Enabling Anomaly-based Intrusion Detection Through Model Generalization

Eduardo Viegas, Altair Santin, Vilmar Abreu
Graduate Program in Computer Science
Pontifical Catholic University of Parana
Curitiba, Parana - Brazil
{eduardo.viegas, santin, vilmar.abreu}@ppgia.pucpr.br

Luiz S. Oliveira
Informatics Department
Federal University of Parana
Curitiba, Parana - Brazil
luiz.oliveira@ufpr.br

***Abstract*—Anomaly-based intrusion detection by the means of machine learning techniques is extensively studied in the literature mainly due to its promise to detect new attacks. However, despite the promising reported results, it is hardly deployed to real world environments. The main challenge in its adoption is the discrepancy between the accuracy rates obtained during the classifier development process and the rates obtained during its use in production environments. Such a discrepancy is mainly caused by non-representative training databases and non-generalizable (scenario-specific) classifier's model. This paper presents a method to create intrusion databases, which aims at mimicking the production environments characteristics by using well-known tools. Moreover, we present and evaluate a new validation technique, which aims at ensuring the generalization capacity of the obtained models, reached using cross-validating with different intrusion databases. The evaluation tests showed the feasibility of the proposed method. The feature selection technique ensured the model generalization capacity, improving its accuracy rate by 13%, while testing in different intrusion databases. Finally, the proposed anomaly-based approach was compared with Snort, reaching an accuracy rate of 99% against 27% of Snort for detecting DoS attacks.**

Keywords—Anomaly-based Intrusion Detection; Model Generalization; Machine Learning; Genetic Algorithm

I. Introduction

Intrusion Detection System (IDS) allows the detection of attacks, malicious or inadequate usage of a computational system or a network of computers [7]. Anomaly-based intrusion detection can help at detecting a growing number of new vulnerabilities [11] on IDSs is mostly framed as a pattern recognition problem, by the means of machine learning techniques.

Machine learning for IDS relies on an inference engine (classifier) and a model to classify new attacks. The process consists of inferring a behavior from a dataset, obtaining then an attack model. The attack model is used for intrusion detection on production (real-world) environments. The behavior is learned from an input dataset – a set of events (e.g. network packets, represented by a preset of features). In this case, the features are derived from the network packets fields (attributes).

Events from different classes (e.g. normal or attack) which presents similar behaviors, previous modeled, can be wrongly classified by an IDS. Thus, the classifier accuracy rate should be tested using a testing dataset, in order to reduce such possible errors (false positive or false negative).

When the IDS is used to detect network attacks, the machine learning expert believe that the classifier accuracy obtained during the model testing process will be observed during its usage in production environments. Thus, the dataset used during the IDS test process must precisely represent the network packets observed in production environments. However, according to Mahbod Tavallaee [15] more than 50% of works in literature used DARPA1998 [10] or similar datasets. The DARPA dataset was created in 1998 and updated in 2000. However, despite its extensive usage, the results obtained using DARPA1998 can be considered unreliable, because the network traffic constantly changes due to new network services offering and new attacks reported every day.

Different IDS evaluation datasets were proposed in the literature [4] since DARPA1998 [10]. However, the majority of the approaches have failed, either by its non-reproducibility, non-representative events or by not being publicly available [17]. Thus, hindering their usage to benchmark intrusion detection approaches.

Sommer and Paxson [19] noted the lack of applicability of the results reported in the literature. The authors mentioned the lack of coherent validation methods and the lack of usage of the anomaly-based approach in production environments. A reason pointed by them consists of unreliable classifier accuracy. Additionally, when an IDS is used on production, generated alerts are not necessarily attacks, in this case, called false-positives. A reason pointed to the detection methods failing is the lack of model generalization ability. In other words, the model is unable to detect attacks independently of the environment and conditions where it was obtained, thus being scenario-specific.

In the light of this, this work proposes a new method to create datasets for network-based intrusion detection evaluation, without the main problems reported in the literature. Furthermore, we propose and evaluate a new method that evaluates the generalization capacity of the used classifiers. Finally, the anomaly-based approach using our proposed method is compared with the commonly used tools to detect such attacks. In summary, the main contributions of this paper are (i) a method to create datasets for IDS evaluation, (ii) a method to obtain the generalization capacity of the attack model and (iii) a comparison of the detection approach used on production environments against our anomaly-based approach.

The paper is organized as follows. Section II presents related works. Section III describes the proposal. Section IV shows scenario and evaluation, and Section V draws conclusions.

II. Related Work

The main problem during the IDS development is the lack of public and updated intrusion datasets to testing the proposals. Currently, the most used IDS evaluation database is the DARPA1998 [10]. Which was obtained in a controlled environment, reproducing a real air-force network traffic behavior. The DARPA1998 was used to create the well-known dataset KDD99 [16], whose main problem is its creation date (1998). The reason was new services are provided and new attacks are discovered daily, therefore DARPA1998 traffic became outdated. Since then, several works have proposed the creation of new intrusion datasets.

Shiravi [1] proposed the use of profiles, which describes the client and attacker behavior in an environment. Each profile is statistically modeled by the analysis of the user behavior during a certain period of time. However, the statistical modeling of the user behavior is strictly for a given period of time and application. In addition, the user profile frequently changes on production environment [5]. Thus, approaches that aim at modeling the user behavior presents difficult to deal with updates and becomes scenario dependent. The usage of real network traces is also proposed in the literature [2, 12, 18]. However, those approaches prevent the dataset sharing among the researchers, requiring a sanitization process in order to remove sensitive data. Thus, the approach we propose in our work aim at providing a publicly available intrusion database by the means of well-known tools in a controlled environment, providing the expected properties from an intrusion database/dataset.

On the other hands, the attack model evaluation technique is hardly considered in the literature. The authors mostly assume a static environment by using the traditional machine learning testing schemes [3, 19], assuming that the training environment will be the same as the production environment. An extensive criticism is made by Sommer and Paxson [19], which relates the lack of anomaly-based usage in production environments. In our point of view, such situation happens due to the differences between the rates obtained during the IDS development (accuracy estimation) and its use in production. Thus, providing a wrong assumption that anomaly-based is less effective, in processing and accuracy terms than the signature-based approach. In fact, the signature-based approach does not have an accuracy estimation, because this kind of evaluation is not applied during the IDS development.

Siqi Ma et al. [21] used the GA (Genetic Algorithm) in order to optimize the classifier inputs in a HIDS (Host-based Intrusion Detection System) environment. On the other side, Gary Stein et al. [6] evaluate the GA (Genetic algorithm) impact while using it as a feature selection technique for the DT (Decision Tree) classifier. The authors used the KDD99 [16] dataset during the tests and the GA to reduce the classifier error rate. V. Bolón-Canedo et al. [22] evaluate the impact of feature selection for the NB (Naïve Bayes) and DT. The authors obtained accuracy gains and an average of 80% reduction in the number of used features, when GA is considered. It was not found any work that evaluate the feature selection impact on the model generalization capacity, neither how the attack model performs in a different environment than it was obtained.

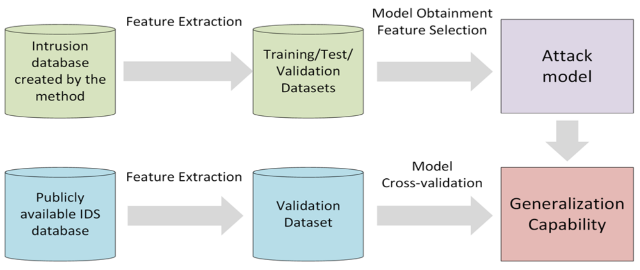
Intrusion database created by the method
Feature Extraction
Training/Test/ Validation Datasets
Model Obtainment Feature Selection
Attack model
Publicly available IDS database
Feature Extraction
Validation Dataset
Model Cross-validation
Generalization Capability

Figure 1 – Proposal overview

III. Proposal

Our proposal is to establish metrics that define the generalization capacity of the attack model. Thus, providing accuracy rates regarding the way the system would perform, if used in another environment. This is reached by performing cross validation over distinct intrusion datasets during the evaluation process, considering the same attack type. Thereby, it becomes possible to establish the model capacity to operate independently of the environment it was created.

The proposal is divided into three steps (Figure 1). Section III.A describes the creation method and intrusion database evaluation, which was used in our work in order to obtain a publicly available intrusion dataset, without the main problems reported in the literature [1]. Section III.B shows the extracted features set in order to obtain the dataset, allowing the cross validation with a distinct database. Finally, section III.C describes the feature selection method adopted in our work, in order to obtain the attack models.

A. Database Creation Method

To create intrusion evaluation databases, in our case, two classes are considered: normal (legitimate content) and intrusion (attack). The normal traffic is generated using the client-server model. For the server-side traffic generation, the *honeypot* technique is used. The *honeypot* is a technique used to obtain information about possible attackers as appearing as a vulnerable host in the network. In this way, all traffic generated by the server (*honeypot*) is real (containing traffic that can be observed in a production environment) and valid (well-formed packets, part of request and response messages). To generate the client-side traffic, workload tools are used. Thus, the generated traffic is real and valid for both client and server. The attacks are generated using well-known system auditing tools.

The method used for the normal traffic generation must ensure that the client-server interaction correctly occurs. Thus, ensuring that the client behavior evidenced on the IDS evaluation database is similar to the behavior evidenced on production environments. To this end, normal network traffic must be generated according to two perspectives, the client and the server. The client is responsible to generate requests to the available services on the server. The server is responsible to properly reply the client requests. It is expected that the provided services, as well as its contents requested by the clients, present a considerable variability in order to ensure network traffic diversity. The goal is to avoid repeated traffic, which is not desirable during an IDS evaluation and neither represents the production environment characteristics.

To mimic the client behavior, a set of services are provided, while each service provides a set of possible contents to be requested. Each client performs a real and valid request by the means of a workload tool to a predetermined service and content. The workload tool in our proposal is used only to generate a real, valid and easy to update traffic on the client-side. The tool becomes responsible to perform the valid communication between the client-server, being specific to the requested service. After the end of the client-server interaction, the client waits a pseudo-random time and performs a new content request.

The usage of a real server difficult the database update. In this way, our proposal uses a technique that mimics the server behavior, allowing the easy update and the generation of a real and valid traffic. A set of predetermined services are provided by the *honeypot*. In this way, every request regardless of the requested service is properly interpreted and a valid response is provided. Thus, the proposed method generates a real, valid and easy to update normal traffic.

On the network attack traffic creation side, the lack of a tool implementation standardization is the main issue. In general, in the literature, the authors implement a known attack according to its discretion. Which difficult the attacks reproducibility, as it is not possible to ensure that the attack implementation follows the attack *modus operandi*. Therefore, it is not possible to establish a common baseline for benchmark and generalization purposes. After a new attack becomes known and reported, initiatives such as the Common Vulnerabilities Exposure (CVE, cve.mitre.org) details the *modus operandi* of a vulnerabilities/attack and affected services. Implementations that are CVE compatible, ensure that attack behavior is, in fact, acting as reported. Thus, tools that uses a *de facto standard* (e.g. CVE compatible) are auditable and allows reproducibility.

In this proposal, we adopt well-known and *de facto standard* tools for the attacks generation. Thus, we are able to ensure that all the available attacks on the database are correctly implemented and will generate the proper kind of attack traffic, allowing the reproducibility and correct detected in production environment.

B. Feature Extraction

For each network packet read from NIC (Network Interface Card), a set of predetermined features are extracted and forwarded to a classifier engine for classification. The set of features used in this work was adapted from [8, 9]; a total of 50 features are extracted for each network packet.

The defined set of extracted features considers a NIDS scenario (further explained in section IV.A).

C. Feature Selection

The set of extracted features are generic enough to cover a broad kind of a NIDS (Network-based Intrusion Detection Systems). However, for a specific purpose, the IDS demands a features selection process, which allows selecting only the features that best characterize the target attacks. In the literature, the feature selection is mostly used to improve the attack model accuracy and to improve the IDS performance.

In this work, the feature selection is twofold. First, it aims at verifying the accuracy impact on the IDS. It is expected that the attack model obtained through the feature selection method will present a better accuracy when compared to the use of all features – without the feature selection. Second, the attack model generalization capacity will be measured. It is reported in the literature that the features reduction allows the improvement on the IDS accuracy. However, none of the literature proposals have studied the generalization impact when using a feature selection method.

This work considers the Genetic Algorithm (GA) for feature selection.

TABLE I. SERVICES DESCRIPTION

Service	Description
HTTP	Each client requests a random web page, from a repository mirrored from the 500 most visited worldwide websites (moz.com/top500), and stored in the honeypot server.
SMTP	Each SMTP client sends an e-mail between 50 and 400 bytes of subject and 100 to 4,000 bytes of content.
SSH	Each SSH client login on the honeypot server and executes a random predetermined command from a list of 30 possibilities.
SNMP	Each SNMP client walks through a predetermined MIB on a list of possible MIBs.
DNS	Every name resolution is performed on the honeypot server.

IV. Scenario and Evaluation

This section describes the scenario developed to evaluate the proposed method and also introduces the intrusion evaluation database that has been created (section III.A). The attack model generalization capacity was measured using a well-known intrusion database. Finally, a comparison against a well-known and broadly used signature-based tool was made, considering a production environment for intrusion detection.

A. Scenario

To generate the normal traffic, the services provided in the scenario were HTTP (*Hypertext Transfer Protocol*), SSH (*Secure Shell*), SMTP (*Simple Mail Transfer Protocol*) and SNMP (*Simple Network Management Protocol*). Every name resolution (DNS, *Domain Name System*) was performed by the *honeypot* server. Each client requests a service through a *workload* tool specific to the requested service. The time between each request is pseudo-random, ranging from zero to four seconds, in order to mimic the client unpredictable behavior while requesting a service. In total, 10 distinct clients were used. Table I describes the client behavior in each service request, in order to guarantee the traffic variability. To generate the attacker traffic, only DoS (*Denial of Service*) attacks were considered. The goal was to become possible the comparison between our proposal and the well-known intrusion evaluation databases, e.g. DARPA1998. It is worth noting that DoS attacks are common nowadays, including its usage in botnets.

TABLE II. NETWORK TRAFFIC DISTRIBUTION ON THE CREATED DATABASE

Type	Number of Packets	Database ratio	Attack ratio
Application-level attacks	39,107	0.40%	28.10%
Network-level attacks	100,072	1.02%	71.90%
Normal	9,640,289	98.58%	-
Total	**9,779,468**	-	-

B. Database Creation

The attacker traffic was generated according to two perspectives, network-level and application-level. In the application-level, the DoS attacks were generated using the LOIC tool. The LOIC generates HTTP flood attacks at a specific URL. The network-level attack was generated using a *synflood* plugin available at the Metasploit tool. Every attack was generated using a well-known and *de facto standardized* tool, allowing the correct implementation and code availability publicly, therefore allowing an anytime database reproducibility.

The client requests, as well as the generated attacks, were performed to the *honeypot* server. The honey tool was used to deploy the *honeypot* server.

The scenario was executed for 30 minutes. The attacks began at the tenth minute and lasted 15 minutes. The generated traffic was captured and stored on the *honeypot* machine, using the *tcpdump* tool. A total of 2.1 gigabytes of traffic was generated. Table II presents the traffic distribution on the created database. The implemented scenario for intrusion databases creation allowed the solution of some of the main problems reported in the literature.

The generated traffic is real and valid, as the *honeypot* generate valid replies to each of the received requests. The events classes were automatically established (labeled in feature vector) in order to avoid manual labeling and providing an error pruning approach, due to the number of packets to be evaluated. The automatic labeling was defined according to the source IP address for each network packet, which was possible, as the attacker machine generates only attack traffic. It is important to note that the IP address was not used as a feature value. Thus, the model is not biased by this knowledge. Additionally, the manual class labeling or clustering techniques [14] was avoided, reducing labeling error.

In order to allow the reproducibility of the deployed scenario, every client and attacker behavior was logged. Finally, privacy problems did not occur because the database was obtained on a controlled environment and the generated traffic does not present any sensitive data.

To elaborate the cross-validation, a well-known and public intrusion database is needed, making it possible to establish the generalization capacity from the used attack models. The public database DARPA1998 was used during the testing process, which is the network traces used to create the well-known intrusion dataset KDD99 [16].

TABLE III. NETWORK TRAFFIC DISTRIBUTION ON THE DARPA1998 DATABASE

Type	Number of Packets	Database ratio	Attack ratio
Ping of Death	9,600	0.03%	0.57%
Synflood - Neptune	1,301,516	4.45%	76.99%
Smurf	379,477	1.29%	22.44%
Normal	27,600,297	94.23%	-
Total	**29,290,890**	-	-

TABLE IV. ACCURACY RATE OBTAINED BY THE USED CLASSIFIERS

Model	No. of features	Accuracy	False-Positive	False-Negative
Naive Bayes	14 (using GA)	99.99%	zero	zero
Naive Bayes	All features	96.63%	3.47%	0.02%
Decision Tree	12 (using GA)	99.99%	zero	zero
Decision Tree	All features	99.98%	0.01%	zero

To perform the cross-validation, 35 network files provided by DARPA1998 were used. Each file has a class description for each connection. The feature extraction, developed in our work, was modified in order to establish the packets class according to each file description. Table III shows the network traffic distribution on DARPA1998. Only DoS attacks were used, as the goal is to establish the attack model generalization capacity for the same type of attacks.

C. Attack Model obtainment

A set of 50 features [8, 9] were adopted and a feature extractor algorithm was developed to obtain each feature from the network packets. The network packets are captured using the *libpcap library* and the feature extractor was implemented using the *C++ language*. Two classifiers were used, the NB and the DT [13]. The NB due to its simplicity during the model obtainment and low processing cost during the classification. The DT, using the C4.5 algorithm, was used due to its low processing demanded during the packets classification. For the feature selection process, the *wrapper-based* (classifier dependent) GA was used, with 100 generations and 100 populations in each generation, the same parameters used in [6]. During the selection, the fitness function objective was the lowest possible error rate with the lowest possible number of used features. To obtain the same proportion of the classes a stratification procedure was used. Thus, it became possible to use the error rate as the objective during the feature selection. Figure 2 and 3 shows the GA evolution for the created intrusion dataset, considering the error rate and the number of used features for the DT and the NB classifiers respectively.

For the DT algorithm, the lowest error occurred in generation 3, with 0.01% error and the lowest number of used features was reached in generation 34 (Figure 2). The NB Algorithm reached lowest error rate in generation 55, considering 0.01% of network packets wrongly classified. However, the lowest number of features was reached in generation 71 (Figure 3).

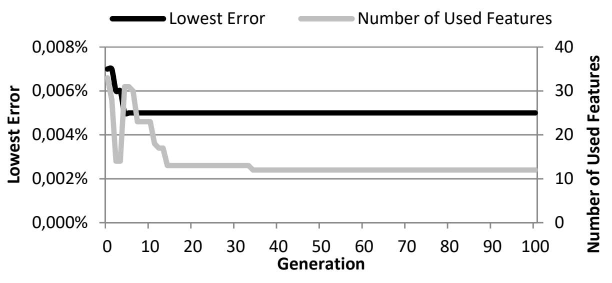

Figure 2 – Genetic Algorithm evolution for the DT classifier

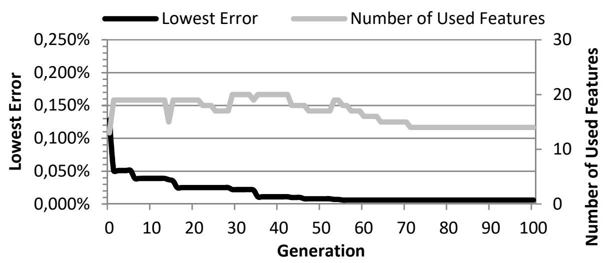

Figure 3 – Genetic Algorithm evolution for the NB classifier.

TABLE V. ACCURACY RATE OBTAINED DURING THE CROSS-VALIDATION (GENERALIZATION CAPACITY)

Model	No. of Features	Accuracy	False-Positive	False-Negative
Naive Bayes	14 (using GA)	94.95%	5.02%	5.97%
Naive Bayes	All features	72.56%	28.08%	0.89%
Decision Tree	12 (using GA)	96.82%	3.22%	0.89%
Decision Tree	All features	92.62%	5.82%	71.30%

Table IV shows the accuracy improvement obtained through the usage of the feature selection technique, using the evaluation dataset (different from that one was used to obtain the model and test it) produced in the scenario. The feature selection technique provided an accuracy improvement of 3.36% for the NB and 0.01% for the DT for the created dataset, besides the reduction of false-positive and false-negative rates. The reduction in the number of used features reduces also the processing demanded for its extraction.

D. Generalization Capacity Evaluation

Table V shows the results using the proposed attack model on the dataset DARPA1998. The feature selection technique obtained expressive results during the cross-validation. The feature selection for the DT classifier obtained an accuracy improvement of 4.20% when compared to the usage of all features. The false-negative rate reduced from 71.30% to 0.89% while the false-positive rate reduced from 5.82% to 3.22%. The NB showed the impact of the feature selection process. The NB attack model using all features obtained only 72.56% accuracy, while the model obtained using the feature selection approach reached 94.95% accuracy. The false-positive rate dropped from 28.08% to 5.02%, while the false-negative rate increase from 0.89% to 5.97%. Despite the increase in the false-negative rate, the NB accuracy improvement was 25.39%. It is possible to note a reduction of 9.91% on accuracy while using the attack model on another dataset. The feature selection method using the GA allowed improving the model generalization for the tested classifiers. The model generalization capacity was improved in 4.2% for the DT and 22.39% for the NB while using a feature selection method.

As shown, the attack model accuracy, regardless of the used classifier, drops when used in another environment. Thereby, when anomaly-based IDS is used in production environments, such property must be considered, the classification engine must be able to cope with the changes in the environment while also providing a reasonably high detection rate. Our evaluation tests have shown that the resource selection process helps maintain the expected accuracy of the attack model while the system is used in another environment.

E. Commercial Tool Comparison

In order to compare our proposal with a commercial product, we chose Snort (www.snort.org). During the tests, two signature parameters were varied (i) The *Number of Occurrences* and (ii) the *Interval*. Any connection attempt that exceeds the *Number of Occurrences* threshold during a defined *Interval* is identified as an intrusion attempt. A total of 10,000 tests were performed using the created intrusion database. Both parameters were tested with values ranging between 1 and 1,000. The Snort accuracy rate was established according to the number of connection attempts performed by the attacker machines correctly alerted by the tool. The best values found for the parameters variations had 3 connections (Number of Occurrences) in a 2 seconds' time window *Interval*, yielding 27.32% of accuracy. Comparing the results, it becomes possible to observe, in this case, that our proposal (anomaly-based intrusion detection) presents (about 3x) better detection rates than the signature-based approach while detecting DoS attacks.

Finally, we have compared the processing time demanded from each detection approach, signature-based using Snort and anomaly-based using our proposed approach. The measurements were performed on Ubuntu 14.04 with an Intel Core I7 and 16GB of RAM.

For the Snort tool, two signature sets were used. The first signature set uses only the signature with the best parameters (with Number of Occurrences = 3 and Interval = 2). This signature set is named as Snort DoS and aim at detecting only DoS attacks. The second signature-based set is named as Snort Default and uses the standard signature set (for the experiments Snort rules snapshot 2.9.62) from Snort website. In this way, we are able to establish the processing time demanded from Snort, when system administrator wants to detect only DoS attacks and when he uses the default signature set. For the anomaly-based approach, we have used the classifiers (Table IV) using all 50 features.

Table VI presents the average processing time demanded by each module, the Data Acquisition, PreProcessing for Snort and Feature Extraction for our approach and the Detection module. The demanded processing time while using the NB classifier used only 18.39% and 3.45% of the processing time demanded

TABLE VI. PROCESSING TIME COMPARISON

Module	Processing Time (Seconds)			
	Snort DoS	Snort Default	Our Approach NB	Our Approach DT
Data Acquisition	125.0	138.0	19.3	19.4
Preprocessing/ Feature Extraction	113.6	979.8	31.3	31.3
Detection	46.4	400.2	7.9	1.7
Total	***285.0***	***1518.0***	***58.5***	***52.4***

by the Snort DoS and Snort Default respectively, while the DT demanded only 20.53% and 3.85% respectively.

V. CONCLUSION

This work presented an approach to test the generalization capacity of an intrusion detection engine that uses anomaly-based classifiers. The proposed intrusion detection evaluation dataset method allowed to solve the main problems reported in the literature. During the generalization tests, the feature selection for DT cross-validation reduced the false-negative rate from 71.30% to 0.89% while the false-positive rate reduced from 5.82% to 3.22%. For NB, the attack model accuracy increased from 72.56% to 94.95% using the feature selection. The false-positive rate dropped from 28.08% to 5.02%, but the false-negative rate increased from 0.89% to 5.97%. Despite the increase in the false-negative rate, the NB accuracy improvement was 25.39%. The proposed method, using GA, shown an improvement, on average, of 1.68% for the overall IDS accuracy and 13.29% for the model generalization (the obtained accuracy while using the attack model on another environment). Finally, it was showed that the anomaly-based approach for DoS attacks detection presents a better accuracy rate when compared to the well-known signature-based IDS to detect similar attacks.

REFERENCES

[1] A. Shiravi, H. Shiravi, M. Tavallaee, and A. a. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur.*, vol. 31, no. 3, pp. 357–374, 2012.

[2] CAIDA. The cooperative association for internet data analysis [online] available: http://www.caida.org/. Accessed Apr./2018.

[3] C. Gates and C. Taylor, "Challenging the Anomaly Detection Paradigm: A Provocative Discussion," *Proc. 2006 Work. New Secur. Paradig.*, pp. 21–29, 2007.

[4] C. F. Tsai, Y. F. Hsu, C. Y. Lin, and W. Y. Lin, "Intrusion detection by machine learning: A review," *Expert Syst. Appl.*, vol. 36, no. 10, pp. 11994–12000, 2009.

[5] H. Ringberg, M. Roughan, and J. Rexford, "The need for simulation in evaluating anomaly detectors," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 1, p. 55, 2008.

[6] G. Stein, B. Chen, A. S. Wu, and K. a. Hua, "Decision tree classifier for network intrusion detection with GA-based feature selection," *Proc. 43rd Annu. southeast Reg. Conf. - ACM-SE 43*, vol. 2, p. 136, 2005.

[7] I. Corona, G. Giacinto, and F. Roli, "Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues," *Inf. Sci. (Ny).*, vol. 239, pp. 201–225, Aug. 2013.

[8] E. K. Viegas, A. O. Santin, and L. S. Oliveira, "Toward a reliable anomaly-based intrusion detection in real-world environments," *Comput. Networks*, vol. 127, pp. 200-216, 2017.

[9] E. Viegas, A. Santin, A. França, R. Jasinski, V. Pedroni, and L. Oliveira, "Towards an Energy-Efficient Anomaly-Based Intrusion Detection Engine for Embedded Systems," *IEEE Trans. On Computers*, vol. 66, pp. 163–177, 2017.

[10] K. Kendall "A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems", 1999

[11] Labs. Kaspersky. 2014. Security Bulletin 2014. 2014.

[12] Lawrence Berkeley Nationatiol Laboratory, The internet traffic archive [online] available: http://ita.ee.lbl.gov/index.html. Accessed April./2018.

[13] E. Viegas, A. Santin, V. Abreu, and L. S. Oliveira, "Stream learning and anomaly-based intrusion detection in the adversarial settings," in *Proceedings - IEEE Symposium on Computers and Communications*, 2017.

[14] L. Portnoy, E. Eskin, and S. Stolfo, "Intrusion detection with unlabeled data using clustering," *Proc. ACM CSS Work. Data Min. Appl. to Secur.* Philadelphia PA, pp. 1–25, 2001.

[15] M. Tavallaee, N. Stakhanova, and A. A. Ghorbani, "Toward credible evaluation of anomaly-based intrusion-detection methods," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 40, no. 5, pp. 516–524, 2010.

[16] M. V Mahoney and P. K. Chan, "An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection," *Proc. Sixth Int. Symp. Recent Adv. Intrusion Detect.*, vol. 2820, no. Ll, pp. 220–237, 2003.

[17] Meli, P., Hu, V., Lipmann, R., Haines, J., Zissman, M.: An Overview of Issues in Testing Intrusion Detection Systems. Technical Report NIST IR 7007, NIST, 2006.

[18] RTI International. PREDICT: Protected repository for the defense of infrastructure against cyber threats [online] available: http://www.predict.org. Accessed Apr./2018.

[19] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," *IEEE Symp. Secur. Priv.*, vol. 0, no. May, pp. 305–316, 2010.

[20] Symantec Corporation. 2013. Internet Security Threat Report. 18, Apr.2013.

[21] S. Ma, S. Wang, D. Lo, R. H. Deng, and C. Sun, "Active Semi-supervised Approach for Checking App Behavior against Its Description," 2015 *IEEE 39th Annu. Comput. Softw. Appl. Conf.*, pp. 179–184, 2015.

[22] V. Bolón-Canedo, N. Sánchez-Maroño, and a. Alonso-Betanzos, "Feature selection and classification in multiple class datasets: An application to KDD Cup 99 dataset," *Expert Syst. Appl.,* vol. 38, no. 5, pp. 5947–5957, 2011