A Machine Learning-based Approach to Build Zero False-Positive IPSs for Industrial IoT and CPS with a Case Study on Power Grids Security

Mohammad Sayad Haghighi, *Senior Member, IEEE,* Faezeh Farivar, *Member, IEEE,* Alireza Jolfaei, *Senior Member, IEEE*

***Abstract*—Intrusion Prevention Systems (IPS) have long been the first layer of defense against malicious attacks. Most sensitive systems employ instances of them (e.g. Firewalls) to secure the network perimeter and filter out attacks or unwanted traffic. A firewall, similar to classifiers, has a boundary to decide which traffic sample is normal and which one is not. This boundary is defined by configuration and is managed by a set of rules which occasionally might also filter normal traffic by mistake. However, for some applications, any interruption of the normal operation is not tolerable e.g. in power plants, water distribution systems, gas or oil pipelines, etc. In this paper, we design a learning firewall that receives labelled samples and configures itself automatically by writing preventive rules in a conservative way that avoids false alarms. We design a new family of classifiers, called λ-classifiers, that unlike the traditional ones which merely target accuracy, rely on zero false-positive as the metric for decision making. First, we analytically show why naive modification of current classifiers like SVM does not yield acceptable results and then, propose a generic iterative algorithm to accomplish this goal. We use the proposed classifier with CART at its heart to build a firewall for a Power Grid Monitoring System. To further evaluate the algorithm, we additionally test it on KDD CUP'99 dataset. The results confirm the effectiveness of our approach.**

***Index Terms*—Artificial Intelligence, Machine Learning, Industrial Control Systems, Power Systems, Firewall, IDS, Internet of Things, Cyber Physical Systems, Classification, Security.**

I. Introduction

INTRUSION Prevention Systems (IPS) are security systems that monitor and control the traffic coming in or going out of network based on a set of pre-defined rules [1]. They usually protect the perimeter of networks and keep insiders away from the malicious attacks launched from outside. The most well known examples of IPSs are firewalls.

In delay sensitive applications, where continuity of operation is vital, blocking a legitimate traffic by mistake is not tolerable. This could lead to disruption in e.g. industrial Cyber Physical Systems (CPS) or governmental services. Classic intrusion detection mechanisms rely on classification of samples and divide them into benign and malicious [2–5]. The border is usually drawn by using a cost minimization algorithm. However, this does not guarantee a zero false alarm and occasionally legitimate traffic is filtered.

Currently, firewalls are mostly configured by experts manually and with the update of attack samples, they are hardly updated, since modifying the rule sets becomes very complex especially when you do not want to tamper with the normal traffic in sensitive applications where business continuity is crucial. Leaving all the decision makings to Intrusion Detection Systems (IDS) is also problematic as IDSs are usually passive systems and have processing limitations which hinder the precise inspection of samples when they are overloaded [6–9]. Even mitigation controls such as trust management between interacting entities does not help with this problem much [10].

In this paper, we aim to design a next generation learning firewall that receives labelled samples (e.g. from a supervised Intrusion Detection System (IDS)) and configures itself automatically by writing preventive rule sets in a conservative way that avoids false alarms. In other words, we assume that malicious and normal samples are fed to an automated algorithm of writing preventing rules in the firewall so that it does not block any normal or benign traffic (zero false positive). This is vital in applications where normal operation interruption is not tolerable e.g. in governmental services or cyber-physical systems like power plants, water distribution systems, etc. We design a new family of classifiers that unlike the traditional ones which merely target accuracy, rely on zero-false-positive as the metric for decision making. We try to use iterative methods to accomplish this. The cost paid is an increase in the false negative rate which we tend to minimize.

We leave the grey area to the IDS as it may evolve and improve over the time. However, the learning firewall proactively prevents the damage compared to the passive approach of letting the traffic in and then decide. Besides, this will reduce the load of the IDS too so that it can increase the complexity or depth of analysis for the remaining traffic. At the final stage, the output of this new classifier is automatically translated into non-conflicting firewall rules. We conduct some experiments on real-world datasets obtained form industrial systems to show how our approach works.

The rest of this paper is organized as follows: In Section II we review the related work. Section III studies the problem of achieving zero false positive rates and then, presents a novel algorithm to make it possible. Section IV tests the algorithm on a power grid monitoring system as well as KDD CUP'99 datasets. Finally, the paper is concluded in Section V.

M. Sayad Haghighi (corresponding author) is with the School of Electrical and Computer Engineering, University of Tehran, Iran, and also, the School of Computer Science, Institute for Research in Fundamental Sciences (IPM). P.o.Box 19395-5746, Tehran, Iran, Email: sayad@ut.ac.ir.

F. Farivar is with the Department of Mechatronics and Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran, Email: f.farivar@srbiau.ac.ir.

A. Jolfaei is with the Department of Computing, Macquarie University, Sydney, 2109 NSW, Australia, Email: Alireza.Jolfaei@mq.edu.au

II. Related Work

Several papers focused on increasing the accuracy and speed of traffic recognition or rule enforcement [11–14]. For example, the aim of [11] is to reduce the memory usage in Internet traffic classification methods using decision trees. Similarly, in [12–14], the authors worked on the fusion of rules in Cisco and Palo Alto firewalls.

Some scientists tried to optimize the rules and their sequence of appearance. In firewalls, the rules are executed in order, and the ones appearing first have higher priority. Therefore, if a packet or data sample is to be passed (or be blocked), performance-wise, it is better to put the corresponding rules at the top. In [15, 16], the authors have worked on the optimization of rules and their order of appearance.

Due to the issue of rule preference, sometimes conflicts are found among them. Imagine a rule lets the packet pass while another one blocks it. In this case, the one that appears first in the list determines the result. Many articles have focused on eliminating such inconsistencies among firewall rules [17–20].

In a related effort, the authors of [21] presented model checking techniques and focused on finding violations of some predefined policies in the rules. They also showed how the developed methods can be used in IPv6 networks.

On a bigger scale, some researchers tried to find ways of investigating whether the firewalls of different parts in the same network are following similar policies or not, especially in industrial networks. For example, [22] has provided a semantic basis for expressing and comparing policies applied in every firewall, which in turn can form the basis for a macro judgment of compliance with (e.g. organizational) policies.

In a valuable effort, the same group [23], in addition to addressing the above problem, proposed the idea of automating firewall configuration for SCADA systems. This idea was developed in the capacity of ANSI/ISA 62443 standard which is intended to express holistic security policies. The authors tried to add extended features to the ones the standard proposes in order to enable the firewall to use policies for autoconfiguration. Although this research took a step towards autoconfiguration, it was not designed to use the learned attack patterns and was merely limited to partitioning industrial network interconnections into zones according to policies.

In [24], the researchers similarly worked on automating firewall rule configuration based on a set of given policies. In this study, Mignis tool was used for generating rules based on the descriptive semantics derived from policies. The goal was to implement high level policies and not to learn attacks for prevention purposes or to lower false positives.

In [25–27], the authors addressed the problem of learning firewall rules, but not by the network owner. They investigated it from an outsider's perspective who intends to discover the rules written in the firewall by trial and error e.g. through sending requests and interpreting the responses.

In [28], a firewall was designed based on a set of fuzzy rules. In this method, the membership level of the input packet to a set of predefined fuzzy functions is evaluated, and then the final decision (either rejection or acceptance) is made based on the aggregated information. The approach taken in this study is more like a fuzzy IDS whose decisions are enforced by an actuating IPS e.g. a firewall. However, in this paper, the problem of false decisions has not yet been addressed.

A similar research has been done in the context of http and web applications, but with learning capabilities [29]. In this approach, it is assumed that normal behavior is fed into the system in the form of XML data, and any behavior that is classified as abnormal is filtered. This approach is also being followed in the CPS context [30]. However, due to the lack of attention to false positives and false negatives, its performance is questionable in real-world industrial applications.

In [31], in addition to working on speed enhancements, moved towards the learning capability in firewalls. They used a Huffman tree of rules for this purpose. Learning happens by adaptations and changes in this tree. However, the learning criterion is merely met by crossing some hard thresholds over the measured indicators.

A network IDS based on biological immune mechanisms has been proposed in [7]. It aims to lower false positives. In this method, three monitoring agents (located in different parts of the network) provide co-stimulation signals to the intrusion detectors in order to reduce false positive alarms. Similarly, the authors in [6] used a hybrid neuro-fuzzy approach to reduce the number of false alarms in IDSs. The proposed approach was experimented with different background knowledge sets in DARPA 1999 dataset. The authors concluded through simulations that their approach required less background knowledge compared to other approaches. In a relevant try, [8] used a neural network to build an IDS that considers the cost ratio of false negative errors to false positive errors. The authors stated that compared with false positive errors, false negative errors incur a greater loss thus must be lowered.

As explained, little work has been done on the automation of connecting firewalls to IDSs. When we add the zero false positive requirement to the problem, there remains no background in the literature.

III. The Proposed Approach: Zero-False Positive Automatic Firewall Rule Generation

In this section, we first look into extending traditional classifiers like SVM to achieve zero false positives. By a few examples, we show why this approach cannot serve the purpose. Then, an iterative algorithm is presented which yields zero positive results and can accommodate almost any classifier at its core. We show how a tree-based classifier can turn the algorithm output into non-conflicting firewall rules.

A. Problem Description and Articulation

Without loss of generality, we focus on the binary classification problem. We start with the linear case. Nonlinearity can be added by e.g. using kernels later though the algorithm we propose is generic and can support almost every classifier.

We assume the two classes are nonseparable. We do not have any problem with separable cases since they lead to zero false positive results. We do not elaborate which classifier is capable of doing so at this stage. However, for the case of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIA.2020.3011397, IEEE Transactions on Industry Applications

linearly separable classes, there exists a line that correctly separates the samples with zero error as in e.g. SVM [32].

To describe the research problem, we use the linear case. However, this will be relaxed later. Fig. 1a shows a two-class set of data samples which are linearly non-separable. A linear boundary has been drawn just as an example. Let us denote the labelled (training) samples of Class 1 (+) and Class 2 (−) by $\boldsymbol{x}_i$; $i = 1, ..., N$. The goal is to find a boundary that achieves zero false positive; the statement that once decided that an input sample is +, is not wrong.

This means that no sample of Class 2 shall be classified as 1 (or +). You may imagine Class 1 is the set of attack samples and Class 2 is the set of normal ones in an industrial network security context. Therefore, no normal sample should be classified as malicious as it will be blocked and this interrupts the normal system operation. Similarly, we could consider biometric security systems. In physical security systems, like fingerprint readers, false positives (false acceptances) are not tolerable, though they come at the cost of some extra false negatives. It is preferred that the granted permissions are always correct, even if sometimes legitimate users have to re-enter their fingerprints for the reader to pick.

One can achieve a zero false positive rate, but it might come at the cost of a low true negative rate (or high false negative rate). We tend to additionally minimize false negatives. This could be translated into minimizing the number of samples in Class 1 (+) that fall on the other side of the boundary and are classified as (−). The two above-mentioned goals are not easy to achieve simultaneously. Next, we mathematically show how a classic classifier works and then in Subsection C, extend it in a try to make a zero false positive classifier.

B. Formulation of the Classic Approach (SVM)

There are different methods for formulation of linear classifiers. Normally, at some point during the development, one faces a discrete variable counting the number of misclassified samples which is usually replaced by a not-so accurate continuous approximation to keep the problem convex [32]. We formulate the binary classification problem following the classic approach for SVM as an example. However, later during the process, we diverge from the normal definitions as we are following different targets. In SVM, depending on where the training sample $\boldsymbol{x}_i$ lies with respect to the linear boundary $\boldsymbol{a}.\boldsymbol{x}_i + b_0 = 0$, we will have different conditions. If y_i is a scalar variable whose value is +1 for the + samples and -1 for the − ones, we can express these conditions with respect to the boundary as below:

- The sample is correctly classified and falls outside the gutter (band) specified by $\boldsymbol{a}.\boldsymbol{x}_i + b_0 = \pm 1$. For such samples which are shown by circles around them and lie outside the gutter in Fig.1a, we have:

$$y_i(\boldsymbol{a}.\boldsymbol{x}_i + b_0) \geq 1 \tag{1}$$

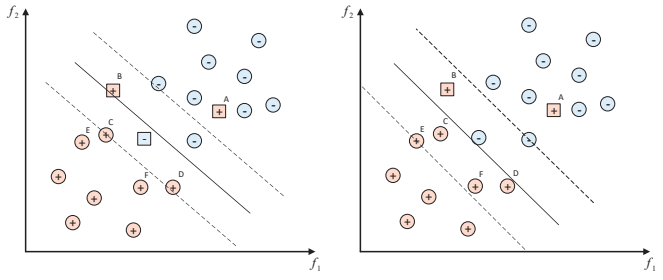
  where equality happens when the sample (support vector) is on either of the two parallel gutter lines.
- The sample is correctly classified with the given $\boldsymbol{a}$ and b_0, but lies inside the gutter:

$$0 \leq y_i(\boldsymbol{a}.\boldsymbol{x}_i + b_0) < 1 \tag{2}$$

  Again, since these are classified correctly, they are shown by circles in Fig. 1(a).
- The sample is classified in the wrong class:

$$y_i(\boldsymbol{a}.\boldsymbol{x}_i + b_0) < 0 \tag{3}$$

  These samples are shown by squares in Fig. 1(a).



(a) (b)

Fig. 1. (a) Traditional classification of linearly non-separable classes in a trade of between gutter and the weight of misclassified samples. The circles show correctly classified samples and the square ones are misclassified. (b) Zero false-positive classification achieved through different criteria.

Some rewrite all the three as one equation with slack variables:

$$y_i(\boldsymbol{a}.\boldsymbol{x}_i + b_0) \geq 1 - \eta_i \tag{4}$$

where $\eta_i = 0$ in the first case, $0 < \eta_i \leq 1$ in the second, and $\eta_i > 1$ in the third. Usually, the classification goal is to make the gutter as wide as possible while keeping the number of misclassified samples minimal. With the above formulation, the gutter width will be $2/||\boldsymbol{a}||$ [32], where $\|.\|$ stands for the Euclidean norm operation. For mathematical convenience, it is usual to minimize $\frac{1}{2}||\boldsymbol{a}||^2$ instead. The cost function for such a minimization problem would be:

$$J(\boldsymbol{a}, b_0, \boldsymbol{\eta}) = \frac{1}{2}||\boldsymbol{a}||^2 + c\sum_{i=1}^{N} \mathcal{F}(\eta_i) \tag{5}$$

$$\mathcal{F}(\eta_i) = \begin{cases} 1, & \text{if } \eta_i > 0 \\ 0, & \text{if } \eta_i = 0 \end{cases} \tag{6}$$

To remove the discontinuous functions and make the problem convex [32], $\sum_{i=1}^{N} \mathcal{F}(\eta_i)$ is usually approximated by $\sum_{i=1}^{N} \eta_i$, which does not actually minimize the number of misclassified samples, but rather a cost function related to that:

$$\begin{aligned} &\underset{\boldsymbol{a}, b_0}{\arg\min}\ J(\boldsymbol{a}, b_0, \boldsymbol{\eta}) = \frac{1}{2}||\boldsymbol{a}||^2 + c\sum_{i=1}^{N} \eta_i \\ &sbj.\ to: \quad y_i(\boldsymbol{a}.\boldsymbol{x}_i + b_0) \geq 1 - \eta_i \quad , \quad \eta_i \geq 0 \end{aligned} \tag{7}$$

where the positive constant c specifies the importance of the second term compared to the gutter width. This was a short introduction to the classic problem formulation.

C. Extension of the Classic Approach (ʒ-SVM)

The above formulation does not guarantee a zero false positive rate. Fig. 1a shows a linearly non-separable constellation in which some samples from each class have been misclassified. Obviously, in Eq. (8), the optimization algorithm does not have any preference over the classes. We can extend the classic formulation towards giving preference to one class and outputting zero false-positive results. If we break down the number of training samples as $N = n_n + n_p$, we can write:

$$\begin{aligned} \underset{\boldsymbol{a},b_0}{\arg\min}\ J(\boldsymbol{a},b_0,\boldsymbol{\eta}) &= \frac{1}{2}||\boldsymbol{a}||^2 + c_1\sum_{i=1}^{n_n}\eta_i + c_2\sum_{i=n_n+1}^{N}\eta_i \quad (8)\\ sbj.\ to: &\quad y_i(\boldsymbol{a}.\boldsymbol{x}_i + b_0) \geq 1-\eta_i \\ &\quad \eta_i \geq 0 \end{aligned}$$

where $c_1 > c_2$. We refer to this classifier as ʒ-SVM. The goal is to achieve something like Fig. 1b with zero false positives while keeping the false negatives as low as possible.

Notice that this approach does not work based on the notion of counting, so for example, in sacrificing the other class samples, it might pick some whose distances are closer to the (hypothetical) boundary rather than one that is a bit farther. This happens due to the approximation (discrete to continuous) we make during the development. What we want is to achieve zero false positive with minimum number of false negatives. In the next subsection, we discuss each of such issues in detail.

D. The Issues with ʒ-SVM

In the previous subsection, we developed a convex formulation based on the classic approach to increase the cost of misclassification for "−" samples. By choosing $c_1 \gg c_2$, one can make the optimization algorithm try to zero the number of misclassifications for the negative samples. However, there are a number of issues with this naive approach which can make it impractical in certain scenarios.

1) Continuous Error Approximation: The optimization algorithm is blind to the details of the problem. In certain cases, it deceives us by manipulating the gutter width. Based on Eq. (8), the classification error contributing to the cost function is the sum of η_i ; $i = 1, ..., N$. When we want to increase c (e.g. c_1, hoping to achieve zero false positive), the optimizer tends to make the gutter width smaller to virtually reduce the errors (see Eq. (8)). This is due to the fact that η_i is a function of the gutter width itself. Fig. 2a to 2c demonstrate this fact.

2) Weight Coefficients Effect: ʒ-SVM is highly sensitive to c_1 and c_2 values. It is not just their ratio that matters, their absolute values are also important along with $||\boldsymbol{a}||$.

There are three cost components in Eq. (8). If we choose a big value for c_1 and a rather small one for c_2, we *might* achieve a zero false positive rate. However, there is still a probability that this does not happen. For example, if exists a "−" sample deep inside the positive class region, it will contribute a large η_i whose cost is further boosted by c_1. However, to make the false positive zero, the algorithm shall move the boundary line and misclassify some positive samples. Each of these samples (let us say j) adds $c_2\eta_j$ to the cost. If the number of such samples is high, their cumulative cost might surpass $c_1\eta_i$. Therefore, the minimization algorithm might not output a zero false-positive result. This is highly dependent on the constellation of samples, how the classes are mixed, and the choice of c_1 and c_2. Fig. 2d demonstrates this issue.

A very large value for c_1 can solve this issue, but it induces another problem. By choosing such a value for c_1, the dominant component in the cost function will be the second term. This implies that the gutter width and the + samples misclassification error will have little or no effect. This might for example lead to a boundary that misclassifies too many positive sample in order to keep $c_1\sum_{i=1}^{n_n}\eta_i$ minimum. Fig. 2e to 2g show this phenomenon. Comparison of Fig. 2f and Fig. 2h which have the same c_1/c_2 ratio shows that the third cost component, i.e. $||\boldsymbol{a}||$ can significantly change the final result. Note that in the settings achieving zero false positive rates, the number of misclassified positive samples are significantly different, ranging from 9 to 16, with Fig. 2e having the minimum number of such samples. The cost coefficients in Fig. 2e are trivially different from those in Fig. 2f, yet the results are very different. This shows how sensitive the final result is to the choice of c_1 and c_2.

3) Sample Unbalance Problem: The training set unbalance problem is well known. This is not something specific to ʒ-SVM. Imagine that there are merely a few negative samples and too many positive ones. In such occasions, the cost of misclassification for the negative samples is practically capped and the optimizer might decide to minimize the cost function based on the dominant factor defined by the too-many positive samples. This implies that we need to adjust c_1/c_2 per case.

The above discussions showed some of the problems of ʒ-SVM; an example of traditional classifiers extension. Many of these problems pertain to the fact that the counting function (e.g. of Eq. (6)) is replaced with an approximate alternative in the convex problem development process. One can of course stick with original optimization problems (e.g. Eq. (5)) and try to extend them for zero false positive goal while honoring the discrete constraints (like Eq. (6)). However, the result will be a non-convex or discrete problem at the end.

In the next section, we propose a novel algorithm for this purpose. We use classic approaches at the core of the algorithm but iteratively prune the samples based on its outputs to achieve zero false positive rates with minimum number of false negatives. Minimizing the number of false negatives will allow the firewall to capture as many malicious samples as possible and lower the load of the IDS, if ever installed. Finally, using a tree classifier, we test our algorithm and show how its output can be translated into simple non-conflicting firewall rules.

Given a classifier, false positives can be avoided if one sacrifices some of the positive samples. We suggest that this is done by removal. It can be proven that a removing strategy always converges to a zero-false positive result. However, the question is which positive samples shall be sacrificed to make this happen. We can represent the issue as a binary satisfiability problem. Imagine the positive samples are the set of variables, each being represented by a bit. A value of 0 for a bit implies that the corresponding sample shall be removed from the set. Regarding the example of Fig. 1, this binary coding will be like $\{b_A, b_B, b_C, b_D, b_E, ...\}; b_i \in \{0, 1\}$. Now,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIA.2020.3011397, IEEE Transactions on Industry Applications

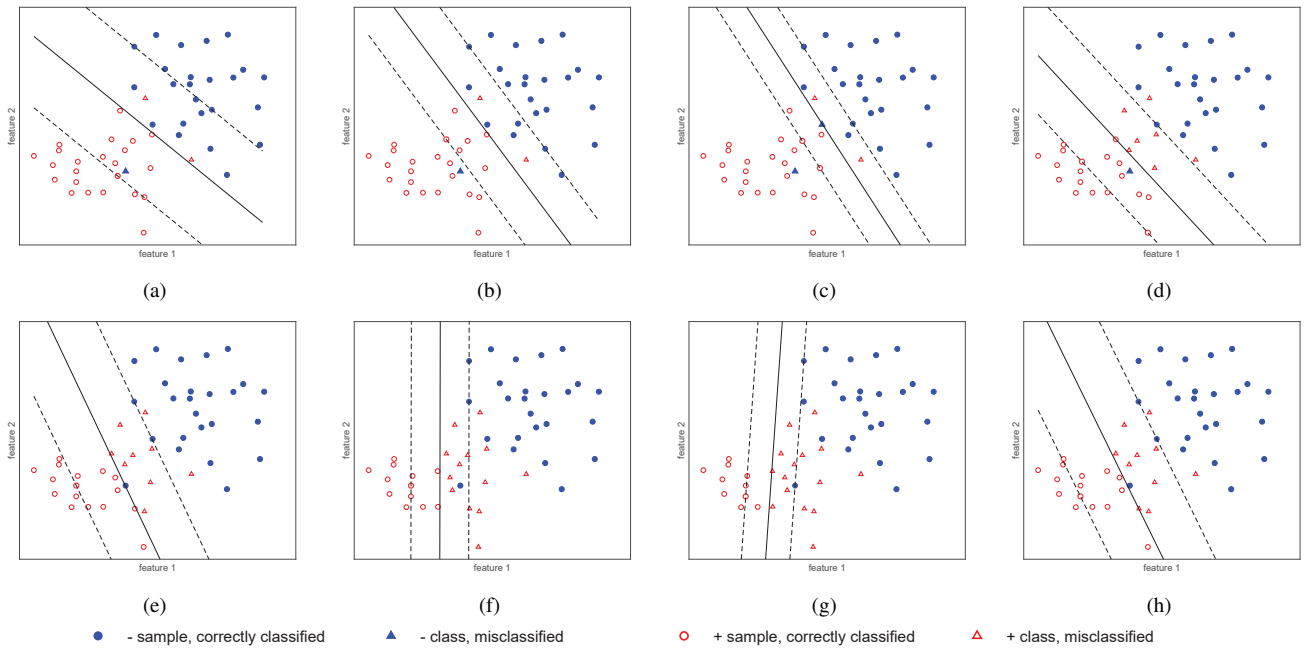


Fig. 2. The effect of cost coefficient (c) on ʒ-SVM result: (a) $c_1 = 1$, $c_2 = 1$ (b) $c_1 = 5$, $c_2 = 5$ (c) $c_1 = 50$, $c_2 = 50$ (d) $c_1 = 500$, $c_2 = 50$ (e) $c_1 = 785$, $c_2 = 50$ (f) $c_1 = 800$, $c_2 = 50$ (g) $c_1 = 1000$, $c_2 = 50$. (h) $c_1 = 80$, $c_2 = 5$.

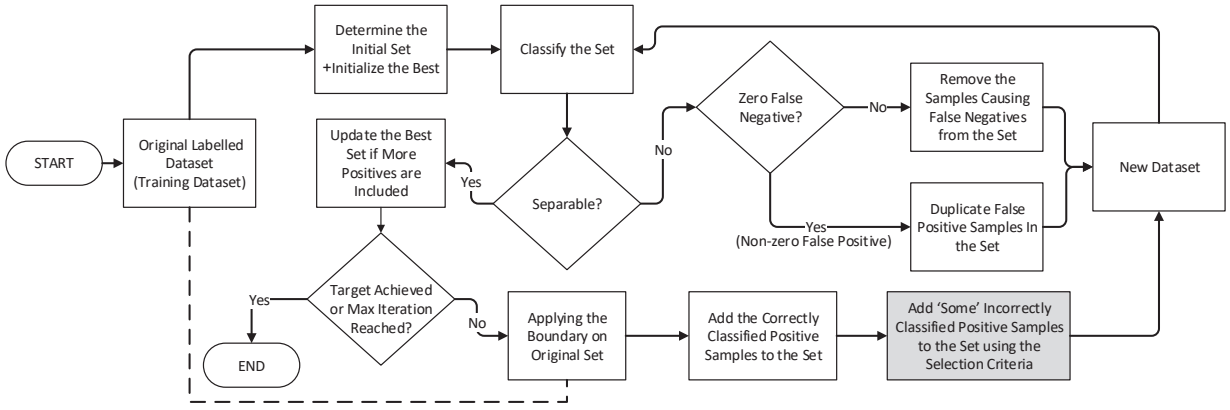


Fig. 3. Flowchart of the proposed iterative classification algorithm (ʒ-Classifier). The goal is to achieve zero false positive rate.

given the classifier $\mathcal{C}$, the remaining set can be tested to see if its samples satisfy separability. Therefore, the problem is analogous to a boolean satisfiability (SAT) problem. It can be proven that the problem is always satisfiable. In general, finding an assignment of zeros and ones that has the minimum number of variables set to one (or similarly to zero, as in our case) and satisfies the problem is NP-complete, even for a satisfiable 2-SAT problem [33]. This is sometimes called Min Ones k-SAT problem in the literature. The solution is not always unique though. For example, in Fig. 1 and with a linear $\mathcal{C}$, both $\{0, 0, 1, 1, 1, 1, ...\}$ and $\{0, 1, 1, 0, 1, 1, ...\}$ will satisfy the separability criterion thus can be acceptable. Although there could be several answers, some additional constraints in the cost function (like maximum gutter width or distance to the border) can reduce the set of answers.

E. The Proposed Zero-FP Classifier (ʒ-Classifier)

The proposed algorithm is an iterative classification method which achieves a zero false detection rate for the positive class. It is a swarm algorithm whose particles are reduced datasets. Each reduced dataset consists of all the negative samples and some of the positive ones. So, each particle can be represented by a binary code similar to before i.e. $\{b_1, b_2, ..., b_{n_p}\}; b_i \in \{0, 1\}$, where for simplicity, we have removed the 1s of the negative class members as they are always present. Each particle follows the algorithm illustrated in Fig. 3. The grey box is where it uses the knowledge obtained by other particles.

Each particle starts with an initial dataset which is derived from the original one by removing some of the positive samples in order to become separable. Ideally, this initial dataset has only one positive sample, and all the negative ones.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIA.2020.3011397, IEEE Transactions on Industry Applications

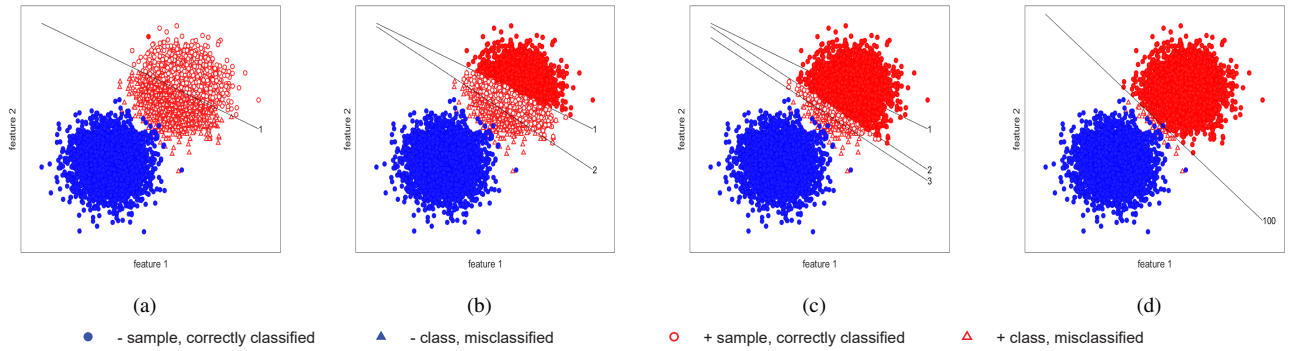


Fig. 4. Demonstration of zero false positive classification using the proposed technique for one particle with SVM as $\mathcal{C}$. The goal is to find a boundary that does not misclassify the negative samples (left/blue). Bold samples indicate inclusion in the particle's set. (a) iteration No. 1 (b) No. 2 (c) No. 3 (d) No. 100.

Given the classifier $\mathcal{C}$, one can do an initial classification on the original dataset and use the true positives as candidates to make this initial dataset. Every particle's reduced dataset goes through a classification by using $\mathcal{C}$. In each iteration, if the result shows separability of the set (zero false positive and zero false negative), the set is compared against the best result obtained by other particles so far. The comparison criterion could be a predefined fitness function e.g. the number of positive samples included as we intend to minimize the number of false negatives. If the particle's reduced set yields a better score, the best record is replaced with its set. Then, the boundary model obtained through the process is applied to the original dataset (as a test set) and the new true positives are tentatively added to the particle's set for the next iteration. This is done by setting their flags to 1. Moreover, some extra positive samples that are misclassified by this boundary are also selected and added to the set. There can be different selection criteria for this purpose, however, we suggest that one uses the global best (as in Particle Swarm Optimization) for this purpose. For example, one can use weighted random selection to pick k samples from the set of falsely classified positive samples (of the original set) and add them to the particle's set. Obviously, the positive samples included in the best set should have more weight in this selection.

Throughout the iterations, if a particle's set becomes inseparable at some point, it will undergo a pruning operation. The pruning is initially done by removing the positive samples that have created false negatives. If this does not help (which will be known in the next iteration(s)) or the source of inseparability is non-zero false positive, the weight of the negative samples contributing to the false positives is increased. This can be done by e.g. duplicating those negative samples in the particle's dataset. The reduced dataset of particle is not necessarily a subset of the original dataset anymore, but they are related. The pruning continues over iterations until the set becomes separable again. The whole addition/removal process terminates when either the target false negative is met or the maximum iteration is reached.

Algorithm 1 shows the $\mathfrak{z}$-Classifier. Notice that each time it tries to minimize the false negatives, depending on which ones it adds or removes, a different path is taken. So, we will not necessarily get the global optimum (real minimum number of false negatives). As proven in Appendix A, no matter which way we go, we always converge to a solution.

Fig. 4 depicts the first three steps as well as the final step of the proposed algorithm applied to a 2D synthetic dataset with 7500 samples. In this case, SVM has been used as the core classifier. The algorithm output could be directly translated into firewall rules if a tree-based classifier was taken.

IV. Evaluation Results and Discussions

Here, we evaluate $\mathfrak{z}$-Classifier by testing it on two datasets. We take the CART algorithm as our core classifier. Classic firewalls can define rules by combining clauses that put constraints on the values of features/dimensions. However, each clause merely constrains one feature. For such construction, even simple linear classifiers like SVM are inappropriate

Algorithm 1 The Proposed Iterative Classification Technique with Zero False Positive ($\mathfrak{z}$-Classifier)

Require: Training Set($\mathcal{T}$), Performance Target/Stop Criterion
Ensure: Zero False Positive Classification

```
Create initial separable particle sets S_1, ...S_k
Determine the Best set (with the most positive samples)
while Stop Criterion/Performance Target is not met do
  for each S_i do
    Classify S_i with C and find the boundary B_i
    if S_i is separable then
      Update the Best if S_i is more fit
      Apply B_i to T & add the true positives to S_i
      Use the Best and S_i knowledge to add k_i false positive samples to S_i
    else
      if there is false negatives then
        Remove false negative samples from S_i
      else
        Duplicate false positive samples in S_i
      end if
    end if
  end for
end while
```

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIA.2020.3011397, IEEE Transactions on Industry Applications

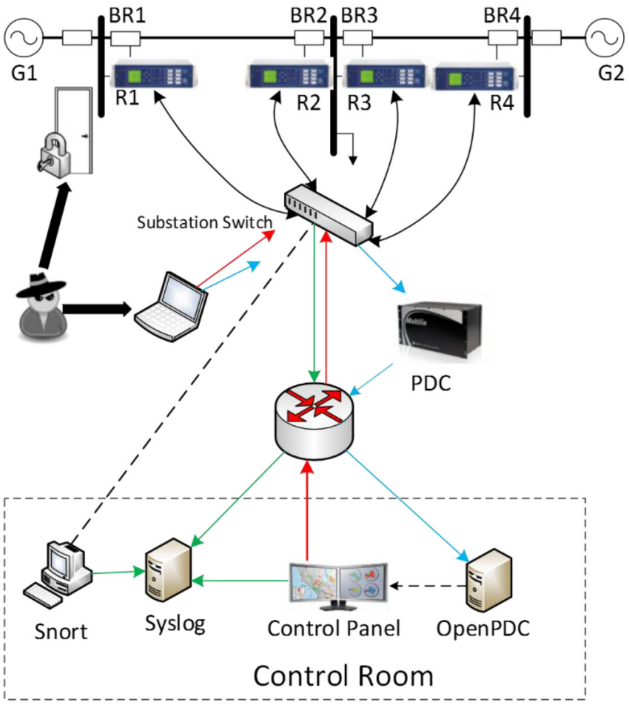


Fig. 5. Lab set of the power system Mississippi State University and Oak Ridge National Laboratory used for data collection [34].

since their output rules are functions of the features. In the simulations, the population size was 5 and each time k_i was increased by a factor of 1.5 (starting from 1) for faster convergence. It was reset to 1 every time the set became non-separable.

A. Power System (Smart Grid) Dataset

Smart grids or power systems have supervisory control systems that interact with different smart electronic devices. They are complemented by network monitoring devices such as SNORT and Syslog. Some researchers created a dataset of attacks launched on power systems [34–36]. In their work, it was assumed that an actor gained access to a substation network and posed an insider threat by issuing commands from the substation switch. The scenarios studied were (1) Short-circuit fault, (2) Line maintenance (3) Remote tripping command injection (Attack), (4) Relay setting change (Attack), and (5) Data Injection (Attack) [34–36]. The first two groups are considered "Natural" events. Each of these scenarios has sub-scenarios; 28 of them are attack scenarios, 8 are natural evens and one represents flawless operation scenario.

Fig. 5 shows the power system setup used by Mississippi State University and Oak Ridge National Laboratory to generate these scenarios [34–36]. Here, G1 and G2 are power plants. R1 to R4 are Intelligent Electronic Devices (IEDs) which can switch the breakers BR1 to BR4 on or off. Power Line 1 goes from BR1 to BR2 and power Line 2 goes from BR3 to BR4. The IEDs employ a remote protection program which switches the breakers on any detected fault, whether valid or invalid. System operators can manually issue commands to the IEDs too (to turn the breakers on or off). Manual override happens when a maintenance is carried out. Each collected sample has 128 features 29 of which are the measurements from each phasor measurement unit (PMU). A PMU is a device that measures the electricity waves on a grid. In Fig. 5, there are four PMUs (integrated with the IEDs $R_i, i = 1, ..., 4$) and each measures 29 features. So altogether, we have 116 PMU measurements. Each feature has a coding, for example, R1-PA1:VH means Phase A voltage phase angle measured by PMU R1. There are 12 more features for control panel logs, Snort alerts and relay logs of the PMUs/relays. The label in this dataset is a detailed coded one showing some characteristics of the fault/attack.

We divide the scenarios into malicious (28) and benign (8+1). We use a subset (No. 15) of the dataset with 1861 normal and 3415 attack samples. The results of applying ʒ-Classifier on this training dataset are shown in Table I. As seen, the proposed classifier always maintains a zero false positive output. Translation of the ʒ-Classifier output to firewall rules is a trivial task now. To simplify the presentation in the paper, we report the rules after 20 iterations. The first few rules of an allow-by-default firewall will be like:

$$\text{If } x_{32} < 167.6 \ \& \ x_{33} < 118145 \ \& \ x_{25} \geq \ \ 60.5 \ \rightarrow \text{REJECT}$$
$$\text{If } x_{32} < 167.6 \ \& \ x_{33} \geq 118145 \ \& \ x_{30} \geq -58.6 \rightarrow \text{REJECT}$$
$$\vdots$$

in which x_{32} ('R2-PA2:VH') stands for the Phase B voltage phase angle measured by PMU R2, x_{33} ('R2-PM2:V') is the Phase B voltage phase magnitude measured by PMU R2, x_{25} ('R1:F') is the frequency of the relay measured by PMU R1, and x_{30} ('R2-PA1:VH') represents Phase A voltage phase angle measured by PMU R2. Note that the rules do not always translate into human-understandable clauses, and in different runs of the algorithm, one might come up with different rules.

B. Results of KDD CUP'99 Dataset

KDD Cup'99 dataset has been developed by MIT Lincoln Laboratory in 1999 [37]. Ever since it was created, it has been used in many machine learning and artificial intelligence applications especially in the context of intrusion detection. We use a subset of the dataset that has 10% of the total records (494,021). Every record has 41 features. We changed all the 24 different attack labels to 'malicious' so that the algorithm is left with an ACCEPT or REJECT decision for each sample. The resultant decision tree is depicted in Fig. 6. Since the complete tree was too large to be plotted, we have reported

TABLE I
ʒ-CLASSIFIER PERFORMANCE ON THE POWER GRID DATASET OF [34–36]

Iterations	TN	TP	FN	FP
10	1861	320	3095	0
50	1861	888	2527	0
100	1861	2211	1204	0
500	1861	3231	184	0
1000	1861	3353	62	0

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIA.2020.3011397, IEEE Transactions on Industry Applications

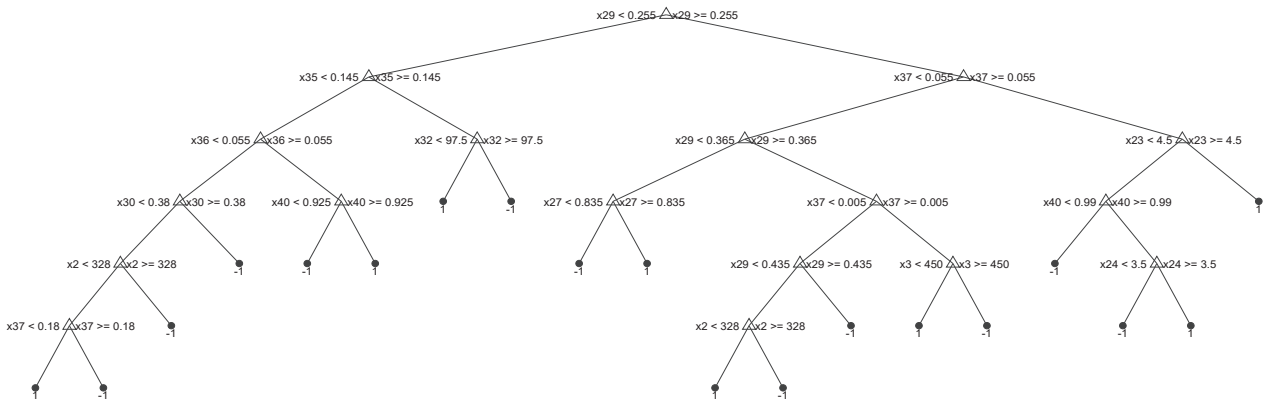


Fig. 6. Demonstration the tree output of ʒ-Classifier at the end of the 10th iteration for KDD CUP'99 dataset. +1 = malicious class -1 = normal class.

the output after 10 iterations. Even this simple classifier can capture 138,869 out of 204,458 malicious samples with zero false positive. The complete tree after 1000 iterations yields much better results which are reported in Table II. Please note that just like any learning classifier, ʒ-Classifier is also prune to over-fitting. We noticed over-fittings after the 100th iteration with the KDD CUP'99 dataset. One can adopt traditional best practices to avoid this phenomenon.

The first few rules of a default-allow firewall will be like:

If $x_{29} < 0.255$ & $x_{35} \geq 0.145$ & $x_{32} < 97.5 \rightarrow$ REJECT
If $x_{29} \geq 0.255$ & $x_{37} \geq 0.055$ & $x_{23} \geq 4.50 \rightarrow$ REJECT
$\vdots$

According to the dataset description, x_{29} is 'same_srv_rate', x_{35} represents 'dst_host_diff_srv_rate', x_{32} stands for 'dst_host_count', and x_{37} shows 'dst_host_srv_diff_host_rate'.

V. Conclusion

It is preferred that attacks are filtered at intrusion preventions systems such as firewalls as much as possible. But rejecting legitimate traffic by mistake is not tolerable in real-time industrial systems e.g. power systems. In this study, a new classifier has been introduced that can help in building self-organizing learning firewalls. Current classifiers tend to minimize a generic cost function which does not necessarily yield zero false positive results. In this paper, we have proposed an algorithm that can turn any generic classifier into a zero false positive one. One can use this classifier to build self-organizing and learning firewalls. In the design of this new approach, zero false positive (or negative) rate has been set as the goal, while false negative (or positive) rate is kept minimum. The designed classifier was tested on two datasets; a power grid dataset and KDD CUP'99 dataset. As a future work, we tend to give the algorithm boundary awareness so that it does not draw the line very close to the training samples that potentially lead to false positives in the test phase.

TABLE II
ʒ-Classifier performance on the KDD CUP'99 dataset.

Iterations	TN	TP	FN	FP
10	289,542	138,869	65,589	0
50	289,542	204,167	291	0
100	289,542	204,425	33	0
500	289,542	204,432	26	0
1000	289,542	204,432	26	0

Appendix A
Satisfiability with the Removal Strategy

Theorem 1. Given a dataset with two classes and a sound classifier like $\mathcal{C}$, an iterative removal strategy that takes the false positive samples out in each iteration will always lead to a zero false positive result.

Proof. Assume that the training dataset is (x_i, y_i) where $i = 1, 2, ..., m$ and $y_i \in \{1, -1\}$. The zero FP algorithm applies a sound classifier $\mathcal{C}$ as the core classifier which minimizes a predefined cost function like J.

The algorithm starts on the complete and original dataset. After classification, there are two types of misclassified samples which make the errors. The first one corresponds to false negatives ($E^{P\rightarrow N}$), and the second one is related to false positives ($E^{N\rightarrow P}$). The cost function of the classifier is $J(E_0^{P\rightarrow N}, E_0^{N\rightarrow P})$, where the 0 superscript shows the iteration number. According to the zero FP algorithm, the positive samples which make $E^{P\rightarrow N}$ must be removed from the original data set at this stage. After applying the classifier boundary to the new data set (after removal), $E^{P\rightarrow N}$ will be zeroed. Hence, the cost function of the zero FP algorithm at this stage becomes $J(E_1^{N\rightarrow P}) < J(E_0^{P\rightarrow N}, E_0^{N\rightarrow P})$. This inequality can be rewritten as,

$$J(E_1^{N\rightarrow P}) = r_1 \ J(E_0^{P\rightarrow N}, E_0^{N\rightarrow P}) \tag{9}$$

where $0 \leq r_1 \leq 1$. Sequentially, the sound classifier provides a new classification boundary for the modified dataset whose cost function is $J(E_2^{P\rightarrow N}, E_2^{N\rightarrow P})$. Since the dataset is modified, $(\mathcal{C})$ tried to minimize the misclassification errors so that it is less than (or in the worst case equal to) the previous iteration, i.e. $J(E_2^{P\rightarrow N}, E_2^{N\rightarrow P}) < J(E_1^{N\rightarrow P})$. Hence,

$$J(E_2^{P\rightarrow N}, E_2^{N\rightarrow P}) = q_1 \ J(E_1^{N\rightarrow P}) \tag{10}$$

where $0 \leq q_1 \leq 1$. Similarly, the algorithm removes the positive samples which have been misclassified. At the k^{th} iteration, the cost function will be,

$$J(E_k^{P\to N}, E_k^{N\to P}) = q_k \; J(E_{k-1}^{N\to P}); \quad 0 \leq q_k \leq 1 \qquad (11)$$

and at the $k+1^{th}$ one,

$$J(E_{k+1}^{N\to P}) = r_k \; J(E_k^{P\to N}, E_k^{N\to P}); \quad 0 \leq r_k \leq 1 \qquad (12)$$

This way, at each iteration a reduced set is prepared for the next iteration. By substituting Eq. (11) in Eq. (12),

$$J(E_{k+1}^{N\to P}) = (r_k \; q_k) \; J(E_{k-1}^{N\to P}) \qquad (13)$$

Similarly,

$$J(E_{k+1}^{N\to P}) = \big(\Pi_{k=1}^{N}(r_k \; q_k)\big) \; J(E_0^{P\to N}, E_0^{N\to P}) \qquad (14)$$

As $\Pi_{k=1}^{N}(r_k \; q_k)$ goes to zero, we have,

$$\lim_{k\to\infty} J(E_{k+1}^{N\to P}) = 0 \qquad (15)$$

Notice that $\max(N)$ is the number of positive samples. Therefore, the cost function of the zero FP algorithm with a sound cost-minimizing classifier will converge to zero, and all negative samples will be truly classified.

ACKNOWLEDGMENT

This research was in part supported by a grant from Institute for Research in Fundamental Sciences (IPM) (CS1397-4-77).

REFERENCES

[1] X. Zhang, C. Li, and W. Zheng, "Intrusion prevention system design," in *Int. Conf. on Computer and Information Technology*, 2004, pp. 386–390.

[2] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[3] F. Farivar, M. S. Haghighi, A. Jolfaei, and M. Alazab, "Artificial intelligence for detection, estimation, and compensation of malicious attacks in nonlinear cyber physical systems and industrial iot," *IEEE transactions on industrial informatics*, 2019.

[4] S. Barchinezhad and M. S. Haghighi, "Compensation of linear attacks to cyber physical systems through arx system identification," in *the 10th Information and Knowledge Technology Conference*, 2019.

[5] F. Farivar, M. Sayad Haghighi, A. Jolfaei, and W. Sheng, "On the security of networked control systems in smart vehicle and its adaptive cruise control," *IEEE Trans. on Intelligent Transportation Systems*, in press, 2020.

[6] R. Alshammari, S. Sonamthiang, M. Teimouri, and D. Riordan, "Using neuro-fuzzy approach to reduce false positive alerts," in *Conf. on Communication Networks and Services Research*, 2007, pp. 345–349.

[7] Y. Qiao and X. Weixin, "A network ids with low false positive rate," in *Cong. on Evolutionary Computation*, vol. 2, 2002, pp. 1121–1126.

[8] D. Joo, T. Hong, and I. Han, "The neural network models for ids based on the asymmetric costs of false negative errors and false positive errors," *Expert Systems with Applications*, vol. 25, no. 1, pp. 69–75, 2003.

[9] M. S. Haghighi and K. Mohamedpour, "Neighbor discovery: Security challenges in wireless ad hoc and sensor networks," in *Trends in Telecom. Tech.* Intech, 2010.

[10] A. Arabsorkhi, M. Sayad Haghighi, and R. Ghorbanloo, "A conceptual trust model for the internet of things interactions," in *the 8th International Symposium on Telecommunications*, 2016, pp. 89–93.

[11] Y.-C. Cheng and P.-C. Wang, "Packet classification using dynamically generated decision trees," *IEEE Trans. on Comp.*, vol. 64, no. 2, pp. 582–586, 2015.

[12] C. Diekmann, J. Michaelis, M. Haslbeck, and G. Carle, "Verified iptables firewall analysis," in *IFIP Networking Conf.*, 2016, pp. 252–260.

[13] P. Kadam and V. Bhusari, "Review on redundancy removal of rules for optimizing firewall," *Int. Jour. of Research in Engineering and Technology*, 2014.

[14] L. Zhang and M. Huang, "A firewall rules optimized model based on service-grouping," in *Web Information System and Application Conf.*, 2015, pp. 142–146.

[15] Z. Trabelsi, L. Zhang, and S. Zeidan, "Dynamic rule and rule-field optimisation for improving firewall performance and security," *IET Inf. Sec.*, vol. 8, no. 4, pp. 250–257, 2014.

[16] Y. Zhu, "Optimization design and implementation of gateway based on firewall for access control," in *Int. Conf. on Inf. Science and Tech.*, 2016, pp. 100–104.

[17] E. S. Al-Shaer and H. H. Hamed, "Discovery of policy anomalies in distributed firewalls," in *IEEE Conf. of Comp. and Comm. Societies*, 2004, pp. 2605–2616.

[18] S. Khummanee, A. Khumseela, and S. Puangpronpitag, "Towards a new design of firewall: Anomaly elimination and fast verifying of firewall rules," in *Conf. on Computer Science and Software Engineering*, 2013.

[19] F. A. Maldonado-Lopez, E. Calle, and Y. Donoso, "Detection and prevention of firewall-rule conflicts on software-defined networking," in *Int. Workshop on Reliable Networks Design and Modeling*, 2015.

[20] A. Bouhoula and A. Yazidi, "A security policy query engine for fully automated resolution of anomalies in firewall configurations," in *IEEE Symp. on Network Computing and Applications*, 2016, pp. 76–80.

[21] C. Lorenz and B. Schnor, "Policy anomaly detection for distributed ipv6 firewalls," in *Int. Conf. on e-Business and Telecommunications*, 2015.

[22] D. Ranathunga, M. Roughan, P. Kernick, and N. Falkner, "Malachite: Firewall policy comparison," in *IEEE Symp. on Computers and Communication*, 2016, pp. 310–317.

[23] D. Ranathunga, M. Roughan, H. Nguyen, P. Kernick, and N. Falkner, "Case studies of scada firewall configurations and the implications for best practices," *IEEE Tran. on Net. and Serv. Manag.*, vol. 13, no. 4, pp. 871–884, 2016.

[24] P. Adao, R. Focardi, J. D. Guttman, and F. L. Luccio, "Localizing firewall security policies," in *IEEE Computer Security Foundations Symp.*, 2016, pp. 194–209.

[25] M. Q. Ali, E. Al-Shaer, and T. Samak, "Firewall policy reconnaissance: Techniques and analysis," *IEEE Trans. on Information Forensics and Security*, vol. 9, no. 2, pp. 296–308, 2014.

[26] A. X. Liu, A. R. Khakpour, J. W. Hulst, Z. Ge, D. Pei, and J. Wang, "Firewall fingerprinting and denial of firewalling attacks," *IEEE Trans. on Inf. Forensics and Security*, vol. 12, no. 7, pp. 1699–1712, 2017.

[27] F. Chen, A. X. Liu, J. Hwang, and T. Xie, "First step towards automatic correction of firewall policy faults," *ACM Trans. on Autonomous and Adaptive Systems*, vol. 7, no. 2, p. 27, 2012.

[28] S.-U. Lar, X. Liao, A. ur Rehman, and M. Qinglu, "Proactive security mechanism and design for firewall," *Journal of Inf. Sec.*, vol. 2, no. 03, p. 122, 2011.

[29] C. Torrano-Gimenez, A. Perez-Villegas, and G. Alvarez, "A self-learning anomaly-based web application firewall," *Computational Intelligence in Security for Information Systems*, pp. 85–92, 2009.

[30] F. Farivar, M. Sayad Haghighi, S. Barchinezhad, and A. Jolfaei, "Detection and compensation of covert service-degrading intrusions in cyber physical systems through intelligent adaptive control," in *IEEE International Conference on Industrial Technology*, 2019.

[31] A. S. Sairam, R. Kumar, and P. Biswas, "Implementation of an adaptive traffic-aware firewall," in *Int. Conf. on Security of Information and Networks*, 2014, p. 385.

[32] S. Theodoridis and K. Koutroumbas, *Pattern Recognition (4th ed.)*. Academic Press, 2009.

[33] N. Misra, N. Narayanaswamy, V. Raman, and B. S. Shankar, "Solving min ones 2-sat as fast as vertex cover," *Theoretical Comp. Sci.*, vol. 506, pp. 115–121, 2013.

[34] Power System Attack Datasets, Mississippi State University and Oak Ridge National Laboratory, 2014. [Online]. Available: http://www.ece.uah.edu/~thm0009/icsdatasets/PowerSystem_Dataset_README.pdf

[35] S. Pan, T. Morris, and U. Adhikari, "Developing a hybrid intrusion detection system using data mining for power systems," *IEEE Trans. on Smart Grid*, vol. 6, no. 6, pp. 3104–3113, 2015.

[36] R. C. B. Hink, J. M. Beaver, M. A. Buckner, T. Morris, U. Adhikari, and S. Pan, "Machine learning for power system disturbance and cyber-attack discrimination," in *Int. Symp. on resilient control systems*, 2014.

[37] KDD Cup Dataset, 1999. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html