptive threat detection schemes based on online machine learning algorithms, i.e., schemes that adapt their parameters in real time, as the data streams arrive. The first two online classification algorithms aim to classify the attacks based on threat behavior captured by the honeypots, whereas the other two are anomaly detection algorithms, to detect suspicious behavior in the network.

Real-time Classification: In our architecture, we label all data coming from honeypots as threats, since there are no real service in them and all access is meant to exploit a vulnerability intentionally installed [5], [6]. Therefore, all

flows that arrive in the honeypots are labeled as threats and are used by the algorithms to update the parameters, since online classification schemes require labeled data to be updated. This real-time data feed ensures adaptive behavior to detect new threats. Then, whenever an attacker performs a new threat or changes his behavior, the classification models update.

We propose a data collection scheme that ensures adaptability for both legitimate and malicious behavior. The architecture considers as normal data the incoming flows from the traffic analyzers in the network during the setup time, which is a period that the network usage is monitored to guarantee that all the flows are legitimate. After this time, if an incoming flow from the network analyzer is classified as a threat, an alert is sent to the visualization module and the parameters are not updated. If the flow is considered normal, the algorithms update their parameters, adapting to network behavior changes. Hence, the architecture adapts to acceptable normal behavior changes in the network and, at the same time, learns new threats performed in the network honeypots. Our proposed data collection scheme is capable of learning and detecting zero-day attacks, while these attacks would pass unnoticed by signature-based intrusion detection systems or offline classification schemes.

1) Stochastic Gradient Descent with Momentum: The Stochastic Gradient Descent (SGD) algorithm is a stochastic approximation of Gradient Descent, in which the gradient is approximated by a single sample. In our application, we consider two classes, normal and threat. Therefore, we use the Sigmoid Function, expressed by

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^\intercal x}}, \tag{1}$$

to perform logistic regression. In the Sigmoid function, low product values of the parameters $\theta^\intercal$ times the sample feature vector x return zero, whereas high values return one. When a new sample $x_{(i)}$ arrives, the SGD evaluates the Sigmoid function and returns one for $h_\theta(x_{(i)})$ greater than 0.5 and zero otherwise. This decision presents an associated cost, based on the real class of the sample $y_{(i)}$. The cost function is defined in Equation 2. This function is convex and the goal of SGD algorithm is to find its minimum, expressed by

$$J_{(i)}(\theta) = y_{(i)} log(h_\theta(x_{(i)})) + (1 - y_{(i)}) log(1 - h_\theta(x_{(i)})). \tag{2}$$

On each new sample, the algorithm takes a step toward the cost function minimum based on the gradient of the cost function.

Algorithm 1 shows the developed SGD algorithm. At each incoming sample vector $x_{(i)}$, the architecture determines the class $y_{(i)}$ based on the source. If it comes from a honeypot, the label is 1, while if it comes from a traffic analyzer the label is 0. If the sample comes from a traffic analyzer and SGD predicts it as a threat, it sends an alert. Otherwise, it updates the parameters based on the cost function gradient. The term $\Delta\theta$ is the momentum and has the value of the previous parameter update. In SGD context, this term considers the past move when updating the parameters θ. The parameters α and β are periodically updated in an offline manner based on the historical cost for each sample, due to the batch processing feedback to the stream processing components, shown in Figure 1.

input : Incoming flow characteristics x, Class y
output: Predicted Class $predict$, training parameters θ

Initialize θ, $\Delta\theta$, α, β;
for $i \leftarrow 1$ **to** m **do**
  $h_\theta(x_{(i)}) = \frac{1}{1+e^{-\theta^\intercal x_{(i)}}}$;
  $predict = round(h_\theta(x_{(i)}))$;
  if $predict == 1$ *and* $y_{(i)} == 0$ **then**
    Send Alert;
  else
    $\theta = \theta - \alpha \nabla J_{(i)}(\theta) + \beta \Delta\theta$;
    $\Delta\theta = \alpha \nabla J_{(i)}(\theta)$;
  end
end

Algorithm 1: Stochastic Gradient Descend with Momentum.

2) Online Support Vector Machine: The Support Vector Machine (SVM) is a binary classifier, based on the concept of a decision plane that defines the decision boundaries. A hyperplane constructed in a multidimensional space splits the data into classes. The online SVM algorithm uses a soft margin approximation with the convex hinge-loss function, given by

$$\max\{0, 1 - y\theta^\intercal x\}. \tag{3}$$

The objective of this algorithm is to minimize the loss function. Just like the previous algorithm, the SVM determines the class $y_{(i)}$ based on the source. If it comes from a honeypot the label is 1, while if it comes from a traffic analyzer the label is -1. When a traffic flow from a network sensor is classified as threat, an alert is sent and the model is not updated. Algorithm 2 shows the implementation of online SVM. The parameters α, λ are periodically updated based on the evaluation of the hinge-loss function over the samples.

input : Incoming flow characteristics x, Class y
output: Predicted Class $predict$, training parameters θ

Initialize θ, α, λ;
for $i \leftarrow 1$ **to** m **do**
  $predict = sign(\theta^\intercal x_{(i)})$;
  if $predict == 1$ *and* $y_{(i)} == -1$ **then**
    Send Alert;
  else
    if $y_{(i)}\theta^\intercal x_{(i)} > 1$ **then**
      $\nabla_{(i)} = \theta$;
    else
      $\nabla_{(i)} = -\lambda y_{(i)} x_{(i)}$;
    end
    $\theta = \theta - \alpha \nabla_{(i)}$
  end
end

Algorithm 2: Online Support Vector Machine.

Real-time Anomaly Detection: We propose two anomaly detection schemes to protect the network if a zero-day attack is performed first against a user. The anomaly detection provides the capability to discover zero-day attacks that are hard to detect because there are not yet any data about the attack. The proposed schemes are based on unsupervised algorithms that model legitimate behavior and detect threats by analyzing unexpected behaviors.

3) Normal Distribution: We propose anomaly detection by the sample feature distance from a normal distribution. Hence, anomalies are detected when the distance from a sample feature to the mean is greater than a threshold times the variance for at least one of the features. The real-time implementation requires anomaly detection as the streaming data is arriving. Therefore, an anomaly is detected if at least one of the following conditions is true for at least one feature j,

$$X_j > \mu_j + threshold * \sigma_j^2 \quad (4) \quad X_j < \mu_j - threshold * \sigma_j^2 \quad (5)$$

taking into account the means μ_j and the variances σ_j^2 calculated online. Furthermore, when a new sample arrives and it is not detected as an anomaly by conditions (4) and (5), the mean μ_j and variance σ_j^2 of each feature, defined by

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} X_j \qquad (6) \quad \sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}(X_j - \mu_j)^2 \quad (7)$$

are updated, considering the new sample. Current values of the total samples N and its sum are stored and incremented when a new sample arrives, considering each feature X_j. Consequently, the normal distribution parameters are always updated by samples considered normal, ensuring adaptability.

4) Entropy Time Series: We further propose anomaly detection by analyzing the entropy value of a sliding window of flows. The sample entropy expressed by

$$H(X) = -\sum_{i=1}^{N}(\frac{n_i}{S})\log_2(\frac{n_i}{S}) \qquad (8)$$

indicates the concentration degree of a characteristic, where S is the total number of observations, n_i is the number of observations within the value range i, and N is the number of ranges. When all values are concentrated in one range, $H(X)$ is equal to zero and when each value is in a different range i, the value of $H(X)$ is $\log_2(N)$. Then, given a series of observations X, the sample entropy summarizes the concentration degree in one value. We define a sliding window of 40 flows and calculate the value of $H(X)$ for each window, generating the time series. Another parameter determined in the set up phase is the range with most samples. These parameters are updated in real time. We observed that the normal traffic entropy values tend to be concentrated and the most frequent value tends to be in its center. Thus, we propose the detection by defining a threshold that limits the accepted distance of the entropy $H(X)$ to the most frequent range. The algorithm updates the most frequent value online, adapting to different networks behaviors.

IV. Results

The two best-known available datasets are DARPA [14] and KDD 99 [15]. However, these datasets present several limitations [16] because the traffic does not correspond to a real network scenario, since it was simulated. Furthermore, they are over 15 years old and do not represent current threats [17]. Therefore, we evaluate the adaptive detection based on two datasets, one captured by a major network operator in Brazil, NetOp dataset, and the other created in the GTA/UFRJ lab.

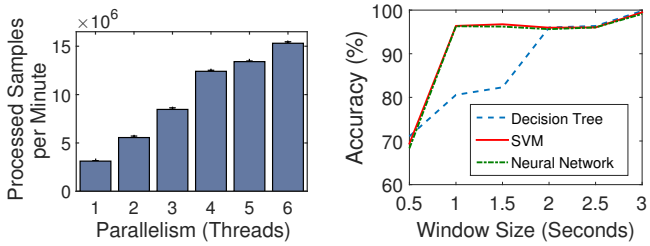
The Network Operator (NetOp) dataset is composed by real-access information from 373 residential broadband users from Rio de Janeiro for a period of one week [18]. Network traffic is anonymized due to privacy concerns. The traffic was filtered by an Intrusion Detection System (IDS), Suricata. We analyzed the logs from the IDS and the proportion of filtered attacks was around 15%. Therefore, in order to evaluate our proposed threat detection, we added real botnet malicious traffic captured in the work of García *et al* [19]. In the combined dataset, we keep 15% threat traffic proportion. The botnet data consists of 13 different scenarios of malware infection. The NetOp dataset contains 5,320,955 flows.

Another contribution of this work is the creation of the GTA/UFRJ dataset with real network traffic to evaluate the proposal. The dataset was created with a traffic capture from our lab, GTA at Federal University of Rio de Janeiro, containing both normal traffic and real network threats. We enrich the dataset with several attacks. Altogether, the dataset contains seven types of Denial of Service (DoS) and nine types of port scan. The DoS attacks are *ICMP flood*, *land*, *nestea*, *smurf*, *SYN flood*, *teardrop*, and *UDP flood*. The different types of probe in the dataset are *TCP SYN scan*, *TCP connect scan*, *SCTP INIT scan*, *Null scan*, *FIN scan*, *Xmas scan*, *TCP ACK scan*, *TCP Window scan*, and *TCP Maimon scan*. We perform the threats using tools from the *Kali Linux* distribution, which focus on penetrating testing. These attacks were labeled based on the destination, since the attacks were sent to honeypots. The GTA/UFRJ contains 154,187 flows with 30% of threat traffic. The GTA/UFRJ dataset can be obtained by emailing the authors.

For both datasets, we group packets into flows. A flow is defined as a packet sequence from the same IP source to the same IP destination during a time window. Each flow has 26 features, generated by the TCP/IP header data. The features are: TCP, UDP, ICMP and IP packet rate (4); number of source and destination ports (2); number of each TCP and IP flags (9); mean and variance of inter-packet arriving time and packet length (4); average TTL (1); average header and packet length (2); average offset and fragment offset (2); number of ICMP types and codes (2).

As mentioned before, we use `Storm` stream processing tool because it provides a high processing throughput, the possibility to process data in memory and samples in parallel, which ensure low latency. Figure 2a shows the number of processed samples per minute by our threat detection schemes

as we vary the parallelism parameter, i.e. the number of threads per component in Storm. With parallelism of four threads, we achieve more than 10 million processed samples per minute, showing the scalability potential of our architecture. We use historical data, gathered during the threat detection, to improve our adaptive real-time processing model. Figure 2b shows the detection accuracy in the GTA/UFRJ dataset. Based on these results, we use the 2-second flow time window in our sensors, because it is the shortest time to achieve high accuracy for all three offline algorithms.



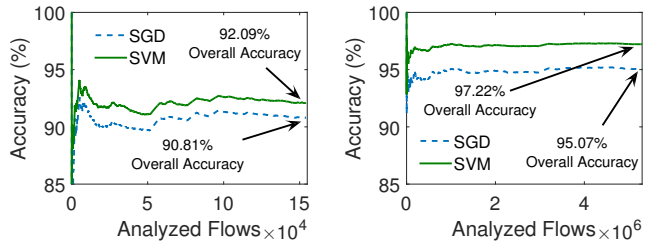
(a) Processing performance when varying the parallelism parameter.

(b) Accuracy for different flow time window sizes.

Figure 2: a) Online processing performance and b) Offline study of flow window size.

A. *Real-time Classification Results*

We achieve great adaptability with the proposed real-time honeypot-based architecture. Figure 3 shows the accuracy behavior over time for each incoming sample. Figures 3a and 3b show the results for each dataset. The accuracies for the implemented algorithms stabilize at high values, always above 90%, even when unknown threats appear in the datasets. At the beginning of the analysis, the methods do not know any threats. However, as honeypot data arrives, the methods learn the threats and achieve high threat detection rates. Since the detection methods do not know any of the 16 threats in the GTA/UFRJ dataset and the 13 threats in the NetOp dataset, these threats are considered zero-day threats that are learned and detected online as new samples arrive.



(a) GTA/UFRJ Dataset

(b) Network Operator Dataset

Figure 3: Accuracy for both proposed schemes as samples arrive. In both datasets, the accuracy stays stable even with new attacks and legitimate usage behavior changes.

Table I shows SGD precision and recall for both classes in the datasets. At the very beginning of the analysis, overfit occurs because the algorithm has few samples and adapts specifically to them, resulting in very high accuracy. As samples arrive, however, the SGD stabilizes and acquires great capability to generalize and adapt to new traffic samples, including new threats that appear in the traffic, ending with accuracies of 90.81% for the GTA/UFRJ dataset and 95.07% for the NetOp dataset. Despite threat detection rates of 76.2% and 65.7%, respectively, this algorithm has very low false positive rates of 2.7% and 0.5%. Low false positive rate is a well-desired characteristic, since it ensures confidence in the security alerts.

Table I: SGD Precision and Recall for both classes.

Dataset	Normal		Threat	
	Precision	Recall	Precision	Recall
GTA/UFRJ	90.2%	97.3%	92.5%	76.2%
NetOp	95.1%	99.5%	94.9%	65.7%

Table II shows SVM precision and recall for both classes. The overall accuracy is better than SGD, because SVM is a robust classifier that maximizes the margin to the hyperplane decision boundaries. Figure 3 also shows that the algorithm adapts really well to usage changes, maintaining high accuracy. SVM learns as threats appear in traffic, showing potential to learn zero-day threats. The overall accuracies are 92.09% for the GTA dataset and 97.22% for the NetOp. As Table II shows, the attack detection rate is higher than SGD, with values of 87.5% and 82.0% respectively. In addition, SVM also gets low false positive rates, with values of 5.9% and 0.5%. SVM presents an overall result better than SGD, since SVM achieves higher accuracy and threat detection rate. However, SGD is a suitable choice in networks that must present lower false positive rate.

Table II: SVM Precision and Recall for both classes.

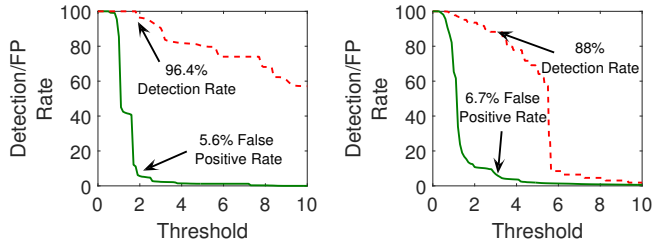
Dataset	Normal		Threat	
	Precision	Recall	Precision	Recall
GTA/UFRJ	94.5%	94.1%	86.8%	87.5%
NetOp	97.4%	99.5%	96.1%	82.0%

B. *Real-time Anomaly Detection Results*

Since both anomaly detection schemes are unsupervised and model normal user behavior, we only use the honeypot data to evaluate algorithm performance. The parameters are updated during the setup time based only in the data captured by the traffic analyzers. After this period, if a sample is considered normal, the parameters are updated. On the other hand, if it is considered malicious, an alert is sent and the algorithms parameters are not updated. To evaluate the algorithms, we use 70% of the normal data in the setup time and the other 30% to evaluate the false positives. The attack detection rate is obtained based on all the threats in the datasets.

Figure 4 shows the Normal Distribution results for different threshold values. For the GTA/UFRJ dataset, we observe that selecting a threshold of 2, the algorithm presents a very good trade-off between 96.4% of detection rate and 5.6% of false

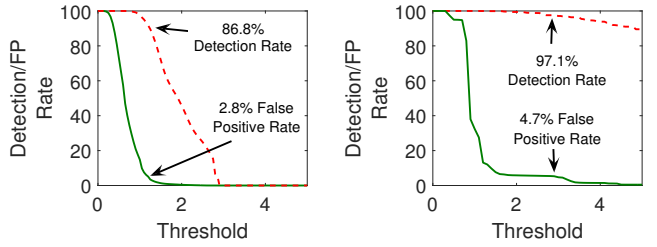
positive rate. The same behavior is presented for the NetOp dataset, with a good trade-off with a threshold of 3, presenting 88% of threat detection rate and 6.7% of false positive rate. The threshold parameter can be adjusted according to the desired network security level. For networks with high security requirements, this threshold should be small, resulting in high detection rate at the cost of a higher false positive rate.



(a) GTA/UFRJ Dataset. (b) Network Operator Dataset.

Figure 4: Normal Distribution Detection and False Positive rates for different threshold parameters. The threshold can be chosen considering different network security requirements.

Figure 5 shows the Entropy Time Series results for different thresholds. For small threshold values, the detection rate is very high, also resulting in a high false positive rate. For low values, the opposite occurs, a low false positive rate at cost also of a low detection rate. However, there are threshold values that present good trade-off between these rates. For the GTA/UFRJ dataset, the threshold 1.3 results in 86.8% of attack detection rate with only 2.8% of false positive rate, whereas for the NetOp dataset, the threshold of 3 gets a low false positive rate of 4.7% with a remarkably high attack detection rate of 97.1%.



(a) GTA/UFRJ Dataset. (b) Network Operator Dataset.

Figure 5: Entropy Time Series Detection and False Positive rates for different threshold parameters.

V. Conclusion

This paper proposes an adaptive real-time threat detection architecture that uses honeypot data to learn zero-day attacks. The architecture uses a hybrid detection scheme, with supervised online threat classification and unsupervised anomaly detection. Moreover, stream processing technology ensures low detection time and high processing throughput. We propose four threat detection schemes and evaluate the results for two real datasets. The supervised classification algorithms update their parameters considering honeypot real-time data collection to learn new attacks. Both algorithms achieve high accuracy, higher than 90%, adapting over time. The unsupervised anomaly detection algorithms adapt to legitimate behavior and have an exceptional trade-off between detection and false positive rates.

Acknowledgment

This research is supported by CNPq, CAPES, FAPERJ, and FAPESP (2015/24514-9, 2015/24485-9, and 2014/50937-1).

References

[1] Symantec, "Internet security threat report," Symantec Co., Tech. Rep. 04, 2016, accessed 20/09/17. [Online]. Available: https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf

[2] L. Bilge and T. Dumitras, "Before we knew it," in *ACM Conference on Computer and Communications Security - CCS '12*. ACM, 2012.

[3] P. Clay, "A modern threat response framework," *Network Security*, no. 4, pp. 5–10, 2015.

[4] M. Nawrocki, M. Wählisch, T. C. Schmidt, C. Keil, and J. Schönfelder, "A survey on honeypot software and data analysis," *arXiv preprint arXiv:1608.06249*, 2016.

[5] P. Owezarski, "A near real-time algorithm for autonomous identification and characterization of honeypot attacks," in *ACM Symp. on Information, Computer and Communications Security*. ACM, 2015, pp. 531–542.

[6] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao, "Statistical analysis of honeypot data and building of kyoto 2006+ dataset for NIDS evaluation," in *Building Analysis Datasets and Gathering Experience Returns for Security*. ACM, 2011, pp. 29–36.

[7] A. A. Cárdenas, P. K. Manadhata, and S. P. Rajan, "Big data analytics for security," *IEEE Security Privacy*, vol. 11, no. 6, pp. 74–76, Nov 2013.

[8] H. Bostani and M. Sheikhan, "Hybrid of anomaly-based and specification-based ids for internet of things using unsupervised opf based on mapreduce approach," *Computer Communications*, vol. 98, pp. 52–71, 2017.

[9] K. Singh, S. C. Guntuku, A. Thakur, and C. Hota, "Big data analytics framework for peer-to-peer botnet detection using random forests," *Information Sciences*, vol. 278, pp. 488–497, 2014.

[10] M. M. Rathore, A. Paul, A. Ahmad, S. Rho, M. Imran, and M. Guizani, "Hadoop based real-time intrusion detection for high-speed networks," in *IEEE GLOBECOM*, Washington, USA, 2016.

[11] J. Wang, L. Yang, J. Wu, and J. H. Abawajy, "Clustering analysis for malicious network traffic," in *IEEE ICC*, May 2017, pp. 1–6.

[12] A. Saied, R. E. Overill, and T. Radzik, "Detection of known and unknown DDoS attacks using artificial neural networks," *Neurocomputing*, vol. 172, no. Supplement C, pp. 385 – 393, 2016.

[13] M. Andreoni Lopez, A. Lobato, and O. C. M. B. Duarte, "A performance comparison of open-source stream processing platforms," in *IEEE GLOBECOM*, Washington, USA, Dec. 2016.

[14] R. P. Lippmann, D. J. Fried *et al.*, "Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation," in *DARPA Information Survivability Conference and Exposition.*, vol. 2. IEEE, 2000.

[15] W. Lee, S. J. Stolfo, and K. W. Mok, "Mining in a data-flow environment: Experience in network intrusion detection," in *ACM SIGKDD*. ACM International Conference on Knowledge Discovery and Data Mining, 1999, pp. 114–124.

[16] M. Tavallaee, E. Bagheri, W. Lu, and A.-A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *IEEE Symposium on Computational Intelligence for Security and Defence Applications*. IEEE, 2009.

[17] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *IEEE Symposium on Security and Privacy (SP)*. IEEE, 2010, pp. 305–316.

[18] M. Andreoni Lopez, R. S. Silva, I. Alvarenga, G. Rebello, I. J. Sanz, A. Lobato, D. Mattos, O. C. M. B. Duarte, and G. Pujolle, "Collecting and characterizing a real broadband access network traffic dataset," in *1st Cyber Security in Networking Conference - CSNet*, Rio de Janeiro, Brazil, Oct. 2017.

[19] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Computers & Security*, vol. 45, pp. 100–123, 2014.