2016 IEEE 27th International Symposium on Software Reliability Engineering

# Quantifying the Attack Detection Accuracy of Intrusion Detection Systems in Virtualized Environments

Aleksandar Milenkoski*, K. R. Jayaram†, Nuno Antunes‡, Marco Vieira‡ and Samuel Kounev*
*University of Würzburg, Würzburg, Germany
Email: {aleksandar.milenkoski, samuel.kounev}@uni-wuerzburg.de
†IBM Research, NY, USA
Email: jayaramkr@us.ibm.com
‡University of Coimbra, Coimbra, Portugal
Email: {nmsa, mvieira}@dei.uc.pt

***Abstract*—With the widespread adoption of virtualization, intrusion detection systems (IDSes) are increasingly being deployed in virtualized environments. When securing an environment, IT security officers are often faced with the question of how accurate deployed IDSes are at detecting attacks. To this end, metrics for assessing the attack detection accuracy of IDSes have been developed. However, these metrics are defined with respect to a fixed set of hardware resources available to the tested IDS. Therefore, IDSes deployed in virtualized environments featuring elasticity (i.e., on-demand allocation or deallocation of virtualized hardware resources during system operation) cannot be evaluated in an accurate manner using existing metrics. In this paper, we demonstrate the impact of elasticity on IDS attack detection accuracy. In addition, we propose a novel metric and measurement methodology for accurately quantifying the accuracy of IDSes deployed in virtualized environments featuring elasticity. We demonstrate their practical use through case studies involving commonly used IDSes.**

***Index Terms*—intrusion detection; accuracy; elasticity; evaluation; metrics;**

## I. Introduction

Intrusion detection is the process of monitoring the events occurring in a computer or networked system and analyzing those events for signs of possible incidents [1]. It enables the timely detection of intrusions, for example, attacks triggering software faults, such as vulnerabilities. Intrusion detection systems (IDSes) — software systems automating the intrusion detection process — are therefore crucial security mechanisms.

Virtualization allows the creation of logical (i.e., "virtual") instances of physical devices, such as network and processing units. In a virtualized system, governed by a hypervisor, resources are shared among virtual machines (VMs). Each VM accesses the physical resources of the infrastructure through the hypervisor and is entitled to a fraction of capacity.

The wide adoption of virtualization has led to the emergence of novel IDSes specifically designed to operate in virtualized environments, such as ACPS (Advanced Cloud Protection System) [2] and vShield Endpoint [3]. These IDSes have components both inside the hypervisor and in a designated VM. The increased adoption of virtualization has also led to the practice of deploying conventional IDSes (e.g., hardware IDS appliances or common software-based IDSes) as virtualized network functions (VNFs). For instance, a network-based IDS, such as Snort [4] or Suricata [5], may be deployed in a designated VM and configured to tap into the physical network interface card used by all VMs. Deploying an IDS in a designated VM brings practical benefits – the IDS can monitor the activities of all co-located VMs at the same time while being isolated from, and transparent to, their users.

Methods and techniques for the rigorous testing of IDSes are needed. The benefits of IDS evaluation are manyfold. For instance, one may compare multiple IDSes in terms of their attack detection accuracy in order to deploy an IDS which operates optimally in a given environment, thus reducing the risks of a security breach. Further, one may tune an already deployed IDS by varying its configuration parameters and investigating their influence through evaluation tests. Any IDS evaluation experiment requires careful planning, which includes selection of metrics and measurement methodologies for quantifying IDS properties (e.g., attack detection accuracy).

A common aspect of all existing metrics for quantifying IDS attack detection accuracy (which we refer to as IDS evaluation metrics) is that they are defined with respect to a *fixed* set of hardware resources available to the IDS under test [6]. However, a virtualized environment may have *elastic* properties. Under elasticity, we understand on-demand provisioning (i.e., allocation or deallocation) of virtualized hardware resources to VMs, which is also known as vertical VM scaling [7]. For example, the hypervisor may be configured to hotplug, or deallocate, virtualized resources on, or from, a VM where an IDS operates, which may have a significant impact on the attack detection accuracy of the IDS (e.g., the IDS may miss attacks due to lack of resources). Thus, using existing IDS evaluation metrics for evaluating IDSes deployed in virtualized environments may lead to inaccurate measurements.

This paper makes the following contributions:

*(i)* it reviews conventional IDS evaluation metrics, and demonstrates how elasticity of virtualized environments may

2332-6549/16 $31.00 © 2016 IEEE
DOI 10.1109/ISSRE.2016.39

impact IDS attack detection accuracy;

*(ii)* it empirically demonstrates that using conventional IDS evaluation metrics may lead to inaccurate measurements when it comes to evaluating IDSes deployed in virtualized environments featuring elasticity;

*(iii)* it proposes a novel metric and measurement methodology that allow for quantifying the impact of elasticity on IDS attack detection accuracy. We designed the metric with respect to a set of criteria for the accurate and practically useful evaluation. For example, the metric penalizes resource deallocation causing the IDS to miss attacks due to lack of resources. Our metric is meant to complement conventional metrics — it is specifically designed for evaluating IDSes that perform run-time monitoring and are deployed in virtualized environments featuring elasticity.

*(iv)* it demonstrates the practical use of the metric and measurement methodology we propose. We evaluate two commonly used IDSes (i.e., Snort [4] and Suricata [5]) running on top of the Xen hypervisor. We consider 15 different configurations of the IDSes and the hypervisor.

This paper is organized as follows: In Section II, we provide an overview of conventional IDS evaluation metrics; in Section III, we discuss and demonstrate the impact of elasticity on IDS attack detection accuracy; in Section IV, we present a novel metric and measurement methodology, which take elasticity into account; in Section V, we demonstrate the practical use of the proposed metric and measurement methodology; in Section VI, we discuss future work and conclude this paper.

## II. Related Work

In this section, we review commonly used IDS evaluation metrics. We distinguish between *basic* and *composite* IDS evaluation metrics.

*a) Basic metrics:* The basic metrics quantify various individual attack detection properties. For instance, the false negative rate $\beta = P(\neg A|I)$ quantifies the probability that an IDS does not generate an alert when an intrusion occurs;[1] therefore, the true positive rate $1-\beta = 1-P(\neg A|I) = P(A|I)$ quantifies the probability that an IDS generates an alert when an intrusion occurs. The false positive rate $\alpha = P(A|\neg I)$ quantifies the probability that an alert generated by an IDS is not an intrusion, but a regular benign activity; therefore, the true negative rate $1 - \alpha = 1 - P(A|\neg I) = P(\neg A|\neg I)$ quantifies the probability that an IDS does not generate an alert when an intrusion does not occur.

*b) Composite metrics:* IDS evaluators often combine the basic metrics in order to analyze relationships between them, for example, to identify an optimal *IDS operating point* — an IDS configuration which yields optimal values of both the true and false positive detection rate — or to compare multiple IDSes.

It is a common practice to use a *ROC (Receiver Operating Characteristic) curve* to investigate the relationship between the true and false positive detection rate exhibited by an IDS. A ROC curve plots true positive rate against the corresponding false positive rate [8]; that is, a ROC curve depicts multiple operating points of an IDS under test and, as such, it is useful for identifying the optimal operating point or for comparing IDSes. However, ROC curves do not express the impact of the rate of occurence of intrusion events ($B = P(I)$; prior probability that an intrusion event occurs), known as *base rate*, on $\alpha$ and $1 - \beta$. The error occurring when $\alpha$ and $1 - \beta$ are assessed without taking the base rate into account is known as the base-rate fallacy [9].

Security researchers have proposed metrics that are more expressive than ROC curves. Gaffney and Ulvila [10] propose a metric called '*expected cost*' ($C_{exp}$). They combine ROC curve analysis with cost estimation by associating an estimated cost with each IDS operating point.

$C_{exp}$ can be calculated for each considered IDS operating point. The formula of $C_{exp}$ is constructed using a decision tree as a basis (see [10]). Using $C_{exp}$, one can identify an optimal IDS operating point in a straightforward manner — an operating point is considered optimal if it has the lowest $C_{exp}$ associated with it. One can compare IDSes by comparing the estimated costs when each IDS operates at its optimal operating point. The IDS that has lower $C_{exp}$ associated with its optimal operating point is considered the best.

Gu et al. [11] propose a metric called '*intrusion detection capability*' ($C_{ID}$). $C_{ID}$ incorporates the base rate $B$ and many basic metrics, such as $(1 - \beta)$ and $\alpha$ (see [11]). Therefore, a value of $C_{ID}$ may be assigned to any operating point of an IDS on the ROC curve. This allows for a straightforward identification of the optimal operating point of an IDS — the point that marks the highest $C_{ID}$. One can compare IDSes by analyzing the maximum $C_{ID}$ of each IDS and considering as better performing the IDS whose optimal operating point has higher $C_{ID}$ associated with it.

As we mentioned in Section I, a common aspect of all metrics discussed above is that they are defined with respect to a fixed set of resources available to the IDS under test [6], [12]. However, since virtualized environments may be elastic (see Section I), using these metrics may lead to inaccurate measurements when it comes to evaluating IDSes deployed in such environments. This, in turn, may result in the deployment of ill-performing IDSes, increasing the risk of security breaches. We discuss on this topic in detail in Section III.

## III. Elasticity and Accuracy

The elastic behavior of a virtualized environment, that is, the hypervisor applying a resource provisioning policy, may have significant impact on the attack detection accuracy exhibited by the IDS deployed in the environment and performing real-time monitoring. This impact is caused by the hypervisor impacting relevant *transient behaviors* of the IDS. Under relevant transient IDS behaviors, we understand IDS behaviors that are influenced by the amount of resources available to an IDS over time and may impact the IDS's attack detection accuracy. For example, the attack detection accuracy exhibited

[1] $P$ denotes a probability; $A$ denotes an alert event (i.e., an IDS generates an attack alert); $I$ denotes an intrusion event.

by a network-based IDS may be correlated to the number of packets dropped by the IDS in the time intervals when attacks are performed. Large amounts of dropped packets in such intervals due to lack of resources may manifest themselves as low IDS attack detection accuracy.

In this section, we demonstrate the impact of the hypervisor, and therefore, of transient IDS behaviors influenced by it (i.e., amount of dropped packets over time), on IDS attack detection accuracy. We demonstrate through this case study the impact of CPU hotplugging considering the case of a network-based IDS deployed as a VNF. We deployed Suricata 2.0.9 in a paravirtualized VM running on top of the Xen 4.4.1 hypervisor, and allocated 12 GB of main memory and a NIC with a maximal data transfer rate of 1 Gbit/second to this VM. We replayed over 240 seconds, at the speed of 150 Mbps, a trace file from the DARPA datasets.[2] All configuration options of Suricata were set to their default values. For each considered experimental scenario, we repeated measurements 30 times and we averaged the results.

We first considered three separate experimental scenarios, where we hotplug two, three, and four virtual CPUs of 2.6 GHz on the VM where Suricata is deployed. In Table I, we present the true positive rate ($1-\beta$) measured for each scenario (2/3/4 CPUs in Table I), in relation to the total amount of dropped packets.

We then allocated two virtual CPUs of 2.6 GHz to the VM where Suricata was deployed so that the VM is under CPU pressure when workloads are run. This enabled us to observe the impact of CPU hotplugging on IDS attack detection accuracy in scenarios where such a hotplugging is normally performed. We considered two separate experimental scenarios, where we hotplug one and two additional virtual CPUs on the VM where Suricata is deployed, at the 120th second of each experiment.

In Figure 1, we depict the number of packets dropped by Suricata over 240 seconds for each considered CPU hotplugging scenario (2→3/4 CPUs in Figure 1). In Table I, section 'CPU hotplugging', we present the true positive rate ($1-\beta$) measured for each scenario, in relation to the total amount of dropped packets. As expected, $1-\beta$ increases as more CPUs are hotplugged on the VM where Suricata is deployed. This is due to the decrease of the number of packets dropped by Suricata after CPUs have been hotplugged (see the trend lines in Figure 1).

In Table I, column 'no. of true alerts', we present the actual number of true alerts issued by Suricata. One can observe that

[2]The used trace file is available at http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/1998/testing/week1/monday/tcpdump.gz. The base rate $B$ is 0.025.

We are aware that the DARPA datasets are outdated, however, this is not relevant to the purpose of this paper; that is, any dataset (workload) can be used for demonstrating the impact of elasticity on IDS attack detection accuracy and the use of the metric and measurement methodology we propose (see Section I). We use the DARPA datasets since they are provided together with comprehensive "ground truth" information (i.e., information about the attacks recorded in the datasets, for example, time of execution of the attacks). The output of an evaluated IDS is compared with "ground truth" information in order to quantify the attack detection accuracy of the IDS.

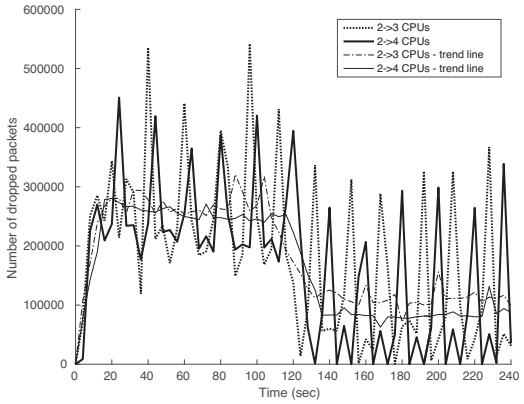


Fig. 1: Number of packets dropped over time

TABLE I: Attack detection accuracy of Suricata

| Scenario | $1-\beta$ | No. of true alerts | Dropped packets (%) |
|---|---|---|---|
| 2 CPUs | 0,393 | 80184 | 49,32% |
| 3 CPUs | 0,507 | 103444 | 33,60% |
| 4 CPUs | 0,562 | 114659 | 22,77% |
| CPU hotplugging | | | |
| 2 → 3 CPUs | 0,465 | 94906 | 37,57% |
| 2 → 4 CPUs | 0,484 | 98771 | 34,46% |

Suricata issued 14722 (i.e., 18.36%) true alerts more when one additional CPU (2→3 CPUs in Table I) was provisioned in comparison to when only two CPUs (2 CPUs in Table I) were made available to the IDS over 240 seconds. Further, Suricata issued 3865 true alerts more when two (2→4 CPUs in Table I), instead of one (2→3 CPUs in Table I), additional CPUs were provisioned. This effectively demonstrates the impact of the hypervisor on Suricata's attack detection accuracy.

In a scenario such as the above, where an IDS evaluator aims to understand the relation between a given transient behavior of an IDS and the attack detection accuracy the IDS exhibits, the use of conventional IDS evaluation metrics (see Section II) introduces the following inter-related issues:

*Approximative metric value correlation*: The IDS evaluator would have to correlate values of metrics belonging to two categories: (i) metrics that quantify attack detection accuracy (e.g., true positive rate), and (ii) metrics that quantify the considered transient IDS behavior (e.g., amount of dropped packets over time). However, given the lack of metrics and measurement methodologies specifically designed for this purpose, such a correlation would be approximative, which may lead to inaccurate assessments;

*Inaccurate comparisons of IDSes*: The approximative nature of the correlation mentioned above rules out accurate comparisons of the attack detection accuracy of multiple IDSes. Note that comparing IDSes is a common goal of IDS evaluation studies [13], where precise measurements of considered metric values are crucial so that the comparisons are accurate and fair.

The metric and measurement methodology we propose aim to address the above issues.

## IV. Metric and Measurement Methodology

In this section, we propose a novel metric and measurement methodology, which enable the accurate measurement of the attack detection accuracy of an IDS that performs run-time monitoring and is deployed in a virtualized environment featuring elasticity; that is, they enable the evaluation of the attack detection accuracy exhibited by such an IDS with respect to the impact that on-demand resource provisioning performed by the underlying hypervisor has on the accuracy.

We name the metric we propose *hypervisor factor (HF)*, since it quantifies the impact of the hypervisor as a factor impacting IDS attack detection accuracy. Quantifying this impact calls for a novel definition of the boundaries of a system-under-test (SUT) in the area of IDS evaluation. The precise definition of the boundaries of an SUT is critical for the accurate measurement of system performance and interpretation of evaluation results. In contrast to the conventional understanding in the area of IDS evaluation about what comprises an SUT (i.e., the IDS under test), we advocate a novel SUT with *extended* boundaries including the hypervisor as well. In Figure 2, we depict the boundaries of the conventional SUT in the area of IDS evaluation and of the novel SUT we propose considering a network-based IDS deployed as a VNF.

### A. Metric design

We distinguish three states in which a given IDS, part of an SUT as we define it, may be over the duration of an IDS evaluation experiment: baseline, underprovisioned, and overprovisioned state. By baseline IDS state, we mean a state of the IDS in which it is provisioned by the hypervisor with the minimum amount of resources such that provisioning more resources does not have an impact on the attack detection accuracy of the IDS (e.g., it does not improve the positive rate exhibited by the IDS, see Section III). Therefore, by overprovisioned, or underprovisioned, IDS state, we mean a state of the IDS in which it is provisioned by the hypervisor with more, or less, resources than the amount needed for the IDS to be considered in baseline state. Given these definitions of IDS states, we design the HF metric with respect to the following criteria, which are crucial for the accurate and practically useful IDS evaluation:

***Criterion*** $C_1$: If configured accordingly, the HF metric penalizes resource overprovisioning with respect to the

a) time the IDS has spent in overprovisioned state over the duration of an IDS evaluation experiment, and

b) the false positive and false negative rate exhibited by the IDS under test when in overprovisioned state, since provisioning excess amount of resources has not contributed towards improving the accuracy of the IDS. We design the HF metric to penalize equally various extents of overprovisioning since we consider any extent of overprovisioning an equally negative phenomenon;

***Criterion*** $C_2$: If configured accordingly, the HF metric penalizes resource underprovisioning with respect to the

a) time the IDS has spent in underprovisioned state over the duration of an IDS evaluation experiment, and

b) the extent of the impact that the underprovisioning has had on the true positive rate exhibited by the IDS. We consider this impact a negative phenomenon since it causes the reduction of the number of true alerts issued by the IDS (see Section III). The HF metric does not penalize resource underprovisioning that has had no impact on the true positive rate since we consider resource saving, which does not cause reduction of this rate, a positive phenomenon.

***Criterion*** $C_3$: If configured accordingly, the HF metric rewards resource underprovisioning with respect to the

a) time the IDS has spent in underprovisioned state over the duration of an IDS evaluation experiment, and

b) the extent of the impact that the underprovisioning has had on the false positive rate exhibited by the IDS. We consider this impact a positive phenomenon since it brings practical benefits - reduced number of issued false alerts and increased amount of saved resources. Underprovisioning may cause the reduction of the false positive rate exhibited by an IDS if a given amount of workload units (e.g., packets), which would have been falsely labelled as malicious by the IDS if processed by it, are not processed by the IDS due to lack of resources.

In summary, the HF metric favors the most an SUT configured in a way such that the hypervisor saves the most resources while impacting the true positive rate exhibited by the IDS to the least extent and the false positive rate exhibited by the IDS to the biggest extent.

***Criterion*** $C_4$: The HF metric expresses the base rate. The attack detection performance of an IDS should be assessed with respect to a base rate measure in order for such an assessment to be accurate (see Section II). Therefore, it is important that the HF metric expresses this rate.

***Criterion*** $C_5$: The HF metric enables the straightforward identification of optimal operating points. In the context of IDS evaluation, an optimal operating point is an IDS configuration which yields values of both the true and false positive rates considered optimal with respect to a given measure (e.g., cost, see Section II). In the context of this work, under optimal operating point, we understand a configuration of both the IDS under test *and* the underlying hypervisor, which yield values of metrics quantifying the performance of the hypervisor at provisioning resources and of metrics quantifying IDS attack detection accuracy (e.g., true and false positive rate) considered optimal with respect to the impact of the former on the latter (see criterion $C_1$, $C_2$, and $C_3$). This is because we consider a novel SUT with boundaries that include an IDS and a hypervisor provisioning the IDS with resources (see Figure 2b).

We design the HF metric to enable a straightforward identification of optimal operating points; that is, for a given set of operating points, the optimal operating point yields an extreme value of HF. In Section IV-C, we discuss more on operating points and on identifying optimal operating points.

***Criterion*** $C_6$: The HF metric enables the accurate comparison of multiple SUTs. This is feasible only if criterion $C_5$ is fulfilled, a topic that we discuss more in Section IV-C.

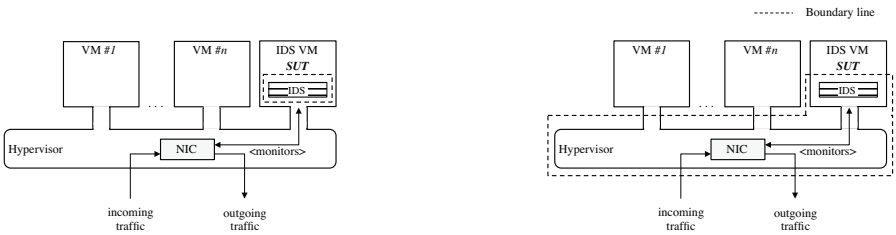


Fig. 2: Boundaries of: a) the conventional SUT, and b) novel SUT in the area of IDS evaluation

### B. Metric construction

We present here the main principles of construction for the HF metric. Similar to Gaffney et al. [10], we construct the HF metric using a construct from decision theory — a decision tree — as a basis. In Figure 3, we depict the decision tree that we use for constructing the HF metric. The tree shows the sequence of uncertain events (circles) that describe:

• the *workload*, say $W[B]$, to which the IDS is subjected over the duration of a given IDS evaluation experiment, say $T_{max}$. We characterize $W$ by the base rate (i.e., probability of an intrusion $B = P(I)$, see Section II);

• the *operation* of the IDS processing workload $W[B]$. The operation of the IDS is characterized by the probabilities of the IDS issuing or not issuing an alert when an intrusion has or has not occurred (i.e., the probabilities $P(A|I)$, $P(\neg A|I)$, and so on, see Section II);

• the *state* of the IDS (i.e., baseline, overprovisioned, or underprovisioned IDS state, see Section IV-A) when it issues or does not issue an alert. The IDS being in one of the considered states during operation primarily depends on the resource provisioning policy applied by the underlying hypervisor, say $H[T_o, T_b, T_u]$; that is, on its precision at meeting the demand for resources by the IDS over time $T_{max}$. We characterize $H$ by the amount of time the IDS has spent in overprovisioned ($T_o$), baseline ($T_b$), and underprovisioned ($T_u$) state over time $T_{max}$ (i.e., $T_{o/b/u} \in [0; T_{max}], T_o + Tb + Tu = T_{max}$).

Associated with each uncertain event is the probability of occurrence. There are six probabilities specified in the tree: $p_1 = P(I) = B$: the probability that an intrusion occurs; $p_2 = P(A|I) = 1 - \beta$: the probability that the IDS issues an alert when an intrusion occurs (i.e., the true positive rate); $p_3 = P(A|\neg I) = \alpha$: the probability that the IDS issues an alert when an intrusion does not occur (i.e., the false positive rate); $p_{4/5/6} = \frac{T_{o/b/u}}{T_{max}}$: the probability that the IDS under test is in overprovisioned/baseline/underprovisioned state when it issues or does not issue an alert (i.e., at any moment in the time interval $[0; T_{max}]$);

The attractiveness of each combination of events represented in the tree depicted in Figure 3 is characterized by the *consequence* (i.e., the penalty or the reward score) associated with it. With respect to the metric design criteria $C_1$, $C_2$, and $C_3$ (see Section IV-A), the HF metric:

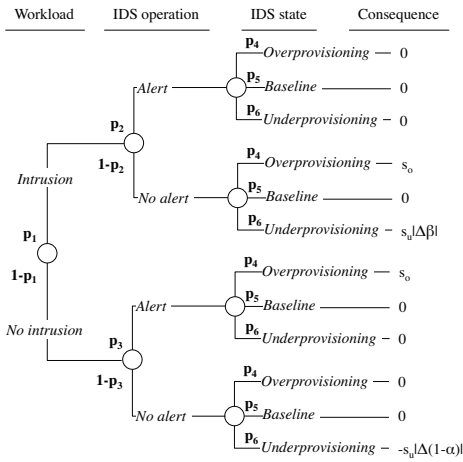


Fig. 3: The decision tree used for constructing the HF metric

• penalizes the SUT for the IDS issuing false positive or false negative alerts when the IDS is in overprovisioned state. A user of the HF metric may disable or enable this penalization by setting the value of $s_o, s_o \in \{0, 1\}$ to 0 or 1, respectively;

• penalizes the SUT for the IDS (in underprovisioned state) not issuing an alert when an intrusion has occurred with the score $s_u|\Delta\beta|$, where $|\Delta\beta| = |\beta - \beta_b|$. A user of the HF metric may disable or enable this penalization by setting the value of $s_u, s_u \in \{0, 1\}$ to 0 or 1, respectively. $\beta_b$ is the false negative rate exhibited by the IDS in a scenario where it has operated in baseline state over time $T_{max}$ and subjected to workload $W[B]$. Therefore, the HF metric quantifies the impact of underprovisioning on the true positive rate $(1 - \beta)$ exhibited by the IDS – it penalizes the SUT for the IDS not issuing a true alert because of discarded workloads due to lack of resources;

• rewards the SUT for the IDS (in underprovisioned state) not issuing an alert when an intrusion has not occurred with the score $s_u|\Delta(1 - \alpha)|$, $|\Delta(1 - \alpha)| = |(1 - \alpha) - (1 - \alpha)_b|$. A user of the HF metric may disable or enable this rewarding by setting the value of $s_u, s_u \in \{0, 1\}$ to 0 or 1, respectively. $(1 - \alpha)_b$ is the true negative rate exhibited by the IDS in a scenario where it has operated in baseline state over time $T_{max}$ and subjected to workload $W[B]$. Therefore, the HF metric quantifies the impact of underprovisioning on the false

positive rate ($\alpha$) exhibited by the IDS – it rewards the SUT for the IDS not issuing a false alert.

The formula of the HF metric can be obtained by "folding back" the decision tree depicted in Figure 3; that is, from right to left, the penalty, or the reward, score at an event node is the sum of products of probabilities and scores for each branch:

$$\begin{aligned} HF &= B[\beta(\frac{T_o}{T_{max}}s_o + \frac{T_u}{T_{max}}s_u|\Delta\beta|)] \\ &+ (1-B)[\alpha\frac{T_o}{T_{max}}s_o - (1-\alpha)\frac{T_u}{T_{max}}s_u|\Delta(1-\alpha)|)] \\ &= \frac{T_o}{T_{max}}s_o[B+(1-B)\alpha]\beta \\ &+ \frac{T_u}{T_{max}}s_u[B\beta|\Delta\beta| - (1-B)(1-\alpha)|\Delta(1-\alpha)|] \end{aligned} \tag{1}$$

If the values of $s_o$ and $s_u$ are set to 1, Equation 1 can be alternatively represented as the sum of the two components of the HF metric, that is, $HF_o$ and $HF_u$, where $HF_o = \frac{T_o}{T_{max}}[B\beta+(1-B)\alpha]$ is the penalty associated with overprovisioning and $HF_u = \frac{T_u}{T_{max}}[B\beta|\Delta\beta|-(1-B)(1-\alpha)|\Delta(1-\alpha)|]$ is the penalty, or reward, associated with underprovisioning. Distinguishing these components of the HF metric allows for separately observing the quantified consequences of the hypervisor over- and/or underprovisioning the IDS in relation to the attack detection accuracy exhibited by the IDS.

*1) On baseline IDS state:* Calculating values of the HF metric requires calculating $T_o$, $T_u$, and $T_b$, and, in addition, $\beta_b$ and $(1-\alpha)_b$ (see Equation 1). This, in turn, may require extensive experimentation in order to: (i) identify the baseline state of the IDS that is part of the SUT; that is, to determine the minimum amount of resources, say $R_b$, such that provisioning more resources does not have an impact on the attack detection accuracy exhibited by the IDS (see Section IV-A); and (ii) compare this amount with the amount of resources provisioned by the hypervisor applying a given resource provisioning policy $H[T_o, T_b, T_u]$, say $R_p$.

The above activities may be practically challenging because they require the use of measurement approaches considering various resource unit and measurement granularities, and determining how $R_b$ changes over time $T_{max}$ with respect to the intensity of the workload to which the IDS is subjected. Therefore, we assume the following simplifications:

- $R_b$ is constant over time $T_{max}$ — we consider $R_b$ the minimum amount of resources allocated to the VM where the IDS operates, such that the IDS does not discard workload when the workload is most intensive. This reflects a realistic scenario where resources are provisioned to an IDS considering the peak intensity of the workload that the IDS may process during operation;
- $R_b$ and $R_p$ differ with regard to a single measurement unit (e.g., MB of memory) — that is, we assume that the hypervisor allocates and/or deallocates a single type of resource over the duration of an IDS evaluation experiment. This allows for determining the difference between $R_b$ and $R_p$ over time $T_{max}$ in a straightforward and accurate manner.

We plan to address the above simplifications as part of our future work.

### C. Properties of the HF metric

In this section, we show how the HF metric satisfies each of the design criteria we presented in Section IV-A:

***Criterion*** $C_1$: For a given $T_{max}$, the value of the $HF_o$ component of the HF metric (see Section IV-B) is positively correlated with $T_o$ (i.e., the time the IDS has spent in overprovisioned state over time $T_{max}$), the false positive rate ($\alpha$), and the false negative rate ($\beta$);

***Criterion*** $C_2$ ***and*** $C_3$: For a given $T_{max}$, the value of the $HF_u$ component of the HF metric (see Section IV-B):

- is positively correlated with $T_u$ (i.e., the time the IDS has spent in underprovisioned state over time $T_{max}$) and $|\Delta\beta|$, which quantifies the extent of the impact that underprovisioning has had on the false negative rate, and therefore on the complementary true positive rate;
- is negatively correlated with $T_u$ and $|\Delta(1-\alpha)|$, which quantifies the extent of the impact that underprovisioning has had on the true negative rate, and therefore on the complementary false positive rate;

***Criterion*** $C_4$: The HF metric expresses the base rate $B$ (see Equation 1);

***Criterion*** $C_5$: In Definition 1, we define an operating point of an SUT (see Figure 2b).

*Definition 1:* An operating point of an SUT consisting of an IDS and a hypervisor, say $O(I \rightarrow (\alpha_b, 1-\beta_b); H[T_o, T_b, T_u]) \rightarrow (\alpha, 1-\beta)$, is a configuration $I$ of the IDS, which yields distinct values of $\alpha_b$ and $(1-\beta)_b$, and a configuration of the hypervisor, that is, a configured resource provisioning policy $H[T_o, T_b, T_u]$. These configurations yield values of $1-\beta$ and $\alpha$ (i.e., the true and false positive rate exhibited by the IDS with configuration $I$ in a scenario where the hypervisor applies resource provisioning policy $H[T_o, T_b, T_u]$).

A single value of the HF metric may be associated with a specific configuration of the IDS and of the hypervisor comprising a given SUT (i.e., with each operating point of the SUT considered in a given evaluation study, see Equation 1 and Definition 1). Given that the HF metric may penalize an SUT, a given operating point of the SUT is considered optimal if it has the lowest value of the HF metric associated with it. Theoretically, there may be more than one operating point having the same lowest value of the HF metric associated with them. In such a scenario, a given operating point may be considered optimal based on subjective criteria. For example, an IDS evaluator may consider optimal the operating point with the highest value of $T_b$ (i.e., the operating point such that the IDS spends at most time in baseline state).

Measuring values of the HF metric and identifying the optimal operating point of an SUT, out of multiple operating points, is performed in practice by executing multiple experiments using a given workload, and varying the configuration of the IDS and/or of the hypervisor between experiments.

***Criterion*** $C_6$: Multiple SUTs can be compared by comparing their optimal operating points—the SUT with the lowest value of the HF metric associated with its optimal operating point is considered best.

## V. Case Studies

We demonstrate here the practical use of the HF metric and the measurement methodology we propose. We conducted multiple experiments considering various SUTs and operating points, that is, configurations of the hypervisor (Section V-A) and of the IDS (Section V-B) comprising an SUT. We evaluate two commonly used IDSes (i.e., Snort [4] and Suricata [5]) running on top of the Xen hypervisor. We consider 15 different configurations of the IDSes and the hypervisor. For each considered experimental scenario, we repeated measurements 30 times and we averaged the results. The transient IDS behavior of interest is amount of packets dropped over time (see Section III).

It is important to emphasize that we focus on demonstrating the practical use of the HF metric and of the proposed measurement methodology, and not on discussing in depth the behavior of the considered SUTs. We specify arbitrary evaluation scenarios with the sole purpose of demonstrating how the proposed metric and measurement methodology may be used in practice for any evaluation and SUT deployment scenario. For example, we replay network traffic at the speed of 150 Mbps, which may not be representative of network traffic speed seen in production virtualized environments, however, this is not relevant to the goals of the experiments we conduct and is not a threat to the validity of the drawn conclusions.

We do not compare the HF metric with conventional IDS evaluation metrics (see Section II) since they cannot be practically compared. Note that the HF metric has a drastically different goal (see Section IV-A) than that of conventional IDS evaluation metrics, which, in contrast to the HF metric, quantify IDS attack detection accuracy and not the impact of elasticity on IDS attack detection accuracy.

### A. Hypervisor configurations

We structure this section with respect to the different hypervisor configurations we consider; we investigate the impact of two relevant characteristics of CPU and memory on-demand provisioning (i.e., hotplugging): (i) hotplugging *intensity* (i.e., amount of hotplugged resources), and (ii) hotplugging *speed* (i.e., the hypervisor's speed at provisioning resources with respect to resource demands).

*a) CPU hotplugging intensity and speed:* We first quantify the impact of CPU hotplugging intensity on IDS attack detecting accuracy. We deployed Suricata 2.0.9 in a paravirtualized VM running on top of the Xen 4.4.1 hypervisor. We allocated 12 GB of main memory and a NIC with a maximal data transfer rate of 1 Gbit/second to the VM where Suricata was deployed. We also allocated two virtual CPUs of 2.6 GHz to this VM so that the VM is under CPU pressure when workloads are run. This enabled us to observe the impact of CPU hotplugging on IDS attack detection accuracy in scenarios where such a hotplugging is normally performed. We replayed over 240 seconds, at the speed of 150 Mbps, a trace file from the DARPA datasets.[2] All configuration options of Suricata were set to their default values.

We considered two separate experimental scenarios, where we hotplug one and two additional virtual CPUs on the VM where Suricata is deployed, at the 120th second of the experiment; that is, we consider two operating points of the SUT, $O_1$ and $O_2$, respectively.

We first identified the baseline state of Suricata at 3 CPUs allocated to the VM where the IDS was deployed. This results in $T_o = 0, T_b = 120, T_u = 120$ for operating point $O_1$, and $T_o = 120, T_b = 0, T_u = 120$ for operating point $O_2$. We measured $\alpha_b$ of $0.182 \times 10^{-4}$ and $(1-\beta)_b$ of 1.0 (see Section IV-B). We then calculated values of the HF metric such that we enabled penalization of overprovisioning and penalization, or rewarding, of underprovisioning (i.e., $s_o, s_u = 1$). In Table II, section 'CPU hotplugging intensity', we present the operating points $O_1$ and $O_2$, the impact that underprovisioning has had on Suricata's attack detection accuracy, and values of the HF metric associated with each operating point, including values of its $HF_o$ and $HF_u$ components.

Since the value of the HF metric for $O_1$ ($0.195 \times 10^{-3}$) is lower than that for $O_2$ ($0.184 \times 10^{-2}$), one can conclude that the SUT performs better when configured at the $O_1$ operating point (see Section IV-C). This is because when the SUT was configured at the $O_1$ operating point, Suricata spent 120 seconds in baseline state; to the contrary, at the $O_2$ operating point, Suricata spent 120 seconds in overprovisioned state issuing false positive and false negative alerts and no time in baseline state. This results in lower value of $HF_o$ for $O_1$ (0) than that for $O_2$ ($0.17 \times 10^{-2}$). The values of the HF metric match the expected behavior of the metric (see criterion $C_1$, Section IV-A). This shows the practical usefulness of the metric and measurement methodology we propose.

We now quantify the impact of CPU hotplugging speed on IDS attack detection accuracy. We allocated two virtual CPUs of 2.6 GHz to the VM where Suricata was deployed. We replayed over 240 seconds, at the speed of 150 Mbps, a trace file from the DARPA datasets.[2] We considered three separate experimental scenarios, where we hotplug one additional virtual CPU at the 80th, 120th, and 160th second of the experiment; that is, we consider three operating points of the SUT, $O_3$, $O_4$, and $O_5$, respectively.

The baseline state of Suricata is at 3 CPUs allocated to the VM where the IDS was deployed ($\alpha_b = 0.182 \times 10^{-4}$, $(1-\beta)_b = 1.0$). This results in $T_o = 0, T_b = 160, T_u = 80$ for operating point $O_1$, $T_o = 0, T_b = 120, T_u = 120$ for operating point $O_2$, and $T_o = 0, T_b = 80, T_u = 160$ for operating point $O_3$. We calculated values of the HF metric such that we enabled penalization of overprovisioning and penalization, or rewarding, of underprovisioning (i.e., $s_o, s_u = 1$).

In Table II, section 'CPU hotplugging speed', where we present the operating points $O_3$, $O_4$, and $O_5$, one can observe that the value of the HF metric (i.e., of its $HF_u$ component)

TABLE II: Hypervisor configurations: Operating points and associated values of the HF metric [$B = 0.025$; operating points are presented in the form $O(I \rightarrow (\alpha_b, 1-\beta_b); H[T_o, T_b, T_u]) \rightarrow (\alpha, 1-\beta)$; $\alpha_b$ and $1-\beta_b$ are the false and true positive rate exhibited by the IDS when it operates in baseline state given a configuration of the IDS $I$; $T_o$, $T_b$, $T_u$ are the times the IDS has spent in overprovisioned, baseline, and underprovisioned state given a configured resource provisioning policy $H$; $\alpha$ and $1-\beta$ are the false and true positive rate exhibited by the IDS for a configuration of the IDS $I$ and configured resource provisioning policy $H$ (see Definition 1)]

| Operating point | Impact on attack detection accuracy | | Metric values | | |
|---|---|---|---|---|---|
| | $\lvert\Delta(1-\alpha)\rvert$ | $\lvert\Delta\beta\rvert$ | $HF_o$ | $HF_u$ | $HF$ |
| **CPU hotplugging intensity** | | | | | |
| $O_1(I \rightarrow (0.182 \times 10^{-4}, 1.0); H[0, 120, 120]) \rightarrow (0.669 \times 10^{-6}, 0.873)$ | $0.175 \times 10^{-4}$ | 0.126 | 0 | $0.195 \times 10^{-3}$ | $0.195 \times 10^{-3}$ |
| $O_2(I \rightarrow (0.182 \times 10^{-4}, 1.0); H[120, 0, 120]) \rightarrow (0.669 \times 10^{-6}, 0.873)$ | $0.175 \times 10^{-4}$ | 0.126 | $0.17 \times 10^{-2}$ | $0.195 \times 10^{-3}$ | $0.184 \times 10^{-2}$ |
| **CPU hotplugging speed** | | | | | |
| $O_3(I \rightarrow (0.182 \times 10^{-4}, 1.0); H[0, 160, 80]) \rightarrow (0.707 \times 10^{-5}, 0.934)$ | $0.111 \times 10^{-4}$ | 0.065 | 0 | $0.328 \times 10^{-4}$ | $0.328 \times 10^{-4}$ |
| $O_4(I \rightarrow (0.182 \times 10^{-4}, 1.0); H[0, 120, 120]) \rightarrow (0.669 \times 10^{-6}, 0.873)$ | $0.175 \times 10^{-4}$ | 0.126 | 0 | $0.195 \times 10^{-3}$ | $0.195 \times 10^{-3}$ |
| $O_5(I \rightarrow (0.182 \times 10^{-4}, 1.0); H[0, 80, 160]) \rightarrow (0.573 \times 10^{-6}, 0.841)$ | $0.176 \times 10^{-4}$ | 0.158 | 0 | $0.416 \times 10^{-3}$ | $0.416 \times 10^{-3}$ |
| **Memory hotplugging intensity** | | | | | |
| $O_6(I \rightarrow (0.189 \times 10^{-4}, 1.0); H[0, 120, 120]) \rightarrow (0.119 \times 10^{-4}, 0.973)$ | $0.697 \times 10^{-5}$ | 0.026 | 0 | $0.574 \times 10^{-5}$ | $0.574 \times 10^{-5}$ |
| $O_7(I \rightarrow (0.189 \times 10^{-4}, 1.0); H[120, 0, 120]) \rightarrow (0.119 \times 10^{-4}, 0.973)$ | $0.697 \times 10^{-5}$ | 0.026 | $0.346 \times 10^{-3}$ | $0.574 \times 10^{-5}$ | $0.369 \times 10^{-3}$ |
| **Memory hotplugging speed** | | | | | |
| $O_8(I \rightarrow (0.189 \times 10^{-4}, 1.0); H[120, 0, 120]) \rightarrow (0.119 \times 10^{-4}, 0.973)$ | $0.697 \times 10^{-5}$ | 0.026 | $0.346 \times 10^{-3}$ | $0.574 \times 10^{-5}$ | $0.369 \times 10^{-3}$ |
| $O_9(I \rightarrow (0.189 \times 10^{-4}, 1.0); H[80, 0, 160]) \rightarrow (0, 0.785)$ | $0.189 \times 10^{-4}$ | 0.214 | $0.18 \times 10^{-2}$ | $0.768 \times 10^{-3}$ | $0.26 \times 10^{-2}$ |

increases as the time at which a CPU was provisioned to the VM where Suricata was deployed increases. The later a CPU was provisioned, the more time the IDS has spent in underprovisioned state, and the bigger is the impact of underprovisioning on the true positive rate exhibited by Suricata (see column '$|\Delta\beta|$'). Therefore, the HF metric penalizes the SUT the most when it is configured at the $O_5$ operating point. This matches the expected behavior of the metric (see criterion $C_2$, Section IV-A).

*b) Memory hotplugging intensity and speed:* We first quantify the impact of memory hotplugging intensity on IDS attack detecting accuracy. We deployed Snort 2.9.8.0 in a paravirtualized VM running on top of the Xen 4.4.1 hypervisor. We allocated four virtual CPUs of 2.6 GHz and a NIC with a maximal data transfer rate of 1 Gbit/second to the VM where Snort was deployed. We also allocated 1240 MB of main memory to this VM so that the VM is under memory pressure when workloads are run. This enabled us to observe the impact of memory hotplugging on IDS attack detection accuracy in scenarios where such a hotplugging is normally performed. We replayed over 240 seconds, at the speed of 150 Mbps, a trace file from the DARPA datasets.[2] All configuration options of Snort were set to their default values.

We considered two separate experimental scenarios, where we hotplug additional 260 and 560 MB of main memory on the VM where Snort was deployed, at the 120th second of each experiment; that is, we consider two operating points of the SUT, $O_6$ and $O_7$, respectively.

We first identified the baseline state of Snort at 1500 MB of memory allocated to the VM where the IDS was deployed. This results in $T_o = 0, T_b = 120, T_u = 120$ for operating point $O_6$, and $T_o = 120, T_b = 0, T_u = 120$ for operating point $O_7$. We measured $\alpha_b$ of $0.189 \times 10^{-4}$ and $(1-\beta)_b$ of 1.0. We then calculated values of the HF metric such that we enabled penalization of overprovisioning and penalization, or rewarding, of underprovisioning (i.e., $s_o, s_u = 1$).

In Table II, section 'Memory hotplugging intensity', we present the operating points $O_6$ and $O_7$, the impact that underprovisioning has had on the attack detection accuracy exhibited by Snort, and values of the HF metric associated with each operating point, including values of its $HF_o$ and $HF_u$ components. As expected, the SUT performs better when configured at $O_6$ since at this point, in contrast to when the SUT was configured at the $O_7$ operating point, the IDS did not spent time in overprovisioned state issuing false positive and negative alerts (i.e., $HF_o = 0$, see criterion $C_1$, Section IV-A).

We now quantify the impact of memory hotplugging speed on IDS attack detection accuracy. We allocated 1240 MB of memory to the VM where Snort was deployed. We replayed over 240 seconds, at the speed of 150 Mbps, a trace file from the DARPA datasets.[2] We considered two separate experimental scenarios, where we hotplug additional 560 MB of memory on the VM where Snort was deployed, at the 120th and 160th second of the experiment; that is, we consider two operating points of the SUT, $O_8$ and $O_9$, respectively.

The baseline state of Snort is at 1500 MB of memory allocated to the VM where the IDS was deployed ($\alpha_b = 0.189 \times 10^{-4}$, $(1-\beta)_b = 1.0$). This results in $T_o = 120, T_b = 0, T_u = 120$ for operating point $O_8$, and $T_o = 80, T_b = 0, T_u = 160$ for operating point $O_9$. We calculated values of the HF metric such that we enabled penalization of overprovisioning and penalization, or rewarding, of underprovisioning

(i.e., $s_o, s_u = 1$).

In Table II, section 'Memory hotplugging speed', where we present the operating points $O_8$ and $O_9$, one can observe that the SUT performs better when configured at the $O_8$ operating point — the value of the HF metric is lower than that for the $O_9$ operating point. This is primarily because, at the $O_9$ operating point, underprovisioning of memory has caused a much more significant reduction of the true positive rate exhibited by Snort ($|\Delta\beta| = 0.214$, $HF_u = 0.768 \times 10^{-3}$) in contrast to when the SUT was configured at the $O_8$ operating point ($|\Delta\beta| = 0.026$, $HF_u = 0.574 \times 10^{-5}$, see criterion $C_2$, Section IV-A). Note that at the $O_9$ operating point, underprovisioning has also caused a reduction of issued false positives to 0 (see column '$|\Delta(1-\alpha)|$', $|\Delta(1-\alpha)| = \alpha_b$), which, although considered a positive phenomenon rewarded by the HF metric (see criterion $C_3$, Section IV-A), does not outweigh the previously mentioned penalization of the reduction of the true positive rate.

### B. IDS configurations

We now consider various IDS configurations for a given hypervisor configuration. Varying configurations of an IDS under test in order to identify the optimal operating point of the IDS is a common practice (see Section II). We demonstrate here the use of the HF metric for identifying an optimal operating point of an IDS that is part of an SUT as we define it. In addition, we show how the HF metric complements conventional IDS evaluation metrics, that is, an ROC curve, which is the most commonly used conventional metric.

We deployed Snort 2.9.8.0 in a paravirtualized VM running on top of the Xen 4.4.1 hypervisor. We allocated two virtual CPUs of 2.6 GHz, 12 GB of main memory, and a NIC with a maximal data transfer rate of 1 Gbit/second to the VM where Snort was deployed. We replayed over 240 seconds, at the speed of 150 Mbps, a trace file from the DARPA datasets.[2]

We considered six separate experimental scenarios, where we vary the configuration of Snort; that is, we consider six operating points $O_{1,2,..6}$. We observed that Snort's rule with ID 1417 led to mislabeling many benign SNMP (Simple Network Management Protocol) packets as malicious. Therefore, we examined the influence of the configuration parameter *threshold* on the attack detection accuracy of Snort. The parameter *threshold* is used for reducing the number of false alerts by suppressing rules that often mislabel benign activities as malicious. A rule may be suppressed such that it is configured to not generate an alert for a specific number of times (specified with the keyword *count*) during a given time interval (specified with the keyword *seconds*).

We considered five different configurations of Snort where the rule with ID 1417 was suppressed by setting the value of *count* to 2, 3, 4, 5, and 6, whereas *seconds* was set to 120. We also considered the default configuration of Snort, according to which the signature with ID 1417 is not suppressed. We configured a resource provisioning policy such that the hypervisor provisions 2 additional CPUs at the 120th second of each experiment.

We first identified the baseline states of Snort and calculated values of $\alpha_b$ and $(1-\beta)_b$ for each considered configuration of the IDS. The baseline state of Snort for all configurations was at 3 CPUs allocated to the VM where the IDS was deployed. This results in $T_o = 120, T_b = 0, T_u = 120$ for all operating points. We then calculated values of the HF metric such that we disabled penalization of overprovisioning and enabled penalization, or rewarding, of underprovisioning (i.e., $s_o = 0, s_u = 1$, $HF_o = 0$); that is, the goal of this study is to identify the optimal configuration of Snort for which underprovisioning is most beneficial.

In Table III, we present the operating points $O_{1,2,..6}$, the impact that underprovisioning has had on the attack detection accuracy exhibited by Snort, and values of the HF metric associated with each operating point. One can observe that underprovisioning has been beneficial in all considered scenarios since it has caused significant reductions of the number of false positives issued by Snort (see column '$|\Delta(1-\alpha)|$' and the negative values of the HF metric).[3] Underprovisioning has been most beneficial when the SUT was configured at the $O_5$ operating point ($HF = -0.446 \times 10^{-4}$, *count*=2; $-0.446 \times 10^{-4}$ is the smallest value of all values of the HF metric presented in Table III). Therefore, one may conclude that the optimal operating point of Snort, such that underprovisioning is most beneficial, is when the IDS was configured such that *count* is set to 2.

*a) The HF metric and ROC curves:* In the area of IDS evaluation, it is a common practice to use ROC curves for analyzing the relationship between the true positive $(1-\beta)$ and false positive rate $(\alpha)$ exhibited by an IDS under test (see Section II). The HF metric expresses $1-\beta$ and $\alpha$ (see Equation 1). Therefore, values of the HF metric can be associated with each IDS operating point depicted using an ROC curve. Thus, evaluation of an IDS with respect to values of the HF metric can be combined with ROC curve analysis. This allows for a visual and comprehensive overview of the attack detection accuracy of the IDS and the impact that the underlying hypervisor has had on the IDS's accuracy.

In Figure 4, we depict an ROC curve expressing the relationship between $\alpha$ and $1-\beta$ exhibited by Snort for the considered configurations of the IDS (see Table III). The ROC curve is annotated with the values of the HF metric (in square brackets) associated with the IDS configuration points. This allows for a visual, straightforward identification of the IDS operating point such that the SUT, which includes the underlying hypervisor, performs optimally (i.e., the IDS operating point with the lowest value of the HF metric associated with it, marked using $\square$ in Figure 4). We note that an IDS evaluator, based on subjective criteria, may consider another IDS operating point optimal (e.g., the one at which Snort exhibits the highest true positive rate). However, in such a case, the evaluator would still have insight into the impact of the hypervisor on

[3]Negative values of the HF metric indicate that an SUT is rewarded for underprovisioning that has caused significant reduction of the false positive rate exhibited by the IDS, which outweighs other behaviors of the SUT that the HF metric penalizes (see Section IV-A).

TABLE III: IDS configurations: Operating points and associated values of the HF metric [see caption of Table II for a description of the form in which operating points are presented]

| IDS configuration [*seconds* = 120] | Operating point | Impact on attack detection accuracy | | Metric values |
|---|---|---|---|---|
| | | $\lvert\Delta(1-\alpha)\rvert$ | $\lvert\Delta\beta\rvert$ | $HF$ |
| *count* = 6 | $O_1(I \rightarrow (0.08 \times 10^{-2}, 0.333); H[120,0,120]) \rightarrow (0.072 \times 10^{-2}, 0.332)$ | $0.753 \times 10^{-4}$ | $0.268 \times 10^{-4}$ | $-0.364 \times 10^{-4}$ |
| *count* = 5 | $O_2(I \rightarrow (0.11 \times 10^{-2}, 0.416); H[120,0,120]) \rightarrow (0.102 \times 10^{-2}, 0.415)$ | $0.788 \times 10^{-4}$ | $0.572 \times 10^{-4}$ | $-0.379 \times 10^{-4}$ |
| *count* = 4 | $O_3(I \rightarrow (0.13 \times 10^{-2}, 0.5); H[120,0,120]) \rightarrow (0.122 \times 10^{-2}, 0.499)$ | $0.733 \times 10^{-4}$ | $0.999 \times 10^{-4}$ | $-0.35 \times 10^{-4}$ |
| *count* = 3 | $O_4(I \rightarrow (0.17 \times 10^{-2}, 0.624); H[120,0,120]) \rightarrow (0.168 \times 10^{-2}, 0.623)$ | $0.1 \times 10^{-4}$ | $0.879 \times 10^{-3}$ | $-0.738 \times 10^{-6}$ |
| *count* = 2 | $O_5(I \rightarrow (0.24 \times 10^{-2}, 0.833); H[120,0,120]) \rightarrow (0.23 \times 10^{-2}, 0.832)$ | $0.952 \times 10^{-4}$ | $0.795 \times 10^{-3}$ | $-0.446 \times 10^{-4}$ |
| Default configuration | $O_6(I \rightarrow (0.26 \times 10^{-2}, 0.958); H[120,0,120]) \rightarrow (0.251 \times 10^{-2}, 0.957)$ | $0.805 \times 10^{-4}$ | $0.297 \times 10^{-3}$ | $-0.39 \times 10^{-4}$ |

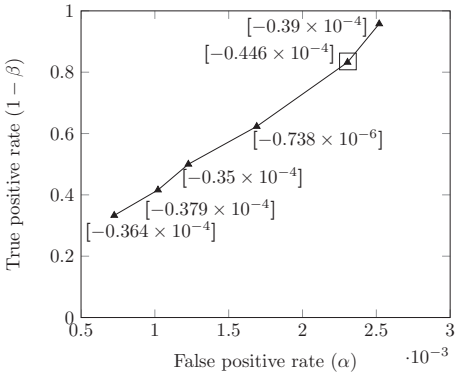


Fig. 4: An ROC curve and values of the HF metric associated with each IDS operating point

the attack detection accuracy of the IDS when configured at this operating point because of the value of the HF metric associated with it.

Multiple ROC curves, each depicting $\alpha$ and $1-\beta$ exhibited by a single IDS, may be used for comparing multiple IDSes. For example, an IDS may be considered better than the other IDSes if it features higher $1-\beta$ at all operating points along the ROC curve [13]. However, such an analysis may be misleading if the ROC curves intersect. In such a case, if the compared IDSes are deployed in virtualized environments featuring elasticity, the IDSes may be compared in a straightforward manner based on values of the HF metric; that is, the IDS that performs optimally considering the impact of the hypervisor on its attack detection accuracy, is the IDS with the lowest value of the HF metric associated with its optimal operating point (see criterion $C_6$, Section IV-C).

## VI. Conclusion

A virtualized environment may be elastic, that is, virtualized resources may be allocated to, or deallocated from, VMs during operation by the hypervisor applying a given resource provisioning policy. Elasticity might have significant impact on the attack detection accuracy exhibited by an IDS deployed in a virtualized environment featuring elasticity (e.g., an IDS deployed as a VNF). Conventional metrics for quantifying IDS attack detection accuracy (i.e., IDS evaluation metrics) do not express this impact, which might lead to inaccurate measurements.

In this paper, we first provided an overview of conventional IDS evaluation metrics. We demonstrated the impact that elasticity may have on IDS attack detection accuracy and we showed how the use of conventional IDS evaluation metrics may lead to practically challenging and inaccurate measurements. We then proposed a novel metric (i.e., the HF metric) and measurement methodology, which take elasticity into account. We designed the metric with respect to a set of criteria for the accurate and practically useful evaluation. For example, the HF metric penalizes resource underprovisioning causing the reduction of the true positive rate exhibited by a given IDS. We demonstrated the practical use of the HF metric through a set of case studies.

Our work can be continued in several directions:

*(i)* We plan to extend the measurement methodology we proposed so that scenarios where the hypervisor allocates or deallocates different types of resources (e.g., both processing and memory resources) may be considered when determining the baseline state of an IDS (see Section IV-B1);

*(ii)* We plan to extend the HF metric so that penalization of overprovisioning and rewarding, or penalization, of underprovisioning can be scaled up or down by a factor configured by a user of the metric. This allows for further customization of the HF metric and its output with respect to user requirements;

*(iii)* We plan to conduct a set of experiments involving a variety of IDSes and hypervisors applying various resource provisioning policies using the HF metric and the measurement methodology we proposed. Among other scenarios, we will consider scenarios involving IDSes that have adaptive characteristics leading to frequently changing resource requirements, such as WIND (Workload-aware INtrusion Detection) [14].

We stress that accurate metrics for quantifying attack detection accuracy of IDSes are essential not only to evaluate specific IDSes, but also as a driver of innovation in the field of intrusion detection by enabling the identification of issues and the improvement of existing intrusion detection techniques and systems.

## Acknowledgment

This research has been supported by the Research Group of the Standard Performance Evaluation Corporation (SPEC; http://www.spec.org, http://research.spec.org).

## REFERENCES


[1] K. Scarfone and P. Mell, "Guide to Intrusion Detection and Prevention Systems (IDPS)," 2007, NIST Special Publication 900-94.

[2] F. Lombardi and R. Di Pietro, "Secure virtualization for cloud computing," *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1113–1122, July 2011.

[3] VMWare Inc., *VMWare vShield Endpoint*, 2015, http://www.vmware.com/ca/en/products/vsphere/features/endpoint.

[4] M. Roesch, "Snort - Lightweight Intrusion Detection for Networks," in *Proceedings of the 13th USENIX conference on System Administration (LISA)*. USENIX Association, 1999, pp. 229–238.

[5] Open Internet Security Foundation (OISF), *Suricata Intrusion Detection System*, 2015, http://suricata-ids.org/.

[6] P. Mell, V. Hu, R. Lippmann, J. Haines, and M. Zissman, "An Overview of Issues in Testing Intrusion Detection Systems," 2003.

[7] S. Spinner, N. Herbst, S. Kounev, X. Zhu, L. Lu, M. Uysal, and R. Griffith, "Proactive Memory Scaling of Virtualized Applications," in *Proceedings of the 2015 IEEE 8th International Conference on Cloud Computing (IEEE CLOUD 2015)*. IEEE, June 2015, pp. 277–284.

[8] R. A. Maxion and R. R. Roberts, "Proper Use of ROC Curves in Intrusion/Anomaly detection," School of Computing Science, University of Newcastle upon Tyne, Tech. Rep. CS-TR-871, November 2004.

[9] S. Axelsson, "The base-rate fallacy and its implications for the difficulty of intrusion detection," *ACM Transactions on Information and Systems Security*, vol. 3, no. 3, pp. 186–205, August 2000.

[10] J. E. Gaffney and J. W. Ulvila, "Evaluation of intrusion detectors: a decision theory approach," in *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, 2001, pp. 50–61.

[11] G. Gu, P. Fogla, D. Dagon, W. Lee, and B. Skorić, "Measuring intrusion detection capability: an information-theoretic approach," in *Proceedings of the 2006 ACM Symposium on Information, computer and communications security (ASIACCS)*. New York, NY, USA: ACM, 2006, pp. 90–101.

[12] M. Hall and K. Wiley, "Capacity verification for high speed network intrusion detection systems," in *Proceedings of the 5th International Conference on Recent Advances in Intrusion Detection*. Berlin, Heidelberg: Springer-Verlag, 2002, pp. 239–251.

[13] A. Milenkoski, M. Vieira, S. Kounev, A. Avritzer, and B. D. Payne, "Evaluating Computer Intrusion Detection Systems: A Survey of Common Practices," *ACM Comput. Surv.*, vol. 48, no. 1, pp. 12:1–12:41, Sep 2015.

[14] S. Sinha, F. Jahanian, and J. M. Patel, "WIND: Workload-Aware INtrusion Detection," in *Recent Advances in Intrusion Detection*, ser. Lecture Notes in Computer Science, D. Zamboni and C. Kruegel, Eds. Springer, 2006, vol. 4219, pp. 290–310.