Received: 24 April 2018 | Revised: 12 June 2018 | Accepted: 4 July 2018 | Published on: 12 August 2018

DOI: 10.1002/spy2.36

ORIGINAL ARTICLE

WILEY

Adaptive anomaly-based intrusion detection system using genetic algorithm and profiling

Paulo Angelo Alves Resende | **André Costa Drummond**

Department of Computer Science, University of Brasilia, Distrito Federal, Brazil

Correspondence
Paulo Angelo Alves Resende, Department of Computer Science, University of Brasilia, Distrito Federal, Brazil.
Email: pa@pauloangelo.com

Intrusion detection systems have been playing an important role in defeating treats in the Cyberspace. In this context, researchers have been proposing anomaly-based methods for intrusion detection, on which the "normal" behavior is defined and the deviations (anomalies) are pointed out as intrusions. In this case, profiling is a relevant procedure used to establish a baseline for the normal behavior. In this work, an adaptive approach based on genetic algorithm is used to select features for profiling and parameters for anomaly-based intrusion detection methods. Additionally, two anomaly-based methods are introduced to be coupled with the proposed approach. One is based on basic statistics and the other is based on a projected clustering procedure. In the presented experiments performed on the CICIDS2017 dataset, our methods achieved results as good as detection rate equals to 92.85% and false positive rate of 0.69%. The presented approach iteratively adapts to new attacks and to the environmental requirements, such as security staff's preferences and available computational resources.

KEYWORDS

adaptive intrusion detection systems, anomaly-based intrusion detection, apache spark, machine learning, profiling, projected clustering

1 | INTRODUCTION

Over the last years, the Cyberspace has been achieving high importance levels for the global society. However, this space has been also used for criminals, which benefit from vulnerabilities in humans and technologies to take advantages in a variety of forms. In this scenario, intrusion detection systems (IDS) are essential tools for defeating such threats in this complex universe.

The misuse and anomaly approaches are commonly used for IDS. The former supposes to known the malicious events' behaviors and use this information to detect attacks. While the latter is based on the normal behavior and points out as intrusions the deviations from a "baseline". Anomaly-based IDS has, among other advantages, the ability to detect unknown attacks.

In the anomaly-based IDS, a baseline can be defined by considering outliers in a set of events, for example, using data mining techniques and supposing that attacks are the minority portion. However, in some kind of attacks (eg, DDoS) this assumption is not true. Another used approach is creating profiles for objects in the network based on the historical observed normal behavior. In the scope of intrusion detection and based on Peng et al,[1] profiling may be defined as the process used to establish a baseline for the normal or malicious behavior of an object, which can be a user, a system, or a group of them. Basically, systems can be profiled by host-related or network-related data, while users may be profiled by biometric or psychometric data.

In the profiling scope, the following question naturally arises: "*Which features are relevant for intrusion detection and, thus, should be included in the profiling procedure?*" Also, profiling may be computationally costly, even considering the abundant available computational resources. For example, the IPFIX standard, currently, has 481 information elements registered in the Internet Assigned Numbers Authority.[2] Profiling these features for n objects would require $n \times 481$ profiles to be extracted,

transmitted, persisted, and processed. Moreover, some features may negatively compromise the detection method rather than contribute.

There are many advances in selecting features for intrusion detection methods, in[3] there is a brief review on this subject. However, it is reasonable to suppose that systems and attacks vary depending on time and on their environment. And, in this case, a static feature selection is not appropriate for an adaptive IDS.

In this scenario, we propose in this work an adaptive and anomaly-based IDS supported by a genetic algorithm, which is used to select features for profiling and optimizing the parameters used in the detection method.

Genetic algorithm is a procedure inspired by Darwin's evolutionary theory in Biology, which is commonly used for searching and optimization in the context of Evolutionary Computation.[4]

On each generation of the used genetic algorithm, a population of models is created based on crossovers and mutations of individuals from the previous generation. The detection models (and their respective selected features) are identified by the individuals in the population. These models are used to detect attacks during a time interval (eg, one day). The events alarmed by the population in this period are then submitted to a security staff for handling, which should point out the found False Positives (FP) (and True Positives [TP]). This output is used to rank the individuals to create the subsequent generation. The method benefits from a human-machine cooperation strategy to evolve according to the environmental particularities.

The individuals' chromosomes, essentially, have the selected features for profiling and parameters used in the anomaly-based detection model. Thus, the approach evolves a subset of features used for profiling and, at the same time, optimize the model's parameters. In the fitness function, used to rank individuals, there are two parameters to tune the acceptable levels of the false positive rates (FPR) and the number of selected features for profiling.

Our proposed fitness function depends only on the quantity of TP and FP. The TP and FP are subsets of the alarmed events, which are commonly reviewed by the security staff. Also, these values are expected to be much smaller when compared to the set of events evaluated by the IDS.

We note that, in a typical application of a genetic algorithm, the fitness function can be computed directly based on the information found in the individuals' chromosomes. However, here, the fitness function requires information provided by a security staff (or an "Oracle"). Due to that, the evolution of each generation takes some time to be performed. In our proposal, the population's evolutions never end. This provides a continuous adaptive approach.

The main contribution of this work is the adaptive approach used to select features for profiling and parameters for an anomaly-based intrusion detection method. However, we also propose two anomaly-based detection methods to evaluate our approach. On the first method, we used basic statistics over the profiled features to estimate distributions. The deviations of these distributions are pointed out as intrusions. The second method is based on a projected clustering approach. Basically, a K-means clustering procedure is used to find centroids and clusters' proportions, which are considered as the profiles. In the detection phase, the events are projected into a subspace, on which the clusters are identified. Intrusions are those events "far" from the normal centroids or in "small" clusters. The selected features are used to identify the subspace used in the projections.

We implemented our approach on a scalable architecture supported by Apache Spark and we used in the presented experiments the CICIDS2017 dataset proposed by Sharafaldin et al.[5] The approach provided good results in terms of adaptability, detection rates (DR), and FPR. The results varied depending on the attack types. We found performances as good as DR equals to 92.85% and FPR of 0.69%.

Albeit we evaluated the approach in the scope of network intrusion detection, it may be used in other contexts in the realm of anomaly-based IDS supported by profiling.

This work is structured as follows: in Section 2 we present some basic theory for genetic algorithms and projected clustering for those readers that are not familiar with these topics; in Section 3 we review works related to feature selection for anomaly-based IDS, genetic algorithms applied in the scope of IDS, and works targeting adaptability for IDS; in Section 4 we describe details about the proposed approach; Section 5 provides evaluation experiments using our proposals; finally, on Section 6, we point out considerations and future directions for our findings.

2 | BACKGROUND THEORY

Genetic algorithms date back at least as far as 1957 when Fraser[6] proposed modeling genetic systems in computers. Later, the proposed model was enhanced and proposed for other settings, such as optimization and searching in the context of machine learning.[4,7]

Presently, there are many variations of genetic algorithms used in the realm of machine learning problems. Below, we present the version used in this work. For further details about such models, we recommend.[8]

A genetic algorithm can be understood as a set of possible solutions for a problem, on which there exists a fitness function to measure the quality of each solution and an evolutionary approach to evolve this set by combining the best solutions and

creating new solution sets, iterating until a stop criterion. This approach is inspired by Darwin's evolutionary theory and inherits the terminology used in biology.

The set of solutions is referred by a "population," which is formed by "individuals." Each individual has a "chromosome", compounded by "genes" (positions) and "alleles" (values). The chromosome is a tuple of numbers that uniquely represent an individual. Usually, this is a tuple of binary numbers, but it can have real numbers. Evolved populations are called by "generations." The combination of the best solutions to create a new generation is done by "crossover" and "mutation" of individuals.

Crossover of two individuals is done by paring their chromosomes and, for each gene, randomly selecting an allele of one parent, for the binary case, or taking the mean of parents' alleles, for the real number genes. This method produces a new chromosome used to refer to the new individual, also named by the "offspring."

Mutation is a process that, based on a predefined probability, randomly changes genes of some individuals in the population.

There is a criterion to select individuals for matching to create a new generation. In our case, we choose the best individuals on two parts of the population randomly taken.

The fitness function provides a quality measure for individuals considering the problem to be solved. For example, if the problem is to select the best feature subset considering a total of r features, which are used to train a supervised method, the chromosome can be a r-tuple of binary values representing the selected features and the fitness function can be the accuracy obtained by using the feature subset represented in the chromosome.

To ensure that the new generations will always evolve, the best k individuals of each generation are kept in the respective subsequent generation. This approach is called by the elitism method.

Genetic algorithms are particularly useful for searching a large search space, on which an exhaustive search is not feasible. We note that the approach is not gradient oriented, which avoids fitting local best solutions, and are suitable for parallel processing.

Our approach also uses clustering, which is a well known unsupervised approach used to divide a dataset into alike subgroups (clusters). Roughly, each element in the dataset is represented as a point in a multivariate metric space (eg, $\mathbb{R}^d$) and a procedure is executed to divide the points into K clusters based on point distances, patterns or densities. There are many algorithms used for clustering. One of the most famous is the K-means method,[9] which requires a predefined number of clusters (K) and uses distances.

In ordinary situations, clustering is easy to implement and to understand. However, clustering can be complex in high-dimensional spaces. Mostly because the clusters may not be found in the fully dimensional space but only in proper subspaces! For such a case, there are plenty of algorithms used to find the best subspaces and their respective projected clusters. We recommend Kriegel et al,[10] which survey this class of methods.

Projected clustering algorithms, in general, are unsupervised and have suitable metrics to find subspaces and its clusters. In this work, we used two distinct anomaly-based alternatives coupled with a genetic algorithm for adaptive intrusion detection. One of these alternatives is based on projected clustering. Here, we use the information found in the chromosome to identify the subspace on which clusters are projected and evaluated.

3 | RELATED WORK

We have not found relevant works using genetic algorithms to select features for profiling to support anomaly-based IDS, which focus on using the human cooperation to evolve a detection model efficiently.

Below we present some works regarding feature selection for anomaly-based IDS, use of genetic algorithms in the scope of IDS, and adaptive intrusion detection approaches.

For profiling, we recommend Peng et al,[11] which provide a survey of approaches based on profiling applied to intrusion detection, authorship attribution, plagiarism detection, astroturfing detection, among others. A wide review of anomaly-based methods applied to intrusion detection can be found in Bhuyan et al.[12] Also, further references about genetic algorithms applied in intrusion detection can be found in Majeed and Kumar,[13] which present a brief review of such methods.

Stein et al[14] proposed the use of genetic algorithms to select features for a network intrusion detection method based on a decision tree classifier. Similarly, Khammassi and Krichen[3] propose the use of genetic algorithms for feature selection using a logistic regression classifier, and Shon et al[15] proposed the use of genetic algorithms to select features for a support vector machine (SVM) classifier. Kim et al[16] also based on an SVM classifier for intrusion detection, but they included the features (true or false) and the SVM parameters (real numbers) into the genetic algorithm chromosome to search for the optimal values for both, that is, the best SVM parameters and the best feature subset. Kuang et al[17] used a genetic algorithm to search for optimal parameters of a misuse detection approach based on Kernel principal component analysis (KPCA) and SVM.

On the realm of feature selection for anomaly-based network IDS, Kloft et al[18] propose an automatic feature selection by generalizing the support vector data description (SVDD). Basically, the features are combined in a linear mapping, which is

used in an optimization problem. The result is an anomaly-based detection model similar to a one-class classifier. Chen et al[19] propose the use of maximal information coefficient (MIC) (a filter approach) to select features and a feature-based multi-scale PCA (MSPCA) to detect anomalies in networks.

Balajinath and Raghavan,[20] Pillai et al,[21] Li,[22] Gong et al,[23] Khan[24] propose the use of genetic algorithms to create rules for signature-based IDS. Basically, the methods evolve a set of matching rules to obtain an efficient and understandable rule set.

Diaz-Gomez and Hougen[25,26] based on the GASSATA[27] approach to detect attacks in audit trail files, which contains the commands executed by the system's users. The method uses genetic algorithms as a searching tool to find a solution to a problem, which is then used to detect intrusions.

Xia et al[28] propose a network intrusion detection approach, in which, basically, a vector in the hyperspace represents the normal behavior. The flows' features are then projected (using the inner product) in this vector. The approach considers as intrusions the flows that have projection values less than a predefined threshold. The genetic algorithm is used to search for this vector. The authors propose the use of a filter approach to select features.

In Lee et al,[29,30] Lee and Stolfo,[31] Lee et al,[32] the authors propose an adaptive framework based on Data Mining to automatically create features and signatures (rules) for intrusion detection. They also propose a misuse and an anomaly-based method coupled with their approach.

Shafi and Abbass[33] propose an approach to actively create intrusion detection signatures based on labeled network traffic (malicious/benign). The method uses genetic algorithms to search for optimal rules and to learn in an evolutionary fashion.

Callegari et al[34] proposed an information-theoretic approach to detect anomalies in network traffic. In their approach, the network traffic is submitted to a hash function considering a time-window interval. An entropy function of each hash and a divergence measure of the hashes' histograms are used to detect anomalies in the traffic, considering some predefined thresholds.

Hamamoto et al[35] propose an anomaly-based network IDS using genetic algorithms, fuzzy logic, and exponentially weighted moving average (EWMA) models. The network flows are profiled by the digital signature of network segment using flow analysis (DSNSF), proposed in,[36] combined with a genetic algorithm. The EWMA is used to compute the deviations from the considered normal behavior. They presented experiments using an ad hoc dataset compound by DDoS attacks and normal traffic. The best found results were DR = 76.50% and FPR = 0.56%.

Aziz et al[37] provided a comparison of the performance of Machine Learning classifiers on the scope of anomaly-based network intrusion detection. The authors compared Naive Bayes, Decision Trees, Neural Networks, among others, using the NSL_KDD dataset. The best performance was achieved by a Naive Bayes classifier—DR = 73.50%.

Tran et al[38] propose an anomaly-based IDS supported by a Neural Network classifier. The best contribution of this work is the architecture based on field-programmable gate arrays (FPGA) combined with graphics processing units (GPU), which is created targeting the achievement of a high-speed detection. The authors argued that the used NetFPGA could handle up to 10 Gbps. However, the communication between the NetFPGA and the GPU limits the solution to only 200 Mbp/s. We highlight that our approach makes use of a scalable architecture, which can theoretically achieve a higher processing capacity.

The use of a genetic algorithm is the scope of IDS is also proposed by Aslahi-Shahri et al,[39] which use such algorithms to select features and parameters of an SVM classifier in a misuse-based detection method.

Most of the above-mentioned works propose the use of genetic algorithms to tune the method before its use in the real environment. This occurs, basically, because the proposed genetic algorithms require labels to compute the fitness function and such labels are not expected to be available continuously in a real setting.

The reviewed works devoted to adaptive intrusion detection, in general, focus on creating rules for signature-based IDS based on the labeled data.

Our approach requires only the TP and False Negatives to compute the fitness function. These values are created by analyzing the set of alarms generated by the intrusion detection method. It is reasonable to think that this set (alarms) is inspected by the network security staff. Also, such a set is much smaller than the total number of considered events.

Thus, our method benefits from the continuous human-machine cooperation to provide an IDS adapted to the environment, which includes the business needs, the type of threats, staff preferences, among other particularities.

4 | PROPOSED APPROACH

We present in this section the proposed method for adaptive intrusion detection based on genetic algorithms and profiling. We also present two anomaly-based approaches that are coupled with the genetic algorithm, which is used to select the optimal parameters and features for profiling. The first detection method is based on profiling variable distributions and the second creates profiles for centroids and proportions of clusters.

On the presented detection models, the input features may be those commonly used for behavior-based IDS for networks or hosts, for example, packet sizes and inter-arrival times, for network IDS, and system calls for host IDS. The features can also be

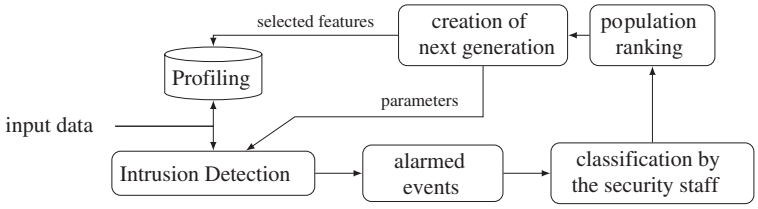

FIGURE 1 Diagram for the proposed adaptive approach

aggregated, for example, by time-windows. This characteristic makes our approach quite general. In the experiments presented in Section 5, we used network flows aggregated by time-windows, for the first proposed anomaly-detection method, and the network flows separately, for the second proposed method.

The genetic algorithm is used to optimize the selected features for profiling and the parameters used in the detection method.

For each generation, the features selected by the individuals are used to create profiles for each host of the protected network. To simplify our exposition, hereafter, we consider the IP address as the host identification. After a time interval of profiling, for example, 1 day, the detection method is tuned by each chromosome and, then, the method is used to detect intrusions in the subsequent time interval. The alarms generated by the entire population (excepting the "wild individuals") are submitted to the security staff for treatment and classification considering the classes "FP" and "TP". These values (FP and TP) are used to rank the population's individuals and create the next generation. This process is illustrated in Figure 1. We note that more than one individual may point out the same event as malicious. In this case, the event should be alarmed once. As "wild individuals," we consider those individuals that generate more than two times the mean of alarms of the population. This avoids that a mutation process produces an individual with high-FPR.

By this method, the life cycle of each generation takes three time intervals for the following procedures: (1) creation of profiles; (2) detection to produce inputs for the evaluation, used to rank and to create the next generation; and (3) detection until the profiles for the next generation are ready for use. Figure 2 shows the allocation time for the generations, which shows the overlapping between life cycles. We emphasize that, even with a new generation, the intrusion detection must keep using the past generation while the profiling for the new generation is created based on the (new) selected features.

In this work, we also propose two detection methods to be used with the above-mentioned genetic algorithm. The first method is based on fitting a distribution for each selected feature and each IP address of the protected network. For continuous features, we propose the use of Gaussian distributions, and for categorical features, we propose the use of normalized histograms. Thus, for each IP address, there exist r (number of selected features) Gaussian distributions (or histograms) represented by their means and standard deviations (or values and proportions). An event is alarmed as malicious if the P-value (or probability of the event) of at least the proportion x_1 of the selected features are less than a threshold y_1. The selected features and values x_1 and y_1 are optimized by the genetic algorithm and identified in each individual's chromosome.

The second method considers the vector space, V, compounded by the features selected by all individuals in the population of the genetic algorithm. For each IP address in the protected network, a K-means clustering algorithm is executed to create K clusters in V. The clusters are represented by their centroids and proportion of elements, considering the total number of events related to the respective IP address. This information provides the profile for the respective IP address. We note that the features should be normalized before clustering.

The features selected by a specific individual provide a subspace of V, suppose $S \subset V$. In the detection process, for an event in which the IP address i is involved, the centroids of its profile is projected into S and those centroids that are closer to each other (Euclidean distance less than a threshold c) are merged by taking the mean value as the new centroid and summing their proportions to create the proportion for the newly merged cluster. In this process, a new set of centroids and proportions is created for each IP address in the protected network and individual in the population. However, these values are produced just when they are needed, on demand, by a simple computation.

For each event, its respective vector is projected into S and its cluster is found out by choosing the closer centroid. An event is alarmed as malicious if one of the following cases occurs: (1) the chosen cluster is small, that is, its proportion is less than

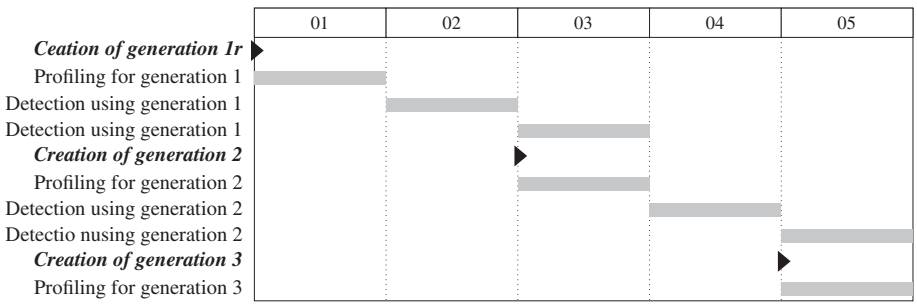

FIGURE 2 Time allocation table to illustrate the life cycle of generations in the adaptive method based on genetic algorithms. The columns represent the time intervals, for example, days

x_2; or (2) the distance between the projected vector and its respective projected centroid is larger than y_2. As in the first method, both x_2 and y_2 are identified in the individuals' chromosomes.

In (1), it is supposed that attacks are rare and provide small clusters. In (2), it is supposed that attacks have different behaviors when compared to normal events, so malicious events should be far from the normal clusters' centroids.

The chromosomes have one boolean gene for each feature and two real genes for the parameters x_1 and y_1 (or x_2 and y_2), for example,

$$chromosome = (0.2, 0.5, true, \dots, false).$$

The crossover of two individuals is performed by taking the mean value of the real genes and taking the value of a parent randomly selected for the boolean genes. We propose the use of the elitism technique, on which the best b individuals of each generation is carried to the subsequent generation. This method ensures that each generation is "better" or equivalent to the previous one. To select pairs for crossover, we choose the best individual on a part of the population randomly taken. This technique is called by "tournament selection." Mutations occur following a defined probability. The population has a fixed size P. The next generation is formed by the best individual and $P - 1$ new individuals.

The fitness function, used to rank the individuals in a population, is fundamental for the evolution of the detection method. Pillai et al[21] proposed the use of genetic algorithms in the realm of intrusion detection and used the following fitness function, for the individual j:

$$fitness(j) = \frac{\mathrm{TP}_j}{\sum_{i \in population} \mathrm{TP}_i} - \mathrm{FPR}_j \tag{1}$$

where TP_j is the true positives and FPR_j is the false positive rate ($\mathrm{FPR}_j = \mathrm{FP}_j/(\mathrm{FP}_j + \mathrm{TN}_j)$ and TN_j is the true positives) for the individual j.

Similarly, Xia et al[28] proposed

$$fitness(j) = \frac{(\mathrm{TP}_j + \mathrm{TN}_j)}{n}, \tag{2}$$

for n the total number of events.

Equations 1 and 2 depend on knowing the number of TN, which, in our case, is the number of normal events that are not alarmed as intrusions. In the intrusion detection problems, the normal events are expected to be much larger than the attack events. Also, this set (TN) is not usually treated by the security staff. Due to these questions, these fitness functions are not suitable for an adaptive approach.

Also, Equation 1 uniformly compares TPR to FPR. This makes results such as FPR = 30% and TPR = 100% equals to FPR = 0% and TPR = 70%. However, commonly, the latter result is preferred, because 30% of FPR is infeasible in real settings. One can show using a simple computation that Equation 2 has the same behavior.

Considering these points, we suggest the fitness function

$$fitness(j) = \frac{\mathrm{TP}_j + 1}{n} \exp\left[-\left(t_1 \frac{\mathrm{FP}_j}{n}\right)^2\right] - t_2 F, \tag{3}$$

where F is the number of selected features, t_1 is a constant that represents the intolerance to FP, that is, large values for t_1 penalize individuals with high FPR, and t_2 is a factor used to penalize individuals with large numbers of selected features. The exponential function "exponentially" penalizes large FPR values. This behavior avoids the above-mentioned questions pointed out for the other found fitness functions in the literature. The 1 summed to TP_j is used to avoid zero value in the first factor, which would make the FP_j free for any value.

Our fitness function has the following advantages:

1. depends only on the TP and FP, which are in the set that is usually treated by a security staff and are smaller values when compared to the total number of events;
2. leads to the best TPR with FPR near to 0;
3. the trade-off between TPR and FPR can be easily adjusted by changing the constant t_1, based on the user's preferences; and
4. the number of expected features selected for profiling can be adjusted by changing t_2, based on the available computational resources.

The information unit used to detect intrusions in both methods can vary according to the available features and the attacks supposed to be detected. For example, for DDoS attacks, it is preferred to aggregate the network flows in time windows because it is obviously expected that the number of flows in the time windows increases in such attacks.

The presented approach requires only the TP and FP to compute the fitness function. As argued above, these data are quite small when compared to the other alternatives and, also, the security staff commonly have to evaluate this universe to treat

TABLE 1 Description of files from the CICIDS2017 dataset

File	Description	Normal flows	Attack flows	% of attacks	Total
0	Normal traffic captured on Monday, July 3, 2017	529,918	0	0	529,918
1	Brute force attack on FTP and SSH servers captured on Tuesday, July 4, 2017	432,074	13,835	3.10	445,909
2	DoS/DDoS and heartbleed attacks captured on Wednesday, July 5, 2017	440,031	252,672	36.47	692,703
3	Web attacks (brute force, XSS, and SQL Injection) captured on the morning of Thursday, July 6, 2017	168,186	290,782	63.35	458,968
4	Infiltration attacks captured on the afternoon of Thursday, July 6, 2017	288,566	36	0.01	288,602
5	Botnet traffic captured on the morning of Friday, July 7, 2017	189,067	1,966	1.02	191,033
6	DDoS traffic captured on the afternoon of Friday, July 7, 2017	183,910	41,835	18.53	225,745
7	Port scan traffic captured on the afternoon of Friday, July 7, 2017	127,537	158,930	55.47	286,467

incidents. Nevertheless, the TP and FP numbers can be estimated with an acceptable confidence. In this case, supposing n alarmed events, the security staff can randomly select a sub-sample of size $n_0 \ll n$ and classify this subset. Based on some statistics, one can roughly consider the alarmed events as a sample of a Binomial Distribution considering the proportion of TP as the parameter p. Using a Normal approximation interval, a confidence level of 95% and considering $\widehat{\text{TP}}$ and $\widehat{\text{FP}}$ the subset classified by the security staff and $\varepsilon = \varepsilon(n)$ a negligible value, we have[40]:

$$\frac{\text{TP}}{\text{TP}+\text{FP}} \in \left[\hat{p} - 1.96\sqrt{\frac{\hat{p}(1-\hat{p})}{n_0}} - \varepsilon, \hat{p} + 1.96\sqrt{\frac{\hat{p}(1-\hat{p})}{n_0}} + \varepsilon \right]$$

for

$$\hat{p} = \frac{\widehat{\text{TP}}}{\widehat{\text{TP}}+\widehat{\text{FP}}}.$$

To exemplify, supposing $n_0 = 100$ and $\hat{p} = 0.5$, we have

$$\frac{\text{TP}}{\text{TP}+\text{FP}} \simeq 0.5 \pm 0.098$$

which is an acceptable approximation to be used in the fitness function.

We have proposed two detection methods to be coupled with the genetic algorithm. However, we note that our approach may be combined with other anomaly-based intrusion detection methods.

5 | EXPERIMENTS AND RESULTS

For the experiments, we implemented the proposed approach (genetic algorithm and the two anomaly-based intrusion detection methods) on the Apache Spark and used the CICIDS2017 dataset,[5] which is distributed into eight CSV files (and respective PCAP files). We describe in Table 1 the content of each file referring to them as "File k," for $k = 0.0.7$. The files have 84 columns, including "Label," "Flow ID," "Source IP," and "Destination IP." These fields are not useful for intrusion detection, thus, there are 80 fields, which were used in our experiments. We note that the CICIDS2017 dataset provides a good variability in the attack types and also in the proportion of attacks in each file, which is useful to compare different scenarios in the experiments.

We considered for profiling the IP addresses on the protected network, in the case, 192.168.10.0/24. The information unit used to detect intrusions in the first method is the set of flows related to each IP address during a time interval of 1 hour. In the second method, the information unit is each flow. A shorter aggregation time interval should provide a better sensibility for attacks, however, it may produce more FP. This occurs because, usually, the network access is not uniform in time. As an example, one host may have a period of time without relevant communication and, suddenly, it can produce a peak of transmission when a user downloads a file. This behavior would produce an alert in short aggregation time intervals using the first detection method proposed in this work. An interval of 1 hour smooths such peaks. But, in this case, stealth attacks may not create sufficient behavior changes to be detected. We point out that a promising research question is to analyze the performance of anomaly-based approaches focusing on the evaluation and comparison of time intervals used in such kind of aggregation.

TABLE 2 Parameter values used in the experiments

Parameter	Used value
t_1	10
t_2	0.001
Mutation probability	0.05
Population size	5
Number of generations on the first file	200
Number of generations on the subsequent files	30
Population subset proportion used for tournaments	50%
Number of best elements taken to the next gen. (elitism)	1
Interval for parameter x_1 ("P-value") of method 1	[0, 0.2]
Interval for parameter y_1 ("proportion") of method 1	[0, 1]
Interval for parameter x_2 ("cl. proportion") of method 2	[0, 0.2]
Interval for parameter y_2 ("distance") of method 2	[0, 0.01]
Cluster merge threshold c of method 2	0.0001
Number of clusters, K, of method 2	100

Some features in the dataset are sensitive to the flow's direction, for example, "Total Backward Packets" and "Total Forward Packets". In such cases, we interchange the respective fields whenever the profiled IP address appears in the destination flow field.

In only 5 days, the dataset provides a wide spectrum of attacks. An adaptive approach may require a larger time for adaptation. To overcome this, we kept the same files for many generations to provide a better approximation of a real setting. Even then, in many presented cases, the first generation in the particular file is good enough to be useful.

We present two experiments, cases 1 and 2. In case 1, the genetic algorithm is coupled to the first proposed method, which is based on fitting a distribution for each selected feature. The network flows are grouped by time-windows of 1 hour. On case 2, the second proposed method is used with flows separately.

In both cases, the profiles are created for each IP address of the protected network and the File 0 (normal traffic) is used to create profiles. Also, we used the parameters as described in Table 2.

We note that large numbers of individuals in the population provide a better genetic variability, which, in general, provides a faster adaptation (evolution). However, a large number of individuals increases the number of total selected features and the number of FP. Considering these points, we set five individuals in the population, based on empirical tests.

We considered one population for each case. We oriented the evolution of generations based on the files available in the dataset. However, the generations can be evolved in any time interval or based on the number of classifications made by the security staff.

In Figure 3, we present the DR and FPR for cases 1 and 2 considering the seven files in the dataset that contain attacks. We point out that the DR and FPR refer to alarms generated by the entire population combined. In the first file (Figure 3B) we used 200 generations to achieve an evolved population to be used in the subsequent files. In real situations, we suggest that the population should be evolved in a labeled dataset before used in the environment. In the subsequent files (Figure 3C to H), we considered 30 generations.

The first proposed anomaly-based method (case 1) performs better in noisy attacks, such as brute force and DDoS attacks (Figure 3B,C and G). This occurs because the information unit is aggregated flows by time intervals, on which the features are summed up. Noisy attacks produce an atypical amount of flows per interval, which provides large accumulated features that deviate from normal behavior.

The second method (case 2) performed better in stealth attacks, such as web and infiltration attacks (Figure 3D,E). Such an attack may have some features that make its behavior different from the normal attacks. The projected clustering approach, as proposed, finds the subspace created by these features which enables to distinguishing attacks from normal behavior. However, a noisy attack, like a DoS, produces many flows, but they may have similar behaviors when compared separately to normal accesses. Due to that, this method may not detect noisy attacks as the former. In other words, the detection difference is mainly due to the input data than the proper method.

Both methods start with good DR (Figure 3B). This occurs because the population is initialized with the values for the method's parameters (x_1, x_2, and y_2) in reasonable intervals (Table 2).

case 1 achieved DR = 95.28% and FPR = 2.73% in generation 200 on File 1. This is a notable mark for a method that evolves based on practically no information about the input features and on the limited information used in the fitness function (TP

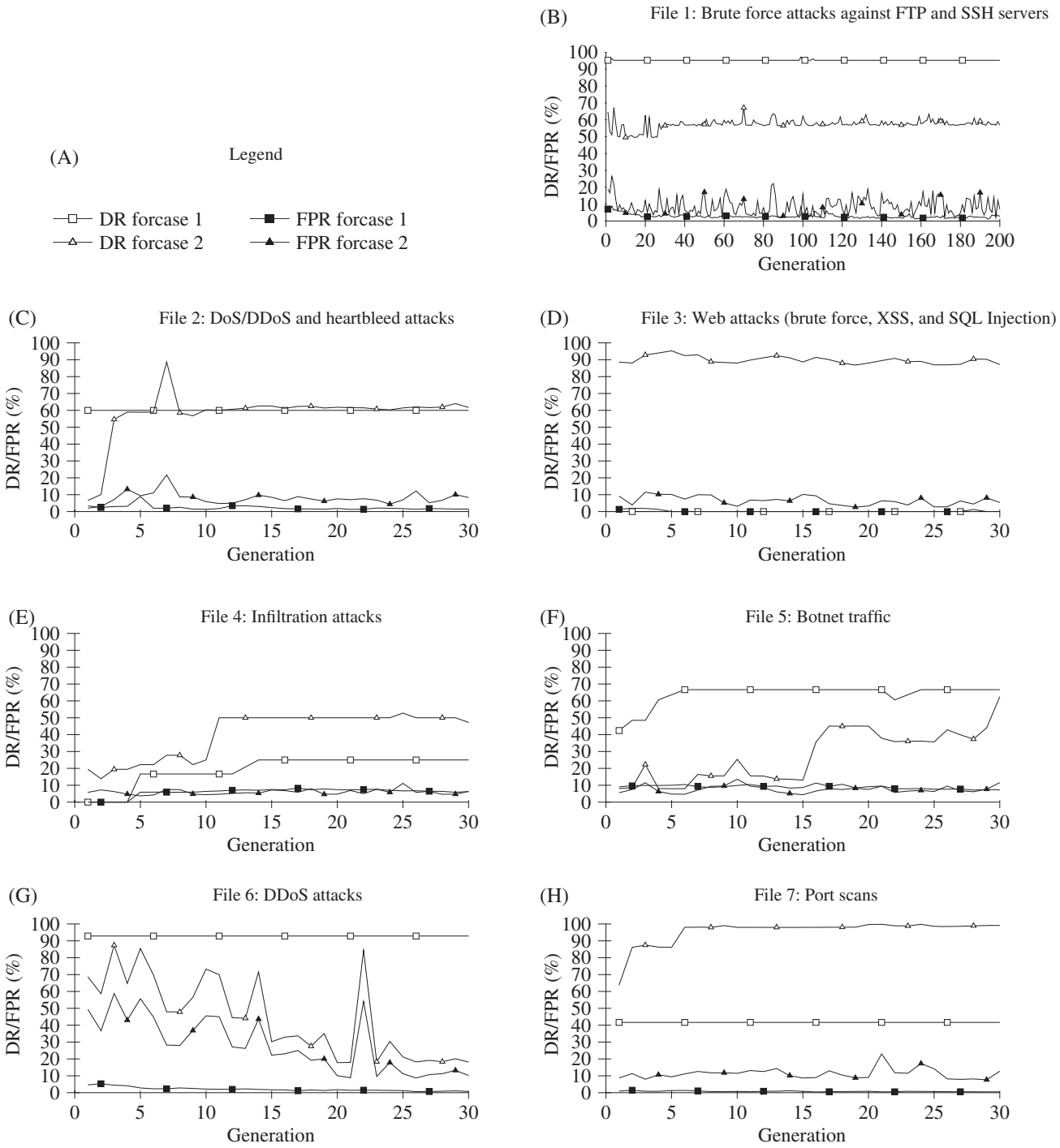

FIGURE 3 DR and FPR for cases 1 and 2 over the CSV files 1 to 7 of the CICIDS2017 dataset

and FP). Also, case 2 in File 1 could find a subspace in which it is possible to distinguish more than 56.83% of attacks (and FPR = 7.08%) based on flows separately.

We point out that the populations evolved in a kind of attack are useful in subsequent attacks, which is not trivially expected. For example, case 1 evolved in File 1 (Figure 3B) and begins the detection in File 2 (Figure 3C) with acceptable DR (60% and FPR = 3.38%). The same population resists an absence of evolution/detection (DR = 0% and FPR near 0%) through File 3 (Figure 3D), evolves in File 4 (Figure 3E) and File 5 (Figure 3F), and returns performing well (DR = 92.85% and FPR = 4.64%) in Files 6 (Figures 3G). In this example, we see that the adaptive approach does not regress in absence of detectable attacks and can evolve in different situations. We note that case 2 in File 6 (Figure 3G) is an interesting example of adaptability. The method was unable to detect intrusions with an acceptable FPR level and consistently reduced the DR and the FPR.

Botnets have particularities on establishing the command and control channels. Basically, there are automation and similarity patterns in such flows. These characteristics are heavily used by detection methods focused on this kind of threat. For further

TABLE 3 Best performances (DR and FPR) found in the last generation of each file

File	Method	DR (%)	FPR (%)
1	1	95.28	2.76
2	1	60.00	1.50
3	2	87.15	5.39
4	2	47.22	6.35
5	1	66.66	7.25
6	1	92.85	0.69
7	2	99.09	12.70

TABLE 4 Best performances (DR and FPR) found in the last generation of each file

Work	Dataset	DR (%)	FPR (%)
Callegari et al[34]	MAWI	90.00	5.00
Hamamoto et al[35]	Ad hoc	76.50	0.56
Aziz et al[37]	NSL_KDD	80.56	n.p.[a]
Tran et al[38]	Ad hoc	61.82	0.99
Our best performance	CICIDS2017	92.85	0.69

[a]Not provided.

details in this direction, we recommend.[41–44] On the other hand, detecting Botnets basing on flows separately is not trivial, because the flows generated by these threats usually are similar to normal accesses.

In all files, the first method (case 1) provided a stable behavior when compared to the other method. This occurs because the first method requires that at least the proportion of x_1 of the selected features deviates from their respective distributions. Thus, the inclusion of a feature which is not useful for detection does not influence the result, because the method does not require that all features deviate from the normal behavior. This is not the case in the second method, in which one feature can take an element out of a cluster. However, the second method is more sensitive to stealth behaviors.

Table 3 shows the best results found in the last generation of each file. We point out that in File 7, the 30th generation of case 2 provided FPR = 12.70%. However, this is a local oscillation peak. In the 29th generation, the FPR is equal to 7.65%, without relevant change in the DR. In Table 4 we present our best result compared with the best results found in some recent anomaly-based intrusion detection methods proposed in the literature and reviewed in the Section 3. We point out that our method provided the best performance in terms of DR and FPR. In addition to this performance, our method provides adaptability and makes use of a labeled information usually produced by the security staff.

Figure 4 shows histograms for the selected features by both cases, considering all files and generations. In case 1 (Figure 4A), the number of selected features was around 42 (of 80). In case 2 (Figure 4B), was around 37. Case 1 accommodated with a larger number of features than case 2. This occurred because case 1 may accept unused features in the model, due to the approach with the threshold x_1. In case 2, each feature can influence the clustering formation. The number of selected features can be reduced by increasing the parameter t_2, used in the fitness function. However, large values for t_2 will reduce the performance (DR and FPR) of methods and increase the number of generations required to achieve a good population.

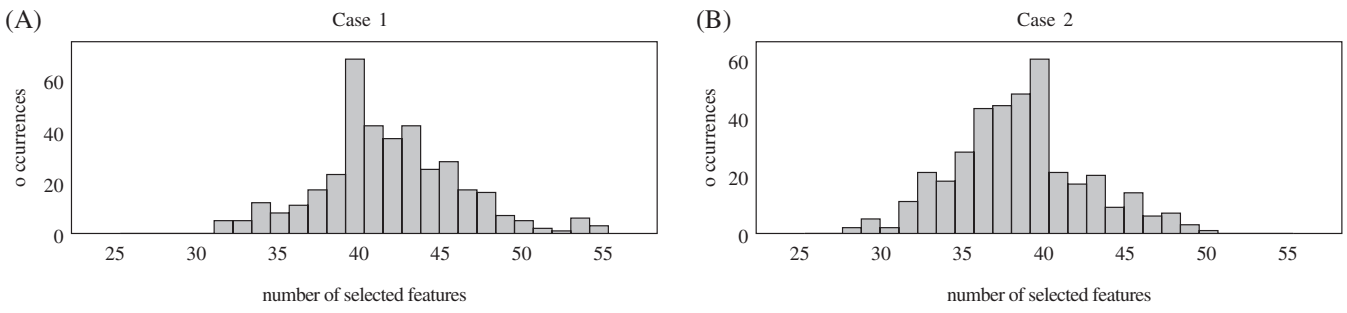

FIGURE 4 Histogram of the number of selected features considering all files and generations

Our approach initiates the data processing with parameters randomly selected in the intervals defined in Table 2. In fact, the proposed methods do not depend on a training process to be able to detect intrusions. A "training" is performed continuously to improve the performance (DR and FPR) in an adaptive fashion. The FPs and TPs identified by the security staff (or other mechanisms) in one generation is used to tune parameters for the next generation. Considering this, the number of flows used in the training is variable, depending on the number of FPs and TPs classified in the last generation. Using the observed performance and the dataset quantities presented in Table 1, it is possible to compute the amount of data used in the generations' training. To illustrate this, the generation 1 of file 1 used 0 flows for training and the first proposed method provided DR = 95.28% and FPR = 7.08%. File 1 has 432, 074 benign and 13, 835 malicious flows, thus, the generation 2 of file 1 used $13,835 \times 0.9528 = 13,182$ TPs and $432,074 \times 0.0708 = 30,591$ FPs, summing 43, 773 (9.8%) flows. Analogously, the generation 30 of file 4 used 13, 667 (4.7%) flows on its training. We point out that, as presented in the last paragraphs of section 4, a sample of this data provides an approximation with a satisfactory confidence interval. Thus, these training quantities can be adjusted in the approach's implementation, considering the resources availability and the required confidence interval.

The presented numerical simulations were performed in a Virtual Machine with 8 vCPUs (Intel Xeon E5-1620 3.70 GHz; Intel Corporation, Santa Clara, California, USA) and 25 GB of RAM. For each case, the processing iterated 200 generations over the files 1 and 30 generations over the subsequent files. Considering the dataset description, in total, it represents 153 487 340 flows extracted from a traffic of 3 TB (using the original pcap file sizes).

The cases 1 and 2 took, respectivelly, 1711 and 85 784 seconds for processing, which is equivalent to 89 706 flows/s and 14 Gbit/s, for case 1, and 1789 flows/s and 286 Mbit/s, for case 2. We point out that the used dataset provides the pcap files and the extracted flows. In our experiments, we used solely the extracted flows and the processing costs for extraction were not considered. Also, the extracted flows were integrally in the server's disks (and RAM), thus, the transmission was not considered too. However, in real settings, commonly, the extraction is performed by collectors in a scalable architecture. In addition, we note that the used architecture (Apache Spark) is scalable, thus, this performance can be improved by adding more processing nodes.

The notable better performance for case 1 is because the first method aggregates the flows and uses a simple computation approach, while the second method uses *K*-means clustering on flows separately, which is computationally more costly.

6 | FINAL REMARKS AND CONCLUSION

In this work, we propose an adaptive method supported by genetic algorithms, which provides an iterative approach to improve the performance of anomaly-based intrusion detection methods. Basically, the approach uses as input only the values for TP and FP, which are in the data commonly checked by the security staff in real settings, when treating alarmed events.

Anomaly-based methods, in general, do not provide notable performance when compared to misuse detection.[45] However, the latter requires labeled datasets for training, which may be costly. Our approach requires practically no assumption about the input data and presented remarkable results in terms of adaptability, DR, and FPR.

There are 80 network features in the dataset used in the presented experiments. However, this value can be much higher and may combine features collected in hosts or extracted from binaries.

The aggregation and profiling strategies are interesting subjects for future studies. Aggregation can be oriented to time-window, communication pairs, number of events, among others. Profiling can be based on hosts, services, users, groups of objects, and so on. Automatic approaches for defining groups can be used on profiles based on groups of objects. This may reduce the amount of data that are transmitted, persisted, and processed on profiling. Also, profiling can inherit some aspects from time series modeling, such as seasonality, the order of dependency, trends, and so on.

In the presented experiments, the profiling was created over the CSV file with normal events, which is available in the used dataset. A normal data is required to create a baseline in the profiling approach. However, normal data may not be easily available in real situations. In this case, there are some techniques that are proposed to clean the data for profiling. For example, Catania et al[46] propose an autonomous labeling approach, on which a misuse intrusion detection system is used to identify malicious flows in the network and exclude them from the dataset. Further investigation still remains to evaluate how much a negligible number of attacks in the dataset used for profiling compromises the profile quality.

In the presented experiments, we focused on showing that the adaptive model quickly converges to an evolved population and on evaluating the differences in detection varying the attacks. Thus, the considered number of generations was relatively small and attacks were considered separately. In real implementations, we recommend evolving an initial population in a large number of generations (>1000) and over a dataset with a variety of attacks combined.

Also, the system may have a database with the best historical individuals, which can be used periodically to keep a chromosome variability and fast adaptability. These individuals can be shared with other similar systems in a cooperative fashion.

The proposed adaptive approach can be coupled with other anomaly-based intrusion detection methods. We point out that a hybrid method combining different strategies may provide good results in all the presented scenarios.

Conflict of interest

The authors declare no potential conflict of interests.

ORCID

Paulo Angelo Alves Resende http://orcid.org/0000-0002-1408-8870

REFERENCES

1. Peng T, Leckie C, Ramamohanarao K. Survey of network-based defense mechanisms countering the DoS and DDoS problems. *ACM Comput Surv (CSUR)*. 2007;39(1):3.
2. IANA. IPFIX Information Elements Registered at IANA; 2007. https://www.iana.org/assignments/ipfix/ipfix.xhtml. Accessed July 31, 2017.
3. Khammassi C, Krichen S. A GA-LR wrapper approach for feature selection in network intrusion detection. *Comput Secur*. 2017;70(suppl C):255-277.
4. Holland JH. Outline for a logical theory of adaptive systems. *J ACM*. 1962;9(3):297-314.
5. Sharafaldin I, Gharib A, Lashkari AH, Ghorbani AA. Towards a reliable intrusion detection benchmark dataset. *J Softw Netw*. 2017;2017(1):177-200.
6. Fraser AS. Simulation of genetic systems by automatic digital computers. I. Introduction. *Austr J Biol Sci*. 1957;10(4):484-491.
7. Whitley D. A genetic algorithm tutorial. *Stat Comput*. 1994;4(2):65-85.
8. Kramer O. *Genetic Algorithm Essentials*. 1st ed. New York, NY: Springer; 2017.
9. Jain AK. Data clustering: 50 years beyond *K*-means. *Pattern Recogn Lett*. 2010;31(8):651-666.
10. Kriegel H-P, Kröger P, Zimek A. Clustering high-dimensional data: a survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Trans Knowl Discov Data*. 2009;3(1):1:1-1:58.
11. Peng J, Choo K-KR, Ashman H. User profiling in intrusion detection: a review. *J Netw Comput Appl*. 2016;72:14-27.
12. Bhuyan MH, Bhattacharyya DK, Kalita JK. Network anomaly detection: methods, systems and tools. *IEEE Commun Surv Tutor*. 2014;16(1):303-336.
13. Majeed PG, Kumar S. Genetic algorithms in intrusion detection systems: a survey. *Int J Innov Appl Stud*. 2014;5(3):233.
14. Stein G, Chen B, Wu AS, Hua KA. *Decision tree classifier for network intrusion detection with GA-based feature selection*. *Proceedings of the 43rd Annual Southeast Regional Conference, ACM-SE 43*. Vol 2. New York, NY: ACM; 2005:136-141.
15. Shon T, Kim Y, Lee C, Moon J. *A machine learning framework for network anomaly detection using SVM and GA*. *Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop*; West Point, NY, USA, USA: IEEE; 2005:176-183.
16. Kim DS, Nguyen H-N, Park JS. *Genetic algorithm to improve SVM based network intrusion detection system*. *19th International Conference on Advanced Information Networking and Applications (AINA'05) Volume 1 (AINA papers)*. Vol 2; Taipei, Taiwan, Taiwan: IEEE; 2005:155-158.
17. Kuang F, Xu W, Zhang S. A novel hybrid KPCA and SVM with GA model for intrusion detection. *Appl Soft Comput*. 2014;18(suppl C):178-184.
18. Kloft M, Brefeld U, Düessel P, Gehl C, Laskov P. *Automatic feature selection for anomaly detection*. *Proceedings of the 1st ACM Workshop on Workshop on AISec, AISec '08*. New York, NY: ACM, USA; 2008:71-76.
19. Chen Z, Yeo CK, BSL F, Lau CT. *Combining MIC feature selection and feature-based MSPCA for network traffic anomaly detection*. *2016 Third International Conference on Digital Information Processing, Data Mining, and Wireless Communications (DIPDMWC)*; Moscow, Russia: IEEE; 2016:176-181.
20. Balajinath B, Raghavan SV. Intrusion detection through learning behavior model. *Comput Commun*. 2001;24(12):1202-1212.
21. Pillai MM, Eloff Jan HP, Venter HS. *An approach to implement a network intrusion detection system using genetic algorithms*. *Proceedings of the 2004 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries, SAICSIT '04*. Republic of South Africa: South African Institute for Computer Scientists and Information Technologists; 2004:221.
22. Wei L. Using genetic algorithm for network intrusion detection. *Proceedings of the United States Department of Energy Cyber Security Group*. 2004;1:1-8.
23. Gong RH, Zulkernine M, Abolmaesumi P. *A software implementation of a genetic algorithm based approach to network intrusion detection*. *Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Network*; Towson, MD, USA, USA: IEEE; 2005:246-253.
24. Khan MSA. Rule based network intrusion detection using genetic algorithm. *Int J Comput Appl*. 2011;18(8):26-29.
25. Diaz-Gomez PA, Hougen DF. *Improved off-line intrusion detection using a genetic algorithm*. *ICEIS (2)*; Miami, FL, USA: The Science and Information Organization; 2005:66-73.
26. Diaz-gomez PA, Hougen DF. *A genetic algorithm approach for doing misuse detection in audit trail files*. *2006 15th International Conference on Computing*; Mexico City, Mexico: IEEE; 2006:329-338.
27. Ludovic ME. *Gassata, a genetic algorithm as an alternative tool for security audit trails analysis*. *Proceedings of the First InternationalWorkshop on the Recent Advances in Intrusion Detection, Louvain-la-Neuve, Belgium*; Louvain-la-Neuve, Belgium: Université catholique de Louvain; 1998:1-11.
28. Xia T, Qu G, Hariri S, Yousif M. *An efficient network intrusion detection method based on information theory and genetic algorithm*. *PCCC 2005. 24th IEEE International Performance, Computing, and Communications Conference, 2005*; Phoenix, AZ, USA, USA: IEEE; 2005:11-17.
29. Lee W, Stolfo SJ, Mok KW. *A data mining framework for building intrusion detection models*. *Proceedings of the 1999 IEEE Symposium on Security and Privacy (Cat. No. 99CB36344)*; Oakland, CA, USA, USA: IEEE; 1999:120-132.
30. Lee W, Stolfo SJ, Mok KW. Adaptive intrusion detection: a data mining approach. *Artif Intell Rev*. 2000;14(6):533-567.
31. Lee W, Stolfo Salvatore J. A framework for constructing features and models for intrusion detection systems. *ACM Trans. Inf Syst Secur*. 2000;3(4):227-261.
32. Lee W, Stolfo SJ, Chan PK, et al. *Real time data mining-based intrusion detection*. *DARPA Information Survivability Conference amp; Exposition II, 2001. DISCEX '01. Proceedings*. Vol 1; Anaheim, CA, USA, USA: IEEE; 2001:89-100.
33. Shafi K, Abbass HA. An adaptive genetic-based signature learning system for intrusion detection. *Expert Syst Appl*. 2009;36(10):12036-12043.
34. Callegari C, Giordano S, Pagano M. An information-theoretic method for the detection of anomalies in network traffic. *Comput Secur*. 2017;70:351-365.
35. Hamamoto AH, Carvalho LF, Sampaio LDH, Abrão T, Proença ML. Network anomaly detection system using genetic algorithm and fuzzy logic. *Expert Syst Appl*. 2018;92:390-402.

36. Zacaron AM, Carvalho LF, Adaniya MHAC, Abrão T, Proença ML Jr. *Digital signature of network segment using flow analysis*. *DCNET/ICE-B/OPTICS, Italy*; Rome, Italy: SciTePress; 2012:35-40.
37. Aziz ASA, Hanafi SE-O, Hassanien AE. Comparison of classification techniques applied for network intrusion detection and classification. *J Appl Log*. 2017;24:109-118.
38. Tran C, Vo TN, Thinh TN. *HA-IDS: a heterogeneous anomaly-based intrusion detection system*. *2017 4th NAFOSTED Conference on Information and Computer Science*; Hanoi, Vietnam: IEEE; 2017:156-161.
39. Aslahi-Shahri BM, Rahmani R, Chizari M, et al. A hybrid method consisting of GA and SVM for intrusion detection system. *Neural Comput Appl*. 2016;27(6):1669-1676.
40. Brown LD, Cai T. Tony, DasGupta Anirban. Interval estimation for a binomial proportion. *Stat. Sci*. 2001;16(2):101-133.
41. Garcia S, Grill M, Stiborek H, Zunino A. An empirical comparison of botnet detection methods. *Comput Secur J*. 2014;45:100-123.
42. Acarali D, Rajarajan M, Komninos N, Herwono I. Survey of approaches and features for the identification of HTTP-based botnet traffic. *J Netw Comput Appl*. 2016;76:1-15.
43. Silva SSC, Silva RMP, Pinto RCG, Salles RM. Botnets: a survey. *Comput Netw*. 2013;57(2):378-403.
44. Zhao D, Traore I, Sayed B, et al. Botnet detection based on traffic behavior analysis and flow intervals. *Comput Secur*. 2013;39(Pt A):2-16.
45. Sommer R, Paxson V. *Outside the closed world: on using machine learning for network intrusion detection*. *Security and Privacy (SP), IEEE Sympouism*; Berkeley/Oakland, CA, USA: IEEE; 2010:305-316.
46. Catania CA, Bromberg F, Garino CG. An autonomous labeling approach to support vector machines algorithms for network traffic anomaly detection. *Expert Syst Appl*. 2012;39(2):1822-1829.

How to cite this article: Resende PAA, Drummond AC. Adaptive anomaly-based intrusion detection system using genetic algorithm and profiling. *Security and Privacy* 2018;1:e36. https://doi.org/10.1002/spy2.36