Autonomic Parameter Tuning of Anomaly-Based IDSs: an SSH Case Study

Anna Sperotto, Michel Mandjes, Ramin Sadre, Pieter-Tjerk de Boer, and Aiko Pras

***Abstract*—Anomaly-based intrusion detection systems classify network traffic instances by comparing them with a model of the normal network behavior. To be effective, such systems are expected to precisely detect intrusions (high true positive rate) while limiting the number of false alarms (low false positive rate). However, there exists a natural trade-off between detecting all anomalies (at the expense of raising alarms too often), and missing anomalies (but not issuing any false alarms). The parameters of a detection system play a central role in this trade-off, since they determine how responsive the system is to an intrusion attempt. Despite the importance of properly tuning the system parameters, the literature has put little emphasis on the topic, and the task of adjusting such parameters is usually left to the expertise of the system manager or expert IT personnel.**

In this paper, we present an autonomic approach for tuning the parameters of anomaly-based intrusion detection systems in case of SSH traffic. We propose a procedure that aims to automatically tune the system parameters and, by doing so, to optimize the system performance. We validate our approach by testing it on a flow-based probabilistic detection system for the detection of SSH attacks.

***Index Terms*—Autonomic, Network Management, Anomalies, Intrusion Detection, Parameter Optimization**

I. Introduction

Anomaly-based intrusion detection systems (IDSs) aim to classify an activity as benign (normal) or malicious (anomalous) by comparing it with a model of normality. Since the Eighties, many flavors of anomaly-based systems have been developed, depending both on the type of information processed (network traffic or system logs, for example) as well as the type of modeling technique used (among others: statistical or machine learning techniques) [1]. In all cases, however, an anomaly detection system can be more or less sensitive to a change in normality. The sensitivity of a system to anomalies controls which instances of traffic are flagged as anomalous and, in general, it determines the performance of the system. The sensitivity level of an anomaly detection system usually depends on a set of system parameters. Such parameters play a central role, since they determine the overall behavior of the detection system and how it reacts to an intrusion.

If the parameters are chosen such that the system recognizes as benign only activities that closely match the normality model and rejects as suspicious all the others, it is likely that all malicious activities will be reported. However, at the same time, alerts will erroneously be raised also for a high number of benign activities. Similarly, if the system is more permissive with respect to the normality models, it will raise fewer alerts, but it is likely that it will miss some malicious attempts. As we can see, there exists a natural trade-off between detecting all anomalies (at the expense of raising alarms too often), and missing anomalies (but not issuing any false alarms). Such a trade-off is governed by the system parameters. In Figure 1 we have sketched how such a trade-off might look like. The figure provides a visual representation of the fact that the number of missed anomalies and the number of false alarms are inter-related measures. Evidently, the precise shape of the curves depends on the case at hand.

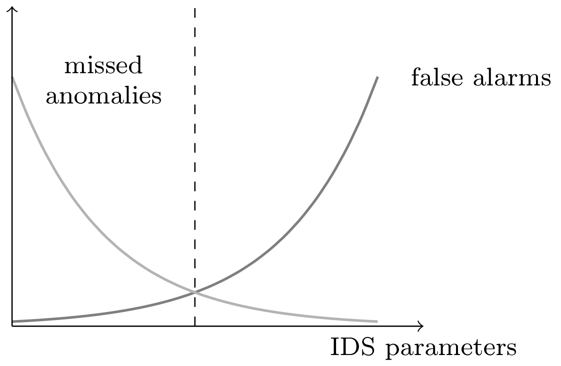


Fig. 1: Trade-off between missed anomalies and false alarms

Despite the deep impact that the system parameters have on the performance of the detection system, the literature has put little emphasis on the topic of autonomic parameter tuning for intrusion detection systems [2]. The task of tuning the system is carried on by the system manager or by expert IT personnel and it requires detailed knowledge of both the detection system as well as the network to be protected. While tuning a system, the system manager implicitly aims to achieve the best compromise, or *optimal solution*, that is keeping the false positives as low as possible while trying to detect as many real attacks as possible.

In this paper, we propose an autonomic approach for the tuning of intrusion detection systems in the case of SSH dictionary attacks. Our approach allows the system performance to be optimized with respect to the aforementioned trade-off. The goal is to allow the system manager to control the system performance by specifying high-level policies. Our approach classifies as autonomic, as it allows the IDS to embed self-

A. Sperotto, R. Sadre, P.T. de Boer and A. Pras are with the Centre for Telematics and Information Technology, University of Twente, The Netherlands (emails: {a.sperotto. r.sadre, p.t.deboer, a.pras}@utwente.nl)

M. Mandjes is with the Korteweg-de Vries Institute, University of Amsterdam, The Netherlands (email:m.r.h.mandjes@uva.nl)

Manuscript received May 12, 2011; revised September 15, 2011 and December 16, 2011.

optimizing operations that rely on an appropriate parameter tuning process. It is the primary objective of our paper to show that, in the SSH setting we consider, such a parameter tuning process can be developed.

The proposed approach can be generalized with respect to the type of anomaly detection system under consideration, as we will discuss in Section VIII. However, in the context of this paper, we focus on a system based on the following assumptions. First, in modern networks we are observing a continuous increase in line speed and load, and throughput of several Gbps is not uncommon anymore. Aggregated network measures, such as *network flows*, became recently an appealing approach to intrusion detection since they help in coping with scalability issues [3], [4]. The data our system analyzes are network flows (NetFlow/IPFIX), that in this context are defined as "a set of IP packets passing an observation point in the network during a certain time interval and having a set of common properties" [5]. Flows also constitute a challenge for intrusion detection systems, since they reduce the amount of information available for analysis. To overcome this problem, we focus on *time series* derived from flow information, i.e., the time series of the number of flows active on the network at the time of the measurement. Flow time series reveal trends in the traffic that can be exploited for detection [2], [6]. In this paper, we focus on *SSH (Secure Shell) traffic*. Our previous studies showed that the SSH service is a regular target of attacks [7], allowing us to collect relevant data sets of malicious and normal traffic. Finally, we apply our approach to a simple detection system, based on *Hidden Markov Models* (HMMs).

This paper is organized as follows. We discuss the related work in Section II. As introduction to our approach, we first present, in general terms, the intrusion detection system we want to optimize in Section III, the type of attack we aim to detect in Section IV, and the performance measures used to validate our findings in Section V. Our optimization procedure, that we present in Section VI, is supported by an extensive validation, presented in Section VII. We then discuss possible extensions of the proposed optimization procedure in Section VIII. Finally, Section IX summaries our findings.

II. Related work

In this section, we provide pointers to related work in the fields of interest linked to this paper. The main conclusion of our brief summary is that until now autonomic tuning procedures received limited attention. We consider this a crucial difference between our work and the existing literature.

As pointed out in Section I, only a few contributions address the topic of parameter tuning for anomaly detection. In [8], the authors explicitly analyze the variation in performance of anomaly-based systems when the system parameters vary, but they do not attempt to automatically tune them. In [9], the authors present a feedback system based on fuzzy logic to control the number of alerts that should be raised to the system operator. Differently from our contribution, this approach relies on the human operator to provide instance-by-instance feedback, in place of specifying policies. The works in [10] and [11] have a different focus compared to our contribution, since they study the dynamical aspects of an intrusion detection system, that is how to adapt the system behavior when a change in normality occurs.

On a methodological level, a substantial body of work focuses on the detection of anomalies in communication networks. Several early papers consider similar issues; without aiming to give an exhaustive overview, we refer to [12], [13], [14]. For the case of Voice-over-IP, guidelines can be found in e.g., [15], but these are of an empirical nature. The main contribution of [16] is developing procedures for anomaly detection that are backed by a mathematical justification. An application of the celebrated Cumulative Sum (CUSUM) technique [17] in the networking domain can be found in [18] and [19]. Several valuable contributions to the changepoint detection problem are due to Tartakovsky and co-authors [20].

It should be noted that there is also a vast literature on statistical techniques for data analysis in networking, from which our work on HMMs originated, as for example in the case of the traffic-classification problem [21], [22], [23], [24], [25]. HMMs are an effective tool to model sequential data [26]. Since they have been introduced in the early 1970s [27], they have been successfully applied to different scientific fields, for instance speech recognition [28], pattern recognition [29] and keystrokes analysis [30]. A few important references on HMMs in the networking context are [31], [22]. However, these papers focus on procedures at packet-level, while our work targets flow-based time series. HMMs are particularly attractive in the context of intrusion detection, as they allow easy computation of the likelihood of a given sequence of events. Behavioral models for host-based intrusion detection have been proposed in [32] and [33]. The authors profile the normal sequence of system calls and raise alarms whenever a sequence is 'sufficiently unlikely'. Contrary to these contributions, we model normal activities based on network data.

This paper focuses on the detection of anomalous SSH traffic, in particular the very common and well-known SSH dictionary attack [34], [35]. Although these attacks have been active for quite some time now, they can still be potentially dangerous. Our study [36] showed that newly set up vulnerable hosts can be compromised within a few days and be used as platform for the same attacks. We also showed that SSH attacks are visible at flow level (as peaks in the SSH flow time series) [37].

III. Detection system principles

The general assumption underlying network anomaly detection is that malicious activities (statistically) deviate from the network's normal behavior [38]. In this section, we present in general terms an anomaly-based system that builds upon the techniques developed in [7]. We then point out the challenges faced when aiming to automatically tune and optimize the system parameters.

Figure 2 presents a high-level overview of the proposed detection system. We assume the system to be based on a probabilistic model of normal behavior, indicated by λ. The system observes a stream of network measurements $\{o_1, o_2, \ldots\}$ and analyzes a window (of length $w \in \mathbb{N}$) of past measurements, to which we refer as *observation sequence*. Observation sequences have been introduced because time series aggregate information and, therefore, in many cases a single time bin is not indicative of the presence of an attack. At time $t \in \mathbb{N}$, our detection procedure evaluates the likelihood of observing the sequence $O_t^w = \{o_{t-w+1}, \ldots, o_t\}$, with respect to our base model λ; we denote this likelihood by $\Phi(O_t^w \mid \lambda)$. Informally, the value $\Phi(O_t^w \mid \lambda)$ is an indication of how likely it is to observe O_t^w according to what we consider normality. Evidently, unlikely sequences will yield relatively low likelihoods, legitimate sequences relatively high likelihoods. The system will then apply a threshold θ to the likelihood value $\Phi(O_t^w \mid \lambda)$ to detect unlikely events.

The challenge is to discriminate between likely and unlikely observation sequences by varying the system parameters w and θ, while at the same time keeping a low classification errors.

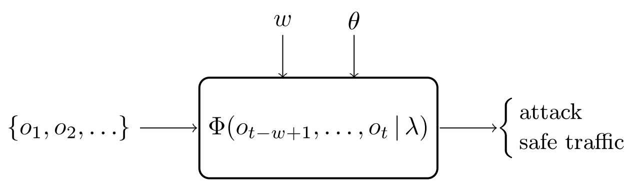


Fig. 2: Detection system.

The selection of the parameters θ and w does play a crucial role in the effectiveness of the detection system. If θ is too low, it is likely that the system would be prone to raise false alerts (false positives, in the sequel denoted by FP), by raising alerts also in case of legitimate observation sequences; if θ is too high, on the other hand, the rate of alerts being generated would decrease, but the system would become less reactive to malicious activities (false negatives; FN). The impact of w is usually model-dependent. Small values of w are likely to prompt the model to give more importance to single observations, possibly causing "too quick" false alerts (FP). Larger values of w may, on the other hand, mitigate the effect of possible unlikely events in the observation sequence, with the risk of missing anomalies (FN).

The values of θ and w define therefore interesting trade-offs that determine the performance of the detection system. Our goal is to present a systematic framework yielding optimal values for the parameters θ and w. It should be realized, though, that this selection mainly depends on the specific goals pursued in the situation at hand: is the primary objective to allow false alerts but to be sure to flag all the anomalies, or vice versa, or perhaps some intermediate scenario? The selection procedure that we propose in the following explicitly takes into account this preference.

IV. SSH CASE STUDY

In the following sections, we will illustrate the performance of the proposed detection system by means of a case study: the detection of SSH dictionary attacks in SSH normal traffic.

As indicated before, we should first define a base model λ describing the normal behavior. Note that, in our case, we consider as normal all the traffic that was not generated by an SSH dictionary attack, the attack on which we concentrate our

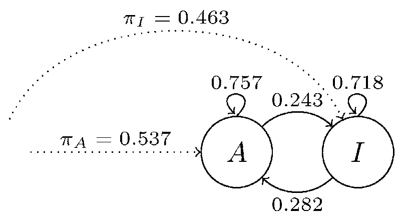


Fig. 3: Normal SSH traffic model.

attention. This means that the base model λ can also include malicious traffic patterns due to attacks different from the SSH dictionary attack, since these attacks are not of interests for this specific demo IDS. For the implementation of our system, we defined the base model λ as an HMM. HMMs are particularly suitable for time-series anomaly detection because they allow a compact representation of the studied system, in this case network traffic, while encoding its evolution in time. Moreover, HMMs offer efficient ways of calculating the probability that a specific sequence of observations would occur (see App. A).

In Figure 3, we present the Discrete-Time Markov Chain (DTMC) for the HMM describing the normal SSH traffic, with an example of initial and transition probabilities. The model was developed and validated in [2]. The DTMC consists of two states: a state A, in which there is SSH traffic on the network (*activity*), and a state I, in which there is no SSH traffic (*inactivity*). The model parameters, as for example the initial and transition probabilities in Figure 3, are estimated by training the model on traces of legitimate SSH traffic collected on the University of Twente (UT) network (see Section VII-A). We let the starting distribution $\pi = [\pi_A, \pi_I]$ be equal to the DTMC's steady state distribution. Each state is augmented with an output probability distribution that in this case is the empirical distribution function of flows per second. Empirical distributions of packets per second and bytes per seconds are also possible. However, flows per second is the discriminating measure between malicious and benign activities for SSH dictionary attack, therefore we concentrate on that distribution.

We indicate with $O_t^w = \{o_{t-w+1}, \ldots, o_t\}$ the observation sequence in the time-frame $t-w+1, \ldots, t$ in our training set. Note that, according to such definition, each observation sequence in the training set is unique, independently from the values o_i that constitute it. In the HMM framework, we refer to the probability of an observation sequence O_t^w as:

$$\begin{aligned}\Phi(O_t^w|\lambda) \quad &= \quad \sum_{\forall Q} \pi_{q_{t-w+1}} b_{q_{t-w+1}}(o_{t-w+1}) \cdot \\ &\quad \cdot \prod_{i=t-w+2}^{t} a_{q_{i-1}q_i} b_{q_i}(o_i),\end{aligned}$$

where $\pi_{q_{t-w+1}} \in \{\pi_A, \pi_I\}$, depending if the first slot of the observation sequence shows SSH traffic or not, and where we assume that the observation sequence O_t^w could have been generated by several state sequences $Q = \{q_{t-w+1}, \ldots, q_t\}$. We indicate, with $a_{q_j q_i}$ the transition probability from state q_j to state q_i, and with $b_{q_i}(o_i)$ the output probability of symbol o_i in state q_i. Since $\Phi(O_t^w|\lambda)$ tends to be small when w grows, it is more convenient to work with

$$\ell_t^w := -\log \Phi(O_t^w|\lambda), \tag{1}$$

where the minus sign allows us to work in $\mathbb{R}^+$.

Since the HMM in Figure 3 is trained on benign SSH traffic, the model is supposed to assign relatively high (low) ℓ_t^w values to sequences of malicious (legitimate) traffic. Figure 4(a) shows the time series of flows per second for an SSH traffic trace measured at the UT on July 16, 2008. Between 7:00 and 7:40, the university network has been the target of an SSH dictionary attack. For the detail of that type of attack, we refer to [7]. In Figure 4(b) we present an example of how ℓ_t^w raises in case of network anomalies, based on the data of Figure 4(a), taking $w = 100$ sec. The window of observations slides over the time series with a step size of 1 second. Such a step size has been chosen since our experiments showed that a 1-second resolution led to a good compromise between data granularity and the risk of observing artifacts due to data aggregation. Figure 4(b) shows that ℓ_t^w attains values roughly between 200 and 400 during the normal phases, whereas during the attack the value of ℓ_t^w abruptly raises to 1000. This graph suggests that ℓ_t^w is a suitable measure for discriminating between normal and anomalous traffic. However, we observe that also other observation sequences provoke a raise of ℓ_t^w, for example around 10 AM, 4 PM, and 10 PM. In these cases, applying a threshold θ of 500, a network anomaly would have been reported even though there was no indication of malicious activity; $\theta = 800$ would be a more appropriate choice here. This underlines that a procedure to soundly select θ is of crucial importance.

V. Attacks, observations and detection

In this section, we will define the performance measures we will later use to evaluate and optimize our prototype detection system. In Section V-A, we describe our system in terms of a binary classifier. This characterization allows us to define the performance metrics we are interested in in Section V-B.

A. Binary classifier

A detection system is traditionally regarded as a *binary classifier* [39]. A *binary classifier* analyzes a set I of instances of a problem and maps these to two *prediction classes*. The prediction classes in the case of intrusion detection are usually indicated as:

- P, or *positive*: $P \subseteq I$, such that $i \in P$ if i has been labeled as an attack;
- N, or *negative*: $N \subseteq I$, such that $i \in N$ if i has been labeled as legitimate.

We assume $P \cap N = \emptyset$ and $P \cup N = I$. The performance of a classifier is evaluated based on how precise the prediction classes are compared to the actual classes of the instances (the ground truth). We indicate the actual classes as:

- M, or *malicious*: $M \subseteq I$, such that $i \in M$ if i is an attack;
- B, or *benign*: $B \subseteq I$, such that $i \in B$ if i is legitimate.

Also in this case we assume $M \cap B = \emptyset$ and $M \cup B = I$.

As described in Section III, we propose an intrusion detection system based on a window of past observations O_t^w. This approach implies that O_t^w is the traffic instance that we aim to classify. We therefore define B, M, P and N as sets of observations sequences.

Given an observation sequence $O_t^w = \{o_{t-w+1}, \ldots, o_t\}_t$, each observation o_i can be either malicious, if an attacker was active at time i, or benign, if no attacker was active at time i. Each observation sequence can therefore contain several malicious observations. Due to the sliding window mechanism, the malicious observation will be added on the right side of the observation sequence and leave the observation sequence on the left side. Since we want to detect an attack as early as possible, we define an attack to begin when the first malicious observation appears in the rightmost (youngest) position in the observation sequence; similarly, an attack ends when a benign observation appears in the rightmost position in the observation sequence. In the following, we will define the sets B, M, P and N for the case under consideration. Please note that such sets will now depend on the observation sequence length w. We therefore stress this dependency by introducing the superscript B^w, M^w, P^w and N^w. Following the reasoning introduced above, we derive the following definitions:

$$M^w = \{O_t^w \mid o_t \text{ is malicious}\}, \tag{2}$$
$$B^w = \{O_t^w \mid o_t \text{ is benign}\}. \tag{3}$$

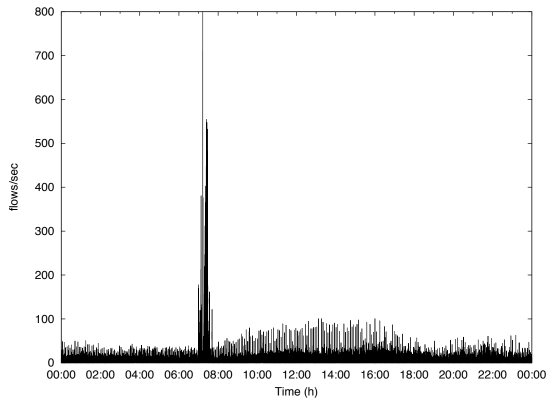
With this definition of M^w and B^w, it is important to note that the discriminating condition between malicious and benign observation sequences is given by the youngest observation. The consequence of this is that an observation sequence will be considered malicious (benign) even if all but the last observation are benign (malicious). This reflects the situation in which an observation sequence is labeled as malicious as soon as an attack becomes active (i.e., it appears in the youngest slot of an observation sequence). Similarly, once an attack is over, the youngest slot of an observation sequence is benign and we consider the entire observation sequence as benign. In other words, we can say that our definitions focus on the classification of the current situation, and use the history for supporting the classification. Note, however, that other definitions of M^w and B^w are possible.

Finally, the prediction classes are such that an observation sequence is flagged as positive if it deviates from the base model of traffic λ. We therefore define:

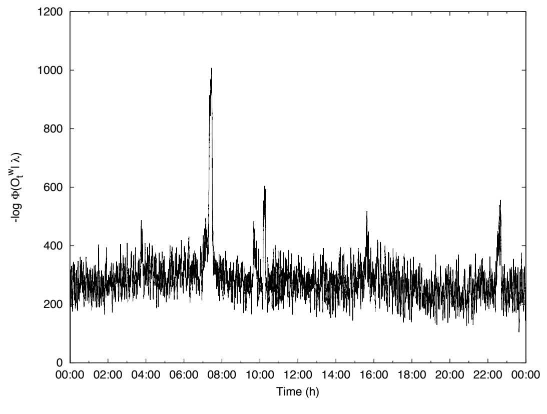
$$P^w = \{O_t^w \mid \ell_t^w > \theta\}, \tag{4}$$
$$N^w = \{O_t^w \mid \ell_t^w \leq \theta\}. \tag{5}$$

B. Performance measures

This section introduces the performance measures of interest for our study. We propose three groups of performance metrics. The first two, the *confusion matrix* and the *detection rate* are traditionally used in evaluating IDSs [40]. The third group, the *detection* and *normalization lags*, has been introduced in this paper to quantify the delays the detection system incurs in detecting an attack and in recognizing the re-establishment of a normal situation.



(a)



(b)

Fig. 4: Time series for a day of SSH traffic (flows/second) (a) and log-likelihood of the time series (b).

1) Confusion matrix: The *confusion matrix* [39] describes both the corrected classified instances and the *errors* between the classes. Based on the confusion matrix, the following performance metrics can be defined:

- the *true positive* TP $= \#\{P \cap M\}$; and the *false negative* FN $= \#\{N \cap M\}$;
- the *true negative* TN $= \#\{N \cap B\}$; and the *false positive* FP $= \#\{P \cap B\}$.

The measures above are often expressed in terms of *rates*, that is TP-rate $=$ TP$/\#\{M\}$, FP-rate $=$ FP$/\#\{B\}$, and so on. In the sequel, we will use the notation TN, TP, FN, FP as rates, such that TN $+$ FP $=$ TP $+$ FN $= 1$.

2) Detection rate: Traditionally, intrusion detection evaluates the system performance in terms of TN, TP, FN, FP, based on the sets M, B, P and N. In our case, the elements of such sets are observation sequences O_t^w. However, it should be noted that an attack could cover more than one observation sequence. We define an attack $\mathbf{A}$ to be detected if at least one observation sequence part of the attack is flagged as positive. Formally, $\mathbf{A}$ is detected if there exists at least one observation sequence O_t^w, $t_s \leq t \leq t_e$ such that $O_t^w \in P \cap M$, where t_s and t_e indicate the starting and ending time of an attack, respectively.

Besides the rate of correctly classified observation sequences, we can therefore also measure, for a set of attacks in our data sets, how many attacks have been detected. We define this performance measure as the *detection rate* DR per attack:

$$DR = \frac{\#\{\text{detected attacks}\}}{\#\{\text{attacks}\}}.$$

3) Detection and normalization lags: A detection system aims to identify the presence of an attack. A likely situation is that classification errors would be observed at the beginning and at the end of the attack. While the traffic is entering the observation window, some observation sequences will be classified as negative, since only few observations are malicious. Conversely, while the window is leaving the attack, some observation sequences will be flagged as positive, since only their youngest observations are benign. It is highly preferable that an attack is detected fast. Similarly, it is desirable for a detection system to also recognize as fast as possible that an anomalous situation is over.

Since in our setup we rely on a sliding window mechanism with steps of 1 second, we can measure how promptly the system reacts by analyzing the classification errors at the beginning and at the end of an attack. It is therefore desirable to have a low rate of false negatives in the beginning of an attack and a low rate of false positives when an attack is over.

Given an attack $\mathbf{A}$, and its starting and ending time, t_s and t_e, we then define the following measures:

- the *detection lag*, i.e., the delay, in our case measured in seconds, before an attack is detected:
$$D_{\mathbf{A}}^w := \min\{t \geq t_s |\ O_t^w \in M \cap P\} - t_s;$$
- the *normalization lag*, i.e., the delay, in seconds, in recognizing that an attack is over. Since we aim to investigate the impact of the history on the analysis of normal traffic, we measure the normalization lag within w observations from the end of the attack:
$$E_{\mathbf{A}}^w := \min\{t_e \leq t < t_e + w |\ O_t^w \in B \cap N\} - t_e.$$

Finally, we can calculate the averages, $\overline{D_{\mathbf{A}}^w}$ and $\overline{E_{\mathbf{A}}^w}$, over the attacks present in our data sets, of the detection and normalization lags.

VI. The optimization procedure

In Section I, we argued that tuning an IDS entails searching for an optimal solution w.r.t. the trade-off between detecting as many attack as possible and having a low rate of false alarms. More generally, let us consider an IDS with parameters $p_1, \ldots, p_n$ and a high-level policy $\mathcal{P}$ describing the required behavior of the IDS in terms of some performance metrics $\mathcal{M}_1, \ldots, \mathcal{M}_m$, such as for example TP and TN. To automatically tune the system parameters to satisfy $\mathcal{P}$, the following steps should be taken. First, it is necessary to define a mathematical relation $Goal$ of the performance metric that we aim to optimize, that keeps into account the policy $\mathcal{P}$:

$$\underset{p_1, \ldots, p_n}{\text{maximize}} \quad \text{Goal}(\mathcal{M}_1, \ldots, \mathcal{M}_m, \mathcal{P}).$$

TABLE I: The optimization procedure.

Optimization procedure

Inputs:
IDS parameters θ, w
Performance metrics TN, TP
Policy (α, β)

step 1:

$$\underset{\theta, w}{\text{maximize}} \quad \alpha \cdot \text{TN} + \beta \cdot \text{TP}$$

step 2: relation between TN, TP and θ:

$$\text{TN} = \mathbb{P}(X^w \leq \theta), \text{TP} = 1 - \mathbb{P}(Y^w \leq \theta)$$

$$X^w, Y^w \overset{\text{d}}{=} \mathbb{G}\text{m}(\boldsymbol{p}, \boldsymbol{\mu}, \boldsymbol{\sigma})$$

step 3: solve Problem (6)

This relation describes the goal, such as for example maximizing the correct classification. However, in the present formulation, the optimization problem requires that we are able to measure the performance metrics $\mathcal{M}_1, \ldots, \mathcal{M}_m$ for varying parameters. This operation can be computationally intensive. To reduce the complexity of this step, we propose an intermediate step where we analytically express the relation between the performance measures and the system parameters. Once we have expressed a mathematical relation between performance measures and the system parameters, the last step is to analytically solve the optimization problem. Table I shows how the aforementioned steps have been implemented in the context of this paper.

A. The optimization problem

This section focuses on formalizing the parameter tuning as an optimization problem (step 1 of the procedure in Table I). Our aim is to determine suitable values for the parameters w and θ.

The performance of an IDS is usually presented by means of a so-called Receiver Operator Characteristic (ROC) [39]. The ROC curve plots the combinations of FP and TP, by varying θ. ROC curves are a technique for visualizing the performances of a classifier, since they show the "relative trade-offs between benefits (TP) and costs (FP)" [39]. Intuitively, the steeper the ROC curve grows, the better is the performance of the classifier.

Figure 5 shows examples of ROC curves obtained by testing the HMM-based detection system on a synthetic data set of SSH traffic for different window sizes. As mentioned above, each point on the ROC curves is determined by a different value of the threshold θ. The curves show a steep growth of TP for low values of FP, while the curves grow slower for higher values of FP. The ROC curves indicate that the proposed detection system is able to discriminate between malicious and benign observation sequences with high TP and low FP. However, Figure 5 also shows that:

- The ROC curves growth depends on the observation sequence length w; we plotted curves for three values w. They indicate that the performance of the procedure can be improved by choosing w optimally. For instance, as shown in the subplot of Figure 5, to obtain TP $= 0.98$, $w = 100$ would provide the lowest FP. Moreover, the optimal value of w will change with the desired TP and FP.
- Since θ controls in which point of a ROC curve we operate, the choice of the threshold value θ depends on the desired performance. For example, for $w = 100$ we can obtain TP $= 0.8$ with FP $= 0.02$, but also TP $= 0.93$ with FP $= 0.1$.

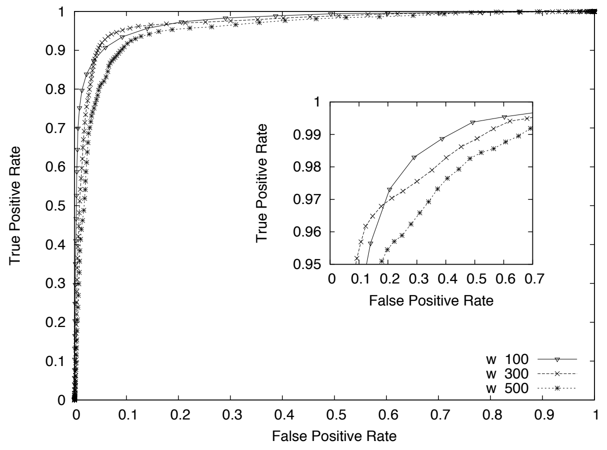


Fig. 5: Receiver Operating Characteristics, for varying window size.

The above reasoning indicates that one can optimize the parameters w and θ, and the optimum critically depends on the relative importance the system manager assigns to TP and FP. Making use of the relation TN $= 1 -$ FP, we formulate the optimization problem as follows:

$$\underset{\theta, w}{\text{maximize}} \quad \alpha \cdot \text{TN} + \beta \cdot \text{TP}. \tag{6}$$

Here the parameters α and β embody the relative importance the system manager assigns to TN and TP. If he chooses α substantially larger than β, the optimization would lead to values of (w, θ) that make the system behave conservatively, flagging anomalies only if there are strong evidences of an attack, at the expense of relatively many false negatives. On the other hand, if he picks α considerably smaller than β the system would become more aware of malicious traffic, favoring in this way TP, but by increasing FP.

B. Probabilistic interpretation of the classification problem

We concentrate now on formalizing the relation between the system parameters and the performance metrics (step 2 of the procedure in Table I). We define the following random variables: for an arbitrary observation sequence of length w,

$$X^w := -\log \Phi(O_t^w | \lambda) \quad \text{for } O_t^w \in B^w$$

for a benign observation sequence, and

$$Y^w := -\log \Phi(O_t^w | \lambda) \quad \text{for } O_t^w \in M^w$$

for a malicious observation sequence. For any given w, we can estimate the density functions $f_X^w(\cdot)$ and $f_Y^w(\cdot)$ in the form of

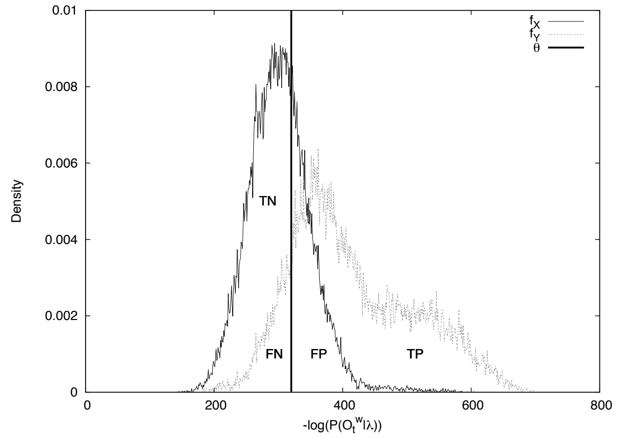


Fig. 6: Performance measures and empirical distribution functions.

normalized frequency histograms obtained from training data. An example of these are shown in Figure 6 for $w = 100$, based on the analysis of normal and malicious traffic. In the figure we also show an example value for θ, which splits the sample space in regions. The subspace to the left of the threshold corresponds to negatively-flagged inputs; in the same way, the subspace to the right of the threshold corresponds to positively-flagged inputs. The intersection between such subspaces and the probability distributions results in regions that we can map directly on the confusion matrix of our binary classifier. In particular, according to this formulation, we have

$$\text{TN} = \mathbb{P}(X^w \leq \theta), \quad \text{FP} = 1 - \mathbb{P}(X^w \leq \theta), \tag{7}$$

$$\text{FN} = \mathbb{P}(Y^w \leq \theta), \quad \text{TP} = 1 - \mathbb{P}(Y^w \leq \theta). \tag{8}$$

We can now rewrite Problem (6) as follows:

$$\underset{\theta,w}{\text{maximize}} \quad \alpha \cdot \mathbb{P}(X^w \leq \theta) + \beta \cdot (1 - \mathbb{P}(Y^w \leq \theta)). \tag{9}$$

Figure 6 also shows that the distributions of X^w and Y^w overlap, and hence θ cannot be chosen such that it perfectly discriminates between malicious and benign. The presence of an overlapping region, and its size, can change from data set to data set, and this imposes a sort of theoretical limit to the classification performance of the detection procedure for given λ and w.

In (9), TN and TP are expressed in terms of the cumulative distribution functions of X^w and Y^w. These cumulative distribution functions can be determined empirically, in the same way as we did for the densities in Figure 6. However, since our goal is to perform the optimization in (9), which is over θ and w, we need the distribution functions $\mathbb{P}(X^w \leq \cdot)$ and $\mathbb{P}(Y^w \leq \cdot)$ for a broad range of values of w. Evidently, estimating these for many and especially for large values of w will make the optimization procedure slow: the complexity of computing ℓ_t^w is linear in w, and such measure is computed for each observation sequence, i.e., each time the window slides over the time series. Therefore, especially for a large amount of training data, such a procedure is costly. To reduce the complexity, we proceed as described in the following. Our analysis of the empirical density function for X^w and Y^w for variable values of w shows that a *Gaussian mixture distribution* (denoted by $\mathbb{Gm}$) offers a suitable fit for the considered distributions. Following the convention to write vectors in bold, we say that a random variable Z is $\mathbb{Gm}(\boldsymbol{p}, \boldsymbol{\mu}, \boldsymbol{\sigma})$ of order $n \in \mathbb{N}$, with $p_i \geq 0$ adding up to 1, if its probability density function is in the form

$$f_Z(x) = \sum_{i=1}^{n} p_i f_{N_i}(x),$$

where the components $N_i = \mathcal{N}(\mu_i, \sigma_i)$, for $i = 1, \ldots, n$, are independent normally distributed random variables. To add flexibility and decrease the cost of estimating the Gaussian parameter for arbitrary values of w, we follow the next steps:

1) for a small set of window sizes $w = w_1, \ldots, w_n$, we fit Gaussian mixtures to the empirical distribution densities f_X^w and f_Y^w; we obtain the set of Gaussian mixture parameters $\boldsymbol{p}_X^w, \boldsymbol{\mu}_X^w$ and $\boldsymbol{\sigma}_X^w$ and $\boldsymbol{p}_Y^w, \boldsymbol{\mu}_Y^w$ and $\boldsymbol{\sigma}_Y^w$;
2) we express the fitted Gaussian parameters as explicit functions of w:
$$p_{iX^w} = g_1(w), \quad \mu_{iX^w} = g_2(w), \text{ and } \sigma_{iX^w} = g_3(w),$$
where for example g_i can be a fitted polynomial function;
3) finally, we apply the explicit functions g_i to extrapolate the parameters values for arbitrary values of w.

Figure 7 shows examples of polynomial fits for the Gaussian parameters of data set *Synthetic 2*.

To conclude our procedure, step 3 can be completed by optimizing (9). Since the empirical density function for X^w and Y^w are fitted with Gaussian mixture distributions, (9) can also be rewritten as the sum of the cumulative distribution functions of the Gaussian component, i.e., as a sum of cumulative distribution functions of normal distributions. We indicate the cumulative distribution function of $\mathcal{N}_i(\mu_i, \sigma_i)$ with

$$F(\theta, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\theta} e^{-(x-\mu)^2/2\sigma^2} dx. \tag{10}$$

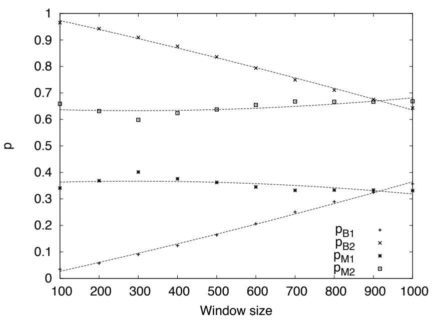
Then, (9) can be rewritten as

$$\begin{aligned} \underset{\theta,w}{\text{maximize}} \quad & \alpha \cdot \sum_{i=1}^{n} p_{iX^w} F(\theta, \mu_{iX^w}, \sigma_{iX^w}) \\ & + \beta \cdot (1 - \sum_{i=1}^{m} p_{iY^w} F(\theta, \mu_{iY^w}, \sigma_{iY^w})), \end{aligned} \tag{11}$$

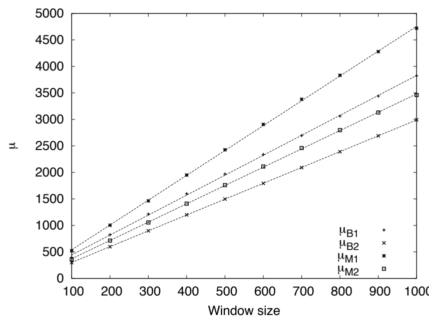
where n and m are the number of Gaussian components for X^w and Y^w, respectively, p_{iX^w}, μ_{iX^w} and σ_{iX^w} the parameters of the Gaussian Mixture for X^w, and p_{iY^w}, μ_{iY^w} and σ_{iY^w} the parameters of the Gaussian Mixture for Y^w. However, the integral in (10) has no analytical solution. Therefore, (11) has to be numerically solved, for example by identifying the zeros of the derivative of the function in Eq. (11).

VII. Validation and Experimentation

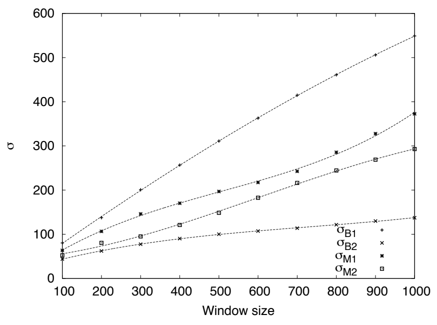
The aim of this section is twofold. First we present the data sets used for testing in Section VII-A, and we describe the impact of the analytical procedure compared to results obtained with empirical data in Section VII-B. Then, we show how varying α and β affects the performance of the system (Section VII-C).



(a) p_{iX^w} and p_{iY^w}



(b) μ_{iX^w} and μ_{iY^w}



(c) σ_{iX^w} and σ_{iY^w}

Fig. 7: Polynomial fits for the Gaussian Mixture parameters for the data set *Synthetic 2*

A. *Data sets*

We tested our procedure using six data sets of SSH time series of flows per seconds, each containing both legitimate and malicious traffic. Two data sets, *Synthetic 1* and *Synthetic 2*, are synthetically generated as described in [7], [2]. The other four data sets, *Original 1*, *Original 2*, *Set 1* and *Set 2* consist instead of time series created directly from real traffic collected at the UT network.

The synthetic data sets *Synthetic 1* and *Synthetic 2* consist of 20 time series of 5400 seconds (time slots) each, each one of them containing one attack. The data sets *Original 1* and *Original 2* have been created from traces captured on 17-20 July 2008 and 23-26 April 2009, respectively. Data set *Original 1* contains 7 attacks, while data set *Original 2* contains 6 attacks. Attacks have been identified based on the reports of a quarantine system deployed at UT, and have been manually labeled. Finally, *Set 1* and *Set 2* have been captured at the UT network on 13-16 April 2008 and 19-22 July 2009, respectively, and they have been manually labeled with the same procedure used for *Original 1* and *Original 2*.

In the following, we used *Set 1* and *Set 2* for the training of the HMM λ, while the other data sets have been used for testing. Figure 3 shows the initial and transition probabilities obtained by training the model on *Set 1*. The HMM λ has been trained on *Set 1* for testing *Synthetic 1* and *Original 1*, and on *Set 2* for testing *Synthetic 2* and *Original 2*.

In Section VI, we proposed Gaussian Mixture distributions as suitable fits for the empirical density functions X^w and Y^w. However, the number of components of the Gaussian mixtures can be data set specific. In our setup, we observed that:

- $n = 2$ yields an excellent fit for the data sets *Synthetic 1* and *Synthetic 2*. We therefore define:
$$X^w \overset{\text{d}}{=} \mathbb{G}\text{m}(\boldsymbol{p}^w_B, \boldsymbol{\mu}^w_B, \boldsymbol{\sigma}^w_B)$$
and
$$Y^w \overset{\text{d}}{=} \mathbb{G}\text{m}(\boldsymbol{p}^w_M, \boldsymbol{\mu}^w_M, \boldsymbol{\sigma}^w_M);$$
- for the data sets *Original 1* and *Original 2*, we choose $n = 1$ for X^w and $n = 2$ for Y^w:
$$X^w \overset{\text{d}}{=} \mathcal{N}(\mu^w_B, \sigma^w_B)$$
and
$$Y^w \overset{\text{d}}{=} \mathbb{G}\text{m}(\boldsymbol{p}^w_M, \boldsymbol{\mu}^w_M, \boldsymbol{\sigma}^w_M).$$

Figure 8 presents an example of the fits for the benign and malicious distributions in *Synthetic 2*, obtained for $w = 300$. Finally, the relation between the $\mathbb{G}\text{m}$ parameters $\boldsymbol{p}^w$, $\boldsymbol{\mu}^w$ and $\boldsymbol{\sigma}^w$ and the window size w, for both the benign and malicious distributions, has been expressed by means of polynomial fits. Our findings show that $\boldsymbol{\mu}^w$ is a linear function of w, $\boldsymbol{\sigma}^w$ can be approximated by a polynomial of degree 3 and $\boldsymbol{p}^w$ by a polynomial of degree 2 (see Figure 7).

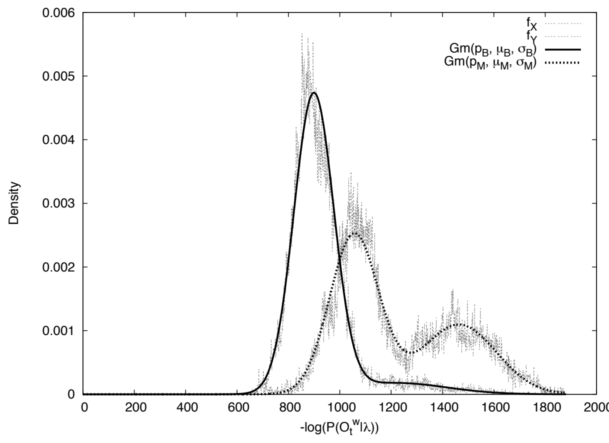


Fig. 8: Example of distribution fits, $w = 300$.

B. *Optimization procedure validation*

In Section VI, we proposed an optimization procedure built upon Gaussian Mixture distributions. In this section, we validate such a procedure, by measuring how precisely we can approximate the empirical optimal results. We therefore compare the results of the optimization problem (i) calculated on empirical data and (ii) analytically calculated by means of Gaussian fits. For this test, we set $\alpha = \beta = 1$. The empirical data are obtained by directly measuring TN and TP on the empirical distribution of ℓ^w_t, similar to the one shown in Figure 6.

Figures 9(a), 9(b) and 9(c) present the relative errors between the empirical and analytical values for the performance measures TP and TN and the threshold θ, for the data sets *Synthetic 1*, *Synthetic 2*, *Original 1* and *Original 2*. The figures show the relative error between the empirical values and the values obtained through our procedure by means of Gaussian fits, expressed in %.

The results in Figures 9(a), 9(b) and 9(c) show that the approach, based on fitted $\mathbb{G}\mathrm{m}$ distributions, very well approximates the results one would have found using the empirical distributions. For the data set *Synthetic 1*, the relative error for TN, TP and θ is for all the considered cases lower than 5%, and such error is lower than 4% for *Synthetic 2*. For *Original 1* we see that the relative error is generally lower or close to 10%. For the data set *Original 2*, on the other hand, we observe a higher error in TP and TN, in certain cases up to 32% for TN. We identify the cause for such higher errors in the polynomial fits used to approximate the parameters of X^w and Y^w as functions of w. In this case, the polynomial fitting does not yield to the same excellent fits as for the other data sets [2]. The error in approximating θ remains, however, in any case lower than 6%.

Our findings confirm that the Gaussian mixtures are a suitable fit for the malicious and benign ℓ_t^w distributions and they can be used to compute the optimal parameter θ.

C. Performance measures

The aim of this section is to study the impact of the parameters α and β on the performance of our simple detection system. We therefore report here the performance measure for varying ratio β/α. Note that it is not needed to know the absolute values of α and β, as only their ratio affects the outcome of the optimization procedure. As explained in Section V-B, the performance is expressed in terms of: (i) the confusion matrix and (ii) the detection rate, as commonly done in intrusion detection; (iii) the detection and normalization lags in recognizing the presence of an attack or the re-establishment of a normal situation.

1) Confusion matrix: Tables II, III, IV and V present the confusion matrix and the detection and normalization lags, as achieved by the system for the data sets *Synthetic 1*, *Synthetic 2*, *Original 1* and *Original 2*, respectively. The rows in the tables are obtained as follows. First, for a given β/α, we compute the optimal w and θ, based on the learned HMM λ and the Gaussian fits for the considered data set. Then the data set is analyzed by calculating the test statistic ℓ_t^w for each observation sequence. An observation sequence is then classified as positive or negative. Finally, TP and TN are calculated. Note that the tables also include results for the detection and normalization lags. We will discuss such results later in this section. We first focus on the effect of varying α/β on the metrics TP and TN. Considering the results:

- we observe that the optimal choice of the design parameters w and θ is data set specific. This observation entails that w and θ have to be determined for the network under consideration; as could be expected, there are no universally suitable values;
- we verified that the optimal choice of the design parameters w and θ is (β/α)-specific. This means that, when increasing β/α, the system slowly shifts from favoring TN to favoring TP, as desired. This is visible by observing that for increasing values of β/α, TN shows a decreasing trend, while TP progressively increases.

Comparing the synthetic and original data sets, we observe that the measured detection rates are sometimes lower than expected. In particular, we observe that there is a decrease in performance for the original data sets with respect to the synthetic ones. More specifically, TP in the original data set shows just a mildly increasing trend, whereas TN decreases considerably faster than in the synthetic case. The explanation for this phenomenon is to be found in the distributions of malicious and benign observation sequences, which, as we pointed out in Section VI-B, determine the limitations to the system performance. In the case of original traces, we observe, compared to the synthetic case, a larger overlap of the benign and malicious distributions, i.e., those of X^w and Y^w. We give an example of such distributions in Figure 10. The distributions in Figure 10 are relative to the data set *Original 2* and calculated for $w = 500$. The consequence of such overlapping distributions is that, despite the fact that the chosen parameters are optimal, no better detection rates are possible in the current setup.

Several causes are behind such a limited performance. First, the definitions of M^w and B^w in Section V-A are such that partially benign and partially malicious observation sequences would lead to similar ℓ_t^w values. We address this topic in Section VIII. Second, the prototype used for our analysis, presented in Section III, is very simple. Such a prototype was a good starting point for describing our optimization procedure, but several extensions would be needed before it could be used as an IDS. Finally, such results provide us with indirect information about the quality of the synthetic time series. In [7], [2], we showed that our models are able to capture the main statistical characteristics of the original data sets, with the exception of the autocorrelation. We suspect that the higher random component in the synthetic time series causes the smaller overlap of the considered distributions. Improving the autocorrelation of the synthetic traces is therefore a key topic for future extensions.

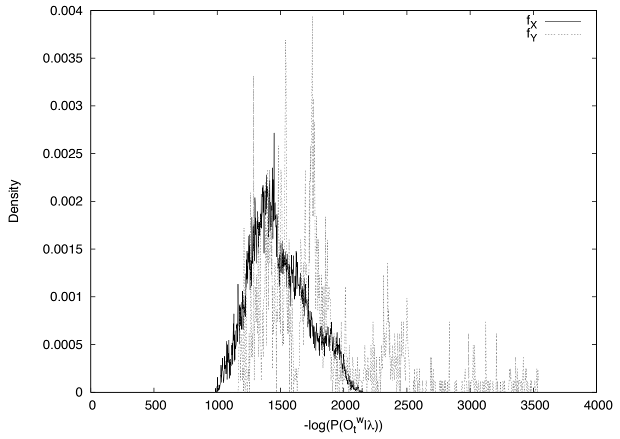


Fig. 10: Distribution for data set *Original 2*, $w = 500$.

2) Detection and normalization lags: Tables II, III, IV and V also report the average detection lag $\overline{D}_{\mathbf{A}}$ and the average normalization lag $\overline{E}_{\mathbf{A}}$ and their respective standard deviations. In calculating the detection and normalization lags,

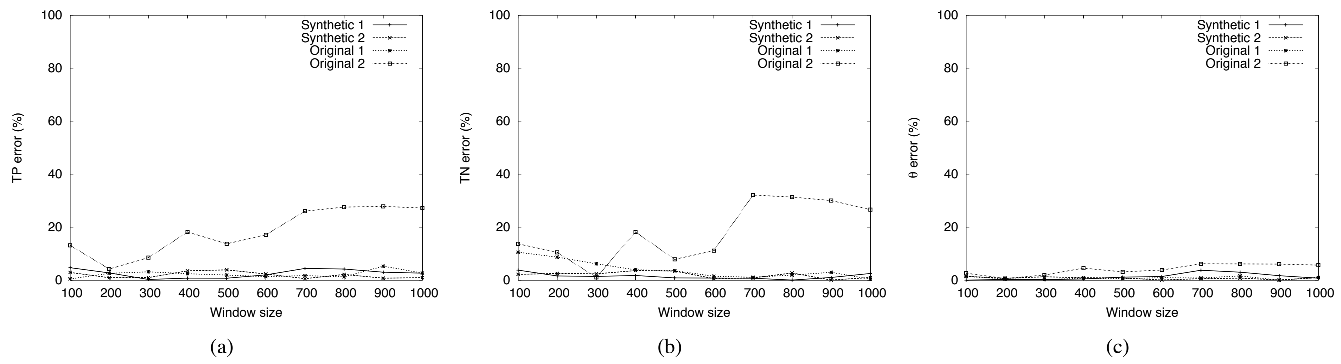


Fig. 9: Relative errors between the empirical and analytical values of TP, TN and θ as results of the optimization problem for the data sets *Synthetic 1*, *Synthetic 2*, *Original 1* and *Original 2*

TABLE II: Performance measures for *Synthetic 1*.

			Confusion matrix		**Detection and normalization lags**			
β/α	w_{opt}	θ_{opt}	TP	TN	$\overline{D}_{\mathbf{A}}$	$std(D_{\mathbf{A}})$	$\overline{E}_{\mathbf{A}}$	$std(E_{\mathbf{A}})$
0.1	110	381.59	0.82	0.99	61.85	38.60	32.42	24.26
0.2	120	404.24	0.87	0.98	53.75	21.76	43.21	25.54
0.3	130	430.12	0.89	0.97	53.50	19.57	52.95	28.89
0.4	130	426.11	0.90	0.97	52.25	19.36	56.37	28.62
0.5	130	423.07	0.91	0.97	51.55	19.65	59.47	28.92
0.6	140	451.3	0.92	0.96	51.45	21.26	68.37	29.93
0.7	140	449.13	0.92	0.96	50.80	21.06	69.11	30.07
0.8	140	447.26	0.92	0.96	50.45	21.01	71.16	30.01
0.9	140	445.63	0.92	0.96	46.95	19.92	72.37	28.52
1	140	444.16	0.92	0.95	46.75	19.83	74.11	28.14
1.5	140	438.54	0.93	0.94	42.45	21.73	76.95	27.09
2	150	464.26	0.94	0.93	43.55	22.67	82.22	28.94
4	150	454.01	0.96	0.90	36.35	22.38	89.82	29.23

TABLE III: Performance measures for *Synthetic 2*.

			Confusion matrix		**Detection and normalization lags**			
β/α	w_{opt}	θ_{opt}	TP	TN	$\overline{D}_{\mathbf{A}}$	$std(D_{\mathbf{A}})$	$\overline{E}_{\mathbf{A}}$	$std(E_{\mathbf{A}})$
0.1	100	418.39	0.41	0.98	258.15	176.79	33.44	23.84
0.2	100	397.30	0.49	0.97	188.25	161.70	40.94	25.45
0.3	100	386.29	0.53	0.96	153.00	144.85	44.06	24.63
0.4	100	378.47	0.57	0.94	115.00	132.34	46.39	25.02
0.5	220	765.59	0.74	0.89	153.65	135.76	153.35	29.43
0.6	250	856.51	0.78	0.87	130.60	139.30	180.25	29.58
0.7	270	915.30	0.81	0.85	140.40	139.89	201.13	31.69
0.8	290	974.55	0.83	0.84	128.25	143.27	222.19	32.73
0.9	300	1001.75	0.85	0.83	129.20	147.28	233.75	33.22
1	320	1061.59	0.85	0.82	118.30	123.74	252.47	34.13
1.5	340	1106.62	0.88	0.77	98.90	119.67	275.93	31.01
2	350	1124.13	0.91	0.74	61.15	75.72	292.00	32.32
4	340	1055.90	0.95	0.62	38.80	72.76	297.85	32.94

we adhered to the following conventions: $D_{\mathbf{A}} = \infty$ if an attack is undetected; $E_{\mathbf{A}} = \infty$ if the system does not recognize the normal situation within a w slots from the end of the attack. This choice was due to the fact that we are interested in the effect the history has in recognizing that an attack is over. $\overline{D}_{\mathbf{A}}$, $\overline{E}_{\mathbf{A}}$ and their respective standard deviations have been calculated only on the finite values of $D_{\mathbf{A}}$ and $E_{\mathbf{A}}$.

In all considered data sets, we observe that the detection lags decrease for increasing values of β/α, that is, if the system favors TP. However, at the same time we also observe that, for increasing β/α, the normalization lag tends to increase. This situation describes well how varying the relative importance of TN and TP does affect not only the confusion matrix, but indirectly also how timely we are able to detect an attack or recover from it. Note that, in Table IV, for $\beta/\alpha < 0.4$, the standard deviation of both the detection and the normalization lags is undefined. This is due to the fact that, for this β/α ratio, the system detects only one attack. In Table V, we observe that $\overline{E}_{\mathbf{A}}$ and $std(E_{\mathbf{A}})$ can be undefined or ∞. For $\beta/\alpha < 0.6$, i.e., for ratios that favor the TN rate, the normalization lag is equal to 0. This value, together with the low TP rate, indicates that the tail of the attack is generally missed and flagged as

TABLE IV: Performance measures for *Original 1*.

			Confusion matrix		**Detection and normalization lags**			
β/α	w_{opt}	θ_{opt}	TP	TN	$\overline{D}_{\mathbf{A}}$	$std(D_{\mathbf{A}})$	$\overline{E}_{\mathbf{A}}$	$std(E_{\mathbf{A}})$
0.1	940	3555.26	0.06	1.00	762.00	-	56.00	-
0.2	1000	3651.20	0.06	0.99	755.00	-	149.00	-
0.3	1000	3577.90	0.06	0.99	733.00	-	182.00	-
0.4	330	1245.08	0.10	0.99	1057.00	1461.53	60.75	82.21
0.5	1000	2817.50	0.22	0.75	774.00	1379.88	560.00	409.26
0.6	1000	2808.64	0.22	0.74	773.20	1379.66	561.00	409.68
0.7	1000	2801.65	0.23	0.74	772.00	1379.05	561.75	409.84
0.8	1000	2795.88	0.23	0.74	771.40	1378.82	563.00	409.94
0.9	1000	2790.95	0.23	0.73	770.80	1378.49	565.75	410.67
1	1000	2786.64	0.23	0.73	770.60	1378.60	569.50	410.52
1.5	1000	2770.58	0.26	0.72	766.40	1380.26	573.25	411.75
2	1000	2759.21	0.28	0.72	746.20	1391.33	575.50	412.86
4	1000	2727.14	0.32	0.70	735.20	1394.97	590.75	417.67

TABLE V: Performance measures for *Original 2*.

			Confusion matrix		**Detection and normalization lags**			
β/α	w_{opt}	θ_{opt}	TP	TN	$\overline{D}_{\mathbf{A}}$	$std(D_{\mathbf{A}})$	$\overline{E}_{\mathbf{A}}$	$std(E_{\mathbf{A}})$
0.1	1000	3923.12	0.22	0.99	640.80	523.94	0	-
0.2	1000	3854.63	0.23	0.98	623.20	531.67	0	-
0.3	1000	3803.11	0.23	0.97	607.00	530.02	0	-
0.4	1000	3752.98	0.24	0.95	597.60	526.69	0	-
0.5	1000	3691.12	0.24	0.93	583.80	526.55	0	-
0.6	1000	3565.83	0.29	0.88	565.80	516.73	0	-
0.7	1000	3351.39	0.45	0.76	467.50	464.88	178.00	398.02
0.8	1000	3247.08	0.63	0.71	442.17	454.67	208.20	408.37
0.9	1000	3179.27	0.70	0.67	416.50	436.71	29.75	59.50
1	1000	3128.34	0.74	0.62	395.67	425.35	33.25	66.50
1.5	1000	2975.06	0.90	0.46	310.17	383.29	146.50	179.09
2	1000	2888.05	0.93	0.36	250.83	349.33	140.67	243.64
4	310	795.16	0.98	0.12	28.00	68.10	∞	∞

negative. For $\beta/\alpha = 4$, i.e., for a ratio that clearly favors the TP rate, the normalization lag rises to ∞. To understand this result it is necessary to keep in mind that β/α describes the relative importance of TN and TP. For such a high β/α ratio, the trace was almost entirely flagged as positive. Therefore, normality was never recognized within w slots after the end of the attack, leading to $\overline{E_{\mathbf{A}}} = \infty$.

3) Detection rates: We investigated the *detection rate* per attack. We conducted our analysis for varying β/α and considering the optimal threshold and observation sequence length, as presented in Tables II, III, IV and V. Table VI presents the detection rate for the four considered data sets. In the case of the synthetic data sets *Synthetic 1* and *Synthetic 2*, the system correctly detects all the attacks, showing $DR = 100\%$. This is due to the fact that the likelihood distributions of benign and malicious observation sequence are fairly separated, therefore the optimal threshold always succeeds in detecting at least part of an attack. In the case of the original data sets *Original 1* and *Original 2*, we observe that the detection rate was not constant, but changed with varying ratio β/α. As expected, the detection rate increases for increasing β/α, i.e., when the system favors the TP rate.

TABLE VI: Detection rates.

	DR (%)			
β/α	*Synthetic 1*	*Synthetic 2*	*Original 1*	*Original 2*
0.1	100	100	14	83
0.2	100	100	14	83
0.3	100	100	14	83
0.4	100	100	57	83
0.5	100	100	71	83
0.6	100	100	71	83
0.7	100	100	71	100
0.8	100	100	71	100
0.9	100	100	71	100
1.0	100	100	71	100
1.5	100	100	71	100
2.0	100	100	71	100
4.0	100	100	71	100

VIII. Discussion

This section discusses the applicability of the proposed optimization procedure and the impact of using a binary classifier as classification and evaluation approach underlying our system.

A. Applicability of the proposed approach

In the introduction to Section VI, we have described an optimization procedure that aims at relating the system parameters to the metrics of interests in terms of an optimization problem. In that form, the optimization procedure is generic and its principles can be applied to a large spectrum of management problems, not only to the field of intrusion detection. When instantiating the generic procedure to a specific field, the challenge is to properly define the optimization problem (Step 1 in the example in Table I) and the analytical relation between the performance measures and the system parameters (Step 2 in Table I).

In this paper, we have applied the proposed optimization procedure to the specific problem of a flow-based, history-based, probabilistic detection system, presented in Section III, which we validated for SSH attacks. Many aspects of the proposed solution can nevertheless be applied to different contexts. Let us consider Step 1 of the procedure in Table I. In this step, we propose an optimization problem that formalizes the trade-off between TN and TP. Such a trade-off plays a crucial role in virtually any detection problem; as we argued in Section I, the number of missed anomalies and the number of false alarms (and hence also TN and TP) are strongly correlated measures. Therefore, this type of formalization can also be applied to other scenarios.

In Step 2, as described in Table I, it is assumed that the likelihoods of benign and malicious traffic can be described as probability distributions. The precondition for Step 2 is therefore that such probability distributions exist, or in other words, that the system behavior is stationary. This means that, in order to create such distributions, the system should guarantee a certain stability of behavior, i.e., it should react in a similar way to similar inputs over time. This condition is satisfied for all systems for which the detection model is fixed at training time, or for adaptive systems with a slow cycle of retraining. It should be noted that the choice of modeling the benign and malicious likelihood distributions as Gaussian Mixtures is instead specific to our example. For other IDSs, other distributions may be more suitable.

Let us now consider how the proposed procedure can be applied to other intrusion detection methods. We have focused our attention on a flow-based, history-based system. Our approach could also be applied to other history-based systems, such as the work of Qian *et al.* [41]. In [41], the authors proposed an approach based on HMMs for the problem of malicious system call sequence classification. One of the parameters is the length of a system call sequence, which can be considered equivalent to the parameter w in our setup.

It is emphasized, though, that our approach is not restricted to history-based systems, since the optimization procedure does not depend on the type of analyzed data. For example, consider the case of a payload-based system that first assigns to each packet a likelihood of being malicious and then uses these values to determine when to report an attack. Independently of which technique is used for calculating the likelihood, the analysis of the performance results with respect to varying the internal system parameters would lead to a system optimization.

Finally, it should be noted that, in this paper, our running example is the detection of SSH dictionary attacks. However, the approach can be extended to other classes of attacks by using a different IDS and, more importantly, by adequately defining Step 2 of the procedure in Table I.

B. The binary classifier

In this paper, we based our detection system on a binary classifier. Binary classifiers are a generally accepted approach to the evaluation of IDSs, since it is generally assumed that a traffic instance is either malicious or not. From a user perspective, this is most likely true. However, from an IDS perspective, the assumption does not necessary hold: there might exist situations in which the system does not have enough information to decide on the nature of the analyzed traffic. For example, our system is likely to erroneously classify observation sequences in the beginning and ending phases of an attack, as we see in Section VII-C2, due to the influence of the past history on the decision process. In such cases, a binary classifier introduces errors that impact on the overall performance of the detection system. We argue that the overall performance of the system could be improved by introducing in the classification process a new state, indicating that the system is "uncertain" regarding the nature of the analyzed traffic. In our opinion, the advantage of a third state is twofold. First, it can help reducing the classification errors since the detection system would classify only traffic instances that are considered *certainly* benign or malicious. Second, the state of uncertainty can also become informative for the system manager: the degree of uncertainty on the nature of the traffic, for example, could be presented to the system manager as an alert level, notifying a change in the traffic behavior that could be notice of the beginning of an attack.

Finally, it should be noted that other types of IDSs classify traffic instances into a set of classes, corresponding to different attack types. An example of such type could be an IDS based on neural networks. However, such a system would not be an anomaly-based one, since the detection model should be trained on samples of both malicious and benign traffic. In this sense, these systems fall outside the scope of this paper. From the point of view of the system performance, a system detecting several classes of attacks can still be evaluated in terms of the classical confusion matrix, by considering only the overall number of flagged or missed attacks. Note however, that in this way we would not know if an attack, besides being flagged, has also been correctly labeled.

IX. Conclusions

This paper proposes an autonomic approach for tuning the parameters of anomaly-based intrusion detection systems. In particular, our contribution is twofold. First, we highlight the importance of tuning the parameters of an anomaly detection system to achieve an optimal performance level. We hope that our analysis can bring new awareness while creating or deploying a new anomaly-based system. In the paper, we propose to approach the parameter-tuning problem by formalizing it in terms of an optimization procedure. The optimization procedure allows to explicitly relate the system parameters and the performance measures in the form of an optimization problem. A key characteristic of the proposed solution is that it regards optimality according to high-level policies.

Second, in this paper we show how to apply the optimization procedure to an HMM-based IDS. In this case, the optimization problem consists in maximizing the number of correctly classified observation sequences. In our example, the policy expresses the relative importance of detecting all the attacks versus keeping the false alarm rate low. Our validation showed that, by varying the relative importance, we are able to fine-tune the system to favor either the detection of all the anomalies or the detection of attacks only when they are certain. Our findings also show that the relative importance has impact on the detection rate and on how timely the system is able to detect an attack or recover from it. We therefore believe that, when expressing a usage policy in terms of the relative importance, it should be taken into account that such a policy affects the system performance in multiple ways. The choice of which would be a suitable policy, is left to the system manager.

Acknowledgment

This research has been supported by the EU FP7-257513 UniverSelf Collaborative Project and the SURFnet's GigaPort3 project for Next-Generation Networks.

Appendix A
Hidden Markov Models

An HMM is a discrete time Markov chain (DTMC) where each state is augmented with a probability distribution over a finite set of output symbols.

With $S = \{s_1, \dots, s_n\}$ we denote the finite set of hidden states. S is called *hidden chain*. With q_t we denote the state at time t, and with a_{ij} the probability of jumping from state s_i to state s_j. Since we are dealing with a DTMC, this probability only depends on the current state s_i, i.e.,

$$a_{ij} = \mathbb{P}(q_{t+1} = s_j \mid q_t = s_i);$$

obviously, $\sum_j a_{ij} = 1$. With π_i we denote the probability of the initial state being s_i, i.e.,

$$\pi_i = \mathbb{P}(q_1 = s_i).$$

Now let $K = \{k_1, \dots k_m\}$ represent the finite set of output symbols. With o_t we denote the output symbol seen at time t. Define by $b_{s_i}(k)$ the probability of seeing output symbol k when the hidden state is s_i, i.e.,

$$b_{s_i}(k) = \mathbb{P}(o_t = k \mid q_t = s_i).$$

Given an *observation sequence* $O = \{o_1, \dots o_t\}$, we let $Q = \{q_1 \dots q_t\}$ be the *hidden sequence* that generates O. It is important to notice that several different hidden sequences Q may lead to the same observation sequence O. Finally, we refer to this model using the compact notation $\lambda = (A, B, \pi)$.

As described in [28], HMMs allow to estimate the probability of a sequence of observation $O = \{o_1, \dots o_t\}$, indicated as $\mathbb{P}(O \mid \lambda)$. Now $\mathbb{P}(O \mid \lambda)$ allows to score 'how well a model matches a given observation sequence' [28]. This motivates why this measure is interesting for anomaly detection: it indicates how likely it is that a sequence of network measurements is observed. Our assumption is that unlikely sequences denote the presence of a network anomaly.

We can compute $\mathbb{P}(O \mid \lambda)$ based on both the observation sequence $O = \{o_1, \dots o_t\}$ and the set $\{Q\}$ of all possible hidden sequences associated to O, as follows:

$$\mathbb{P}(O \mid \lambda) = \sum_{\{Q\}} \pi_{q_1} b_{q_1}(o_1) \prod_{i=2}^{t} a_{q_{i-1}q_i} b_{q_i}(o_i). \qquad (12)$$

References

[1] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Computers & Security*, vol. 28, no. 1-2, pp. 18 – 28, 2009.

[2] A. Sperotto, "Flow-Based Intrusion Detection," Ph.D. dissertation, Universiteit Twente, October 2010.

[3] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An Overview of IP Flow-based Intrusion Detection," *IEEE Communications Surveys & Tutorials*, vol. 12, no. 3, pp. 343–356, 2010.

[4] A. Pras, R. Sadre, A. Sperotto, T. Fioreze, D. Hausheer, and J. Schoenwaelder, "Using netflow/ipfix for network management," *Journal of Network and Systems Management*, vol. 17, no. 4, pp. 482–487, 2009.

[5] J. Quittek, T. Zseby, B. Claise, and S. Zander, "Requirements for IP Flow Information Export (IPFIX)," RFC 3917 (Informational).

[6] A. Sperotto and A. Pras, "Flow-based intrusion detection," in *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, 2011, pp. 958 –963.

[7] A. Sperotto, R. Sadre, P. de Boer, and A. Pras, "Hidden Markov Model modeling of SSH brute-force attacks," in *Proc. of the 20th IFIP/IEEE Int. Workshop on Distributed Systems: Operations and Management (DSOM '09)*, 2009, pp. 164–176.

[8] V. A. Siris and F. Papagalou, "Application of anomaly detection algorithms for detecting syn flooding attacks," *Computer Communications*, vol. 29, no. 9, pp. 1433 – 1442, 2006.

[9] Z. Yu, J. J. P. Tsai, and T. Weigert, "An adaptive automatically tuning intrusion detection system," *ACM Trans. Auton. Adapt. Syst.*, vol. 3, pp. 1–25, 2008.

[10] S. il Kim, N. Nwanze, and J. Kintner, "Towards dynamic self-tuning for intrusion detection systems," in *Proc. of the 29th Int. IEEE Performance Computing and Communications Conference (IPCCC'10)*, 2010, pp. 17 –24.

[11] Y. Himura, K. Fukuda, K. Cho, and H. Esaki, "An evaluation of automatic parameter tuning of a statistics-based anomaly detection algorithm," *Int. J. Netw. Manag.*, vol. 20, pp. 295–316, 2010.

[12] L. Ho, D. Cavuto, S. Papavassiliou, and A. Zawadzki, "Adaptive/automated detection of service anomalies in transaction-oriented WANS: Network analysis, algorithms, implementation, and deployment," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 5, pp. 744–757, 2000.

[13] M. Thottan and C. Ji, "Proactive anomaly detection using distributed intelligent agents," *IEEE Network*, vol. 12, pp. 21–27, 1998.

[14] ——, "Anomaly detection in IP networks," *IEEE Transactions on Signal Processing*, vol. 51, pp. 2191–2204, 2003.

[15] M. Mandjes, I. Saniee, and A. Stolyar, "Load characterization, overload prediction, and load anomaly detection for voice over IP traffic," *IEEE Transactions on Neural Networks*, vol. 16, no. 5, pp. 1019–1028, 2005.

[16] M. Mandjes and P. Żuraniewski, "A queueing-based approach to overload detection," in *Proc. of the 3rd Euro-NF Conference on Network Control and Optimization (NET-COOP '09)*, 2009, pp. 91–106.

[17] D. Siegmund, *Sequential Analysis*. Springer-Verlag, 1985.

[18] S.-Y. Lin, J.-C. Liu, and W. Zhao, "Adaptive CUSUM for anomaly detection and its application to detect shared congestion," Texas A&M University, Technical Report TAMU-CS-TR-2007-1-2, 2007.

[19] G. Münz and G. Carle, "Application of forecasting techniques and control charts for traffic anomaly detection," in *Proc. 19th ITC Specialist Seminar on Network Usage and Traffic*, 2008.

[20] A. Tartakovsky and V. Veeravalli, "Changepoint detection in multichannel and distributed systems with applications," *Applications of Sequential Methodologies*, pp. 331–363, 2004.

[21] E. Freire, A. Ziviani, and R. Salles, "Detecting voip calls hidden in web traffic," *Network and Service Management, IEEE Transactions on*, vol. 5, no. 4, pp. 204 –214, 2008.

[22] A. Dainotti, W. de Donato, A. Pescapè, and P. Rossi, "Classification of Network Traffic via Packet-Level Hidden Markov Models," in *Proc. of IEEE Global Telecommunications Conference (GLOBECOM 2008)*, 2008, pp. 1–5.

[23] T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys Tutorials*, vol. 10, no. 4, pp. 56 –76, 2008.
[24] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," *SIGMETRICS Perform. Eval. Rev.*, vol. 33, pp. 50–60, 2005.
[25] J. François, H. Abdelnur, R. State, and O. Festor, "Machine learning techniques for passive network inventory," *IEEE Trans. on Network and Service Management*, vol. 7, no. 4, pp. 244 –257, 2010.
[26] F. Camastra and A. Vinciarelli, *Markovian Models for Sequential Data*. Springer London, 2008.
[27] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains," *Annals of Mathematical Statistics*, vol. 41, no. 1, pp. 164–171, 1970.
[28] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.
[29] G. A. Fink, *Markov Models for Pattern Recognition: From Theory to Applications*. Springer-Verlag, 2008.
[30] D. X. Song, D. Wagner, and X. Tian, "Timing analysis of keystrokes and timing attacks on ssh," in *Proc. of the 10th conference on USENIX Security Symposium - Volume 10*, 2001, pp. 25–25.
[31] C. V. Wright, F. Monrose, and G. M. Masson, "HMM Profiles for Network Traffic Classification," in *Proc. of the Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC '04)*, 2004, pp. 9–15.
[32] D. Gao, M. K. Reiter, and D. X. Song, "Behavioral Distance Measurement Using Hidden Markov Models," in *Proc. of the 9th Int. Symposium Recent Advances in Intrusion Detection (RAID '06)*, 2006, pp. 19–40.
[33] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting Intrusions Using System Calls: Alternative Data Models," in *Proc. of the 1999 IEEE Symposium on Security and Privacy*, pp. 133–145, 1999.
[34] C. Seifert, "Analyzing Malicious SSH Login Attempts," http://www.securityfocus.com/infocus/1876, May 2011.
[35] SANS Institute, "Top-20 2007 Security Risks (2007 Annual Update)," www.sans.org, May 2011.
[36] A. Sperotto, R. Sadre, D. F. van Vliet, and A. Pras, "A Labeled Data Set For Flow-based Intrusion Detection," in *Proc. of the 9th IEEE Int. Workshop on IP Operations and Management (IPOM '09)*, 2009, pp. 39–50.
[37] A. Sperotto, R. Sadre, and A. Pras, "Anomaly Characterization in Flow-Based Traffic Time Series," in *Proc. of the 8th IEEE Int. Workshop on IP Operations and Management (IPOM '08)*, 2008, pp. 15–27.
[38] H. Debar, M. Dacier, and A. Wespi, "A Revised Taxonomy for Intrusion Detection Systems," *Annales des Telecommunications*, vol. 55, no. 7–8, pp. 361–378, 2000.
[39] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.
[40] D. Bolzoni, "Revisiting anomaly-based network intrusion detection systems," Ph.D. dissertation, University of Twente, Enschede, June 2009.
[41] Q. Qian and M. Xin, "Research on Hidden Markov Model for System Call Anomaly Detection," in *Proc. of Pacific Asia Workshop on Intelligence and Security Informatics (PAISI '07)*, 2007, pp. 152–159.

Anna Sperotto is postdoctoral researcher at the Design and Analysis of Communication Systems Group (DACS) of the University of Twente, The Netherlands. She received a MSc degree in Computer Science from the Ca'Foscari University, Venice, Italy, in 2006 and a PhD degree from the University of Twente, in 2010. Her main topics of interest include intrusion detection and traffic modeling.

Michel Mandjes received M.Sc. (in both mathematics and econometrics) and Ph.D. degrees from the Vrije Universiteit (VU), Amsterdam, the Netherlands. After having worked as a member of technical staff at KPN Research (Leidschendam, the Netherlands) and Bell Laboratories/Lucent Technologies (Murray Hill NJ, USA), as a part-time full professor at the University of Twente, and as department head at CWI, Amsterdam, he currently holds a full professorship (chair of Applied Probability) at the University of Amsterdam, the Netherlands. He is also affiliated (as an advisor) with EURANDOM, Eindhoven, the Netherlands. In 2008, Mandjes spent a sabbatical at Stanford. His research interests include performance analysis of communication networks, queueing theory, advanced simulation methods, Gaussian traffic models, traffic management and control, and pricing in multi-service networks. He is the author of Large Deviations for Gaussian Queues, Wiley, 2007. He is currently editor of "Queueing Systems", "Stochastic Models", "Stochastic Systems", and "Journal of Applied Probability".

Ramin Sadre is a postdoctoral researcher at the Design and Analysis of Communication Systems Group (DACS) of the University of Twente, The Netherlands. In 2006, he received a PhD degree from the same university with the title "Decomposition Based Analysis of Queueing Networks". He is currently active in the European FP7 Integrated Project UniverSelf on autonomic network management. His research interests include traffic modeling, the performance evaluation of communication systems, and the design of intrusion detection systems.

Pieter-Tjerk de Boer received the MSc degree in applied physics in 1996 and the PhD degree in computer science in 2000, both from the University of Twente, The Netherlands. He is currently an assistant professor at the department of Electrical Engineering, Mathematics and Computer Science at the University of Twente. His research interests include communication networks, their mathematical performance modelling and simulation, and rare-event simulation techniques in particular.

Aiko Pras is Associate Professor in the Departments of Electrical Engineering and Computer Science at the University of Twente, the Netherlands where he is leading the Design and Analysis of Communication Systems Group. He received a PhD degree for his thesis titled "Network Management Architectures". His research interests include network management technologies, network monitoring, measurements and security. He is chairing the IFIP Working Group 6.6 on "Management of Networks and Distributed Systems", is steering committee member of the major conferences in this field, and series/associate editor of IEEE ComMag, TNSM and IJNM.