Chapter 22
Benchmarking Intrusion Detection Systems with Adaptive Provisioning of Virtualized Resources

Aleksandar Milenkoski and K. R. Jayaram and Samuel Kounev

Abstract With the increasing popularity of virtualization, deploying intrusion detection systems (IDSes) in virtualized environments, for example, in virtual machines as virtualized network functions, has become an emerging practice. Modern virtualized environments feature on-demand provisioning of virtualized processing and memory resources to virtual machines, dynamically adapting its intensity in order to meet resource demands. Such a provisioning may have a significant impact on many properties of an IDS deployed in a virtual machine, for example, on its attack detection accuracy. However, conventional metrics for quantifying IDS attack detection accuracy do not capture this impact, which may lead to inaccurate assessments of the IDS's accuracy at detecting attacks. In this chapter, we discuss in detail on the impact of on-demand provisioning of virtualized resources on IDS attack detection accuracy. Further, we discuss on relevant issues related to the use of conventional metrics for quantifying IDS attack detection accuracy. Finally, we present a preliminary metric and measurement methodology, which allow for the accurate assessment of IDS attack detection accuracy taking on-demand resource provisioning into account.

Aleksandar Milenkoski
University of Würzburg, Am Hubland, 97074 Würzburg, Germany e-mail: milenkoski@acm.org

K. R. Jayaram
Thomas J. Watson Research Center, Yorktown Heights, NY USA e-mail: jayaramkr@us.ibm.com

Samuel Kounev
University of Würzburg, Am Hubland, 97074 Würzburg, Germany e-mail: skounev@acm.org

22.1 Introduction

In recent years, virtualization has received increasing interest, both from industry and academia, as a way to reduce costs through server consolidation and to enhance the flexibility of physical infrastructures. In a virtualized system, governed by a hypervisor, resources such as processor time, disk capacity, and network bandwidth, are shared among virtual machines (VMs). Each VM accesses physical resources through the hypervisor and is entitled to a predefined fraction of capacity.

While server consolidation through virtualization provides many benefits, it also introduces some new challenges. For example, the increased dynamics and flexibility of virtualized systems increase the need for automated resource management mechanisms to guarantee system stability and adequate quality-of-service [8], [9]. Further, the introduction of a hypervisor and the allocation of multiple VMs on a single physical server are additional critical aspects introducing new potential threats and vulnerabilities [11], [17]. For instance, Gens et al. [4] report that security is a major concern for users of modern virtualized service infrastructures, followed by availability and performance. Some critical security issues include data integrity, authentication, application security, and so on.

Intrusion detection is a common security mechanism for detecting malicious activities (attacks) in host and/or network environments. The accurate attack detection brings multiple benefits, for example, it allows for timely reaction in order to stop an on-going attack. This is crucial for mission-critical systems with high integrity and availability requirements, such as self-aware systems featuring self-protection (see Chapter 14).

The National Institute of Standards and Technology (NIST) defines intrusion detection as "*the process of monitoring the events occurring in a computer system or network and analyzing them for signs of possible incidents, which are violations or imminent threats of violation of computer security policies, acceptable use policies, or standard security practices*" [19]. Given this definition, under intrusion detection system (IDS), we understand the software that automates the intrusion detection process. The research and industrial communities have designed and developed many intrusion detection systems (IDSes), for example, the community-driven Snort [18] by Sourcefire and the commercial ISS (Internet Security Systems) by IBM,[1] which use a diverse set of intrusion detection techniques.

The wide adoption of virtualization technology has lead to the emergence of novel IDSes specifically designed to operate in virtualized environments, such as ACPS (Advanced Cloud Protection System) [10], Invincea,[2] Juniper Firefly Host,[3] and vShield Endpoint.[4] Many of these IDSes have components both inside the hypervisor and in a designated, secure VM (normally the host VM), which has several

[1] http://www-935.ibm.com/services/in/en/it-services/intrusion-detection.html

[2] http://www.invincea.com/

[3] http://www.juniper.net/techpubs/en_US/firefly6.0/information-products/pathway-pages/security-virtual-host-product-family-index.html

[4] https://www.vmware.com/de/products/vsphere/features/endpoint

benefits. For instance, they can monitor the network and/or host activities of all colocated VMs and are isolated from malicious VMs' users since they do not operate inside the VMs, but leverage functionalities of the underlying hypervisor.

The adoption of virtualization technology has also lead to the emergence of the practice of deploying conventional IDSes (e.g., hardware IDS appliances or common software-based IDSes) as virtualized network functions (VNFs). For instance, the network-based IDS Snort [18] may be deployed in a designated VM and configured to tap into the physical network interface card used by all VMs. Thus, the IDS can monitor the network activities of all VMs at the same time while being isolated from, and transparent to, their users. Further, in comparison to deploying hardware IDS appliances, which are expensive and challenging to manage, deploying IDSes as VNFs is cost-effective and makes their management an easier task.

With the increasing complexity of IDSes, the development of methodologies, techniques, and tools for evaluating IDSes has become an important research topic. 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. This enables comparison of the evaluation results with respect to the configuration space of the IDS, which can help to identify an optimal configuration. In Chapter 14, we discuss in detail on the importance of IDS evaluation in the context of self-aware systems that feature self-protection.

Any IDS evaluation experiment requires careful planning including selection of (i) workloads, which are used for exercising the sensors of an IDS under test, and (ii) metrics and measurement methodologies, which are used for quantifying IDS properties (e.g., attack detection accuracy). Workloads, metrics, and measurement methodologies are considered as the standard components of any evaluation experiment.

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 resources (i.e., CPU, memory, or network resources) to VMs, whose intensity dynamically adapts with respect to changes in the intensity of the workloads that the VMs process. For instance, the Xen and VMware VSphere hypervisors allow for on-demand hotplugging virtual CPUs and memory on VMs.[5]

On-demand provisioning of virtualized resources to VMs is also known as *vertical VM scaling*, a topic that has received a considerable amount of attention. Researchers, such as Spinner et al. [21], Xu et al. [22], and Dawoud et al. [2], have developed approaches for fine controlled provisioning of virtualized resources for

[5] See, for example, https://pubs.vmware.com/vsphere-60/index.jsp?topic=%2Fcom.vmware.vsphere.hostclient.doc%2FGUID-F102B9BD-1B92-4AC5-ADC0-BE4E90473C5F.html.

the purpose of optimising benefits gained, improving performance isolation between VMs, and so on.

Virtualized resources may be hotplugged on a VM where an IDS operates. This implies that resources can be provisioned and used by an IDS deployed in a virtualized environment during operation, which may have a significant impact on many properties of the IDS, including its attack detection accuracy. Thus, we argue that the use of existing IDS evaluation metrics, which do not take elasticity into account, for evaluating IDSes deployed in virtualized environments may lead to inaccurate measurements.

In this chapter, we first systematize and review commonly used IDS evaluation metrics. We then discuss and show through case studies how elasticity of virtualized environments affects IDS attack detection accurracy. Further, we discuss issues related to the use of conventional IDS evaluation metrics for evaluating IDSes deployed in elastic virtualized environments. Finally, we propose a preliminary metric and measurement methodology, which take elasticity into account.

This chapter is organized as follows: In Section 22.2, we survey existing IDS evaluation metrics; in Section 22.3, we discuss and demonstrate through case studies relevant issues related to the use of these metrics; in Section 22.4, we present our preliminary ideas on a metric and measurement methodology that take elasticity into account; in Section 22.5, we discuss future work and conclude this chapter.

22.2 An Overview of IDS Evaluation Metrics

In this section, we provide a compact overview of commonly used IDS evaluation metrics. In Chapter 14, we provide related contents focussing on IDS evaluation metrics relevant when it comes to evaluating IDSes employed as part of self-aware systems featuring self-protection.

We distinquish between two types of metrics: (i) performance-related metrics, and (ii) security-related metrics (see Chapter 14). By performance-related metrics, we mean metrics that quantify non-functional properties of a tested IDS, such as capacity, performance overhead, resource consumption, and similar. The metrics that quantify these properties, such as processing throughput and CPU utilization, are typical for traditional performance evaluation suites. The practice in the area of IDS evaluation has shown that performance-related metrics are also applicable for evaluating IDSes. For instance, Meng et al. [14] measure workload processing throughput, Lombardi et al. [10] measure performance overhead, Mohammed et al. [16] measure power consumption of a distributed IDS, and Sinha et al. [20] measure memory consumption. We focus here on security-related metrics, not elaborating further on the performance-related ones.

22.2.1 Security-related metrics

By security-related metrics, we mean metrics that quantify security-relevant properties of an IDS under test. We focus here on metrics for quantifying the attack detection accuracy of an IDS, where attack detection is considered as having a binary output - attack/no attack - since this is the most typical case and common practice in IDS evaluation. When it comes to attack detection in general, an IDS may also be evaluated with respect to other features (e.g., evaluating provided additional information about the detected attack attempts). We refer the reader to the evaluation methodology specification of NSS Labs for further information on how other IDS attack detection features may be evaluated.[6]

We distinguish between basic and composite security-related metrics (see Chapter 14). We provide an overview of these metrics in Table 22.1. In Table 22.1, we also show the notation of used symbols (including variables).

22.2.1.1 Basic metrics

The basic metrics are most common and they quantify various individual attack detection properties. Although they are quantified individually, these properties need to be analyzed together in order to accurately characterize the attack detection efficiency of a given IDS. The true positive rate $1-\beta = P(A|I)$ quantifies the probability that an alert generated by an IDS is really an intrusion. 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. The respective complementary metrics, i.e., the true negative rate $1-\alpha = P(\neg A|I)$ and the false negative rate $\beta = P(\neg A|\neg I)$, are also relevant. We do not list these metrics in Table 22.1 since they are simply arithmetically related to the true and false positive rate.

The positive predictive value (PPV) quantifies the probability that there is an intrusion when an IDS generates an alert whereas the the negative predictive value (NPV) quantifies the probability that there is no intrusion when an IDS does not generate an alert. These metrics are normally calculated once one has already calculated $P(A|I)$, $P(A|\neg I)$, $P(\neg A|\neg I)$ and $P(\neg A|I)$ by using the Bayesian theorem for calculating a conditional probability (see Table 22.1). Thus, PPV and NPV are also known as Bayesian positive detection rate and Bayesian negative detection rate, respectively.

The basic metrics listed in Table 22.1 originate from signal detection theory. Hancock and Wintz [7] describe the use of these metrics in the area of signal detection. In this paper, we denote these metrics as security-related only because we refer to them in the context of detecting attacks against computer systems and/or networks in particular.

---

[6] http://www.nsslabs.com/sites/default/files/import/assets/Methodologies/NSS_Labs_IPS%20Group%20Test%20Methodology%20v6.1.pdf

Table 22.1: Metrics for quantifying IDS attack detection accuracy

Metric	**Annotation/Formula**
	Basic metrics
True positive rate	$1-\beta = 1-P(\neg A\|I) = P(A\|I)$
False positive rate	$\alpha = P(A\|\neg I)$
Positive predictive value	$P(I\|A) = \frac{P(I)P(A\|I)}{P(I)P(A\|I)+P(\neg I)P(A\|\neg I)}$
Negative predictive value	$P(\neg I\|\neg A) = \frac{P(\neg I)P(\neg A\|\neg I)}{P(\neg I)P(\neg A\|\neg I)+P(I)P(\neg A\|I)}$
	Composite metrics
Expected cost	$C_{exp} = Min(C\beta B,(1-\alpha)(1-B))+Min(C(1-\beta)B,\alpha(1-B))$
Intrusion detection capability	$C_{ID} = \frac{I(X;Y)}{H(X)}$

Notations of used symbols

Symbol	**Meaning**
A	*Alert event*: An IDS generates an attack alert
I	*Intrusion event*: An attack is performed
C_α	Cost of an IDS generating an alert when an intrusion has not occured
C_β	Cost of an IDS failing to detect an intrusion
C	*Cost ratio*: The ratio between the costs C_α and C_β
$B = P(I)$	*Base rate*: Prior probability that an intrusion event occurs
X	*IDS input*: Discrete random variable used to model input to an IDS such that $X = 0$ represents a benign activity and $X = 1$ represents a malicious activity (i.e., an intrusion)
Y	*IDS output*: Discrete random variable used to model the generation of alerts by an IDS such that $Y = 0$ represents no alert and $Y = 1$ represents an alert
$H(X)$	*Uncertainty of* X: Entropy measure quantifying the uncertainty of the IDS input X
$I(X;Y)$	*Mutual information*: The amount of information shared between the random variables X and Y, i.e., the amount of reduction of the uncertainty of the IDS input (X) after the IDS output (Y) is known

22.2.1.2 Composite metrics

IDS evaluators often combine the basic metrics in order to analyze relationships between them, for example, to discover 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 measured true positive and false positive detection rate of an IDS. A ROC curve plots true positive rate against the corresponding false positive rate [12]; that is, a ROC curve depicts multiple IDS operating points of an IDS under test and, as such, it is useful for identifying an optimal operating point or for comparing multiple IDSes.

Security researchers have proposed metrics that are more expressive than ROC curves. One of the most prominent metrics that belong to this category are the expected cost metric (C_{exp}) proposed by Gaffney et al. [3] and the intrusion detection capability metric (C_{ID}) proposed by Gu et al. [5]. We discuss in detail the expected-cost metric in Chapter 14. We focus here on the intrusion detection capability metric.

Intrusion detection capability

Gu et al. [5] propose a metric called intrusion detection capability (denoted by C_{ID}, see Table 22.1). They model the input to an IDS as a stream of a random variable X ($X = 1$ denotes an intrusion, $X = 0$ denotes benign activity), and the IDS output respectively as a stream of a random variable Y ($Y = 1$ denotes IDS alert, $Y = 0$ denotes no alert). It is assumed that both the input stream and the output stream have a certain degree of uncertainty reflected by the entropies $H(X)$ and $H(Y)$, respectively. Thus, Gu et al. [5] model the number of correct guesses of an IDS, i.e., $I(X;Y)$, as mutual shared information between the random variables X and Y — $I(X;Y) = H(X)H(X|Y)$. An alternative interpretation is that the accuracy of an IDS is modeled as the reduction of the uncertainty of the IDS input, $H(X)$, after the IDS output Y is known. Finally, by normalizing the shared information $I(X;Y)$ with the entropy of the input variable $H(X)$, the intrusion detection capability metric C_{ID} is obtained. Note that C_{ID} incorporates the uncertainty of the input stream $H(X)$ (i.e., the distribution of intrusions in the IDS input) and the accuracy of an IDS under test $I(X;Y)$. Thus, one may conclude that C_{ID} incorporates the base rate B and many basic metrics, such as the true positive rate $(1-\beta)$, the false positive rate (α), and similar. For the definition of the relationship between C_{ID}, on the one hand, and B, $1-\beta$, and α, on the other hand, we refer the reader to [5]. Given this relationship, a value of C_{ID} may be assigned to any operating point of an IDS on the ROC curve. With this assignment, one obtains a new curve, i.e., a C_{ID} curve. A C_{ID} curve provides a straightforward identification of the optimal operating point of an IDS, i.e., the point that marks the highest C_{ID}.

22.2.2 Case study

We now present a case study involving the de-facto standard network-based IDS Snort [18] in order to demonstrate the use of the previously discussed metrics. We evaluate Snort 2.9.22 using a database of rules dated 11.07.2013. We deployed Snort in a host with a dual-core CPU, each core operating at the speed of 2 GHz, 3 GB of memory, and a Debian 7.0 OS. We use the DARPA (Defense Advanced Research Projects Agency) datasets as workloads, which have been recorded over several weeks in 1998. We note that at the time of writing, the accuracy of Snort in detecting the attacks recorded in the DARPA datasets is of no practical relevance since the attacks recorded in these datasets do not (or extremely rarely) occur in cur-

rent real-world attack scenarios. However, the DARPA datasets are still the largest datasets that contain both malicious and benign activities. We replayed a trace file from the DARPA datasets that has been recorded on Monday of the first week of trace recording. In order to calculate metric values, we used the "ground truth" files provided by the Lincoln Laboratory at MIT.[7] They contain information useful for uniquely identifying each attack recorded in the trace file that we replayed, such as time of execution of the attack, IP addresses of the attacking and victim host, and the network protocol through which the attack has been carried out. To calculate values of basic and composite security-related metrics, we compared the "ground truth" information with the alerts produced by Snort. As a result, we were able to calculate the number of detected and missed attacks as well as the number of false alerts, which information is required for calculating values of basic and composite security-related metrics.

With its default configuration enabled, Snort detected almost all replayed attacks. However, Snort also issued false alerts. More specifically, Snorts rule with ID 1417 led to mislabeling many benign SNMP (Simple Network Management Protocol) packets as malicious. To investigate whether Snort can be tuned such that the number of false alerts is reduced while the number of true alerts remains sufficiently high, 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 generated by suppressing rules that often mislabel benign activities as malicious. A rule may be suppressed in a way such that it will not trigger the generation of an alert everytime it labels an activity as malicious. Similarly, a suppressed rule may be configured to not trigger the generation of an alert for a specific number of times (specified with the keyword *count*) during a given time interval (specified with the keyword *seconds*).

The measurement of the attack detection accuracy of an IDS for different configurations of the IDS enables the identification of an optimal operating point (i.e., an IDS configuration that yields optimal values of both the true and false positive detection rate). We considered 6 operating points of Snort. We measured the attack detection accuracy of Snort for 5 different configurations where the rule with ID 1417 was suppressed by setting the value of *count* to 2, 3, 4, 5, and 6, while *seconds* was set to 12026. We also measured the attack detection accuracy of Snort when its default configuration was used, according to which the rule with ID 1417 is not suppressed.

In Table 22.2, we present the values of the basic security-related metrics true positive rate $(1-\beta)$, false positive rate (α), positive predictive value (PPV), and negative predictive value (NPV). In Table 22.2, one can observe that the values of α and $1-\beta$ decrease as the value of *count* increases. This is expected since the rule with ID 1477 is suppressed more often as the value of *count* is increased.

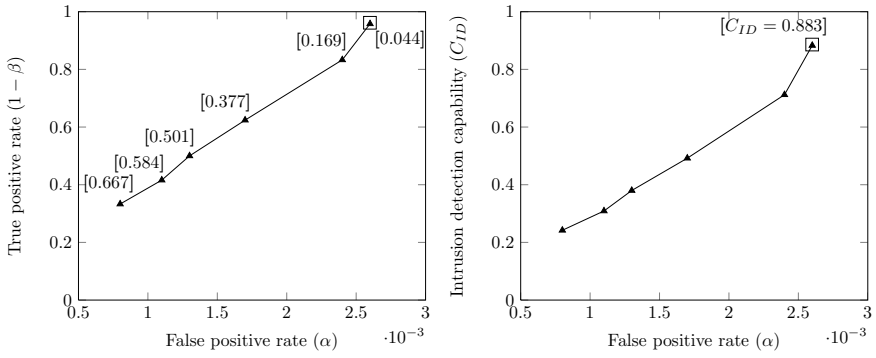
[7] The trace file we used is available at http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/1998/testing/week1/monday/tcpdump.gz. The corresponding "ground truth" data is available at http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/1998/Truth_Week_1.llist.tar.gz. To replay the trace file, we used tcpreplay [1]. We used this trace file for all experiments presented in this paper.

Table 22.2: Attack detection accuracy of Snort (`seconds=120`)

Configuration	Metric values			
	α	$1-\beta$	PPV	NPV
`count=6`	0.0008	0.333	0.9788	0.9310
`count=5`	0.0011	0.416	0.9768	0.9390
`count=4`	0.0013	0.5	0.9771	0.9473
`count=3`	0.0017	0.624	0.9761	0.9598
`count=2`	0.0024	0.833	0.9747	0.9817
Default configuration	0.0026	0.958	0.9762	0.9953

Increasing the value of *count* leads to decreasing the number of generated false alerts, which is manifested by the decreasing values of α presented in Table 22.2. However, increasing the value of *count* also leads to worsening of the true positive rate $1-\beta$. This is a typical trade-off situation between the true and the false positive rate of an IDS.

Although the basic security-related metrics are quantified individually, they need to be analyzed together in order to accurately characterize the attack detection efficiency of a given IDS. To this end, we use composite security-related metrics.



(a) (b)

Fig. 22.1: Attack detection accuracy of Snort: a) ROC curve and estimated costs associated with the depicted operating points, and b) C_{ID} curve (□ marks an optimal operating point)

In Figure 22.1a, we depict a ROC curve that provides an overview of the previously mentioned trade-off between the true and false positive rate exhibited by Snort. In addition, in Figure 22.1a, we annotate the depicted operating points with the associated estimated costs C_{exp}. The values of the estimated costs are values of the expected-cost metric proposed by Gaffney et al. [3], which we discuss in detail

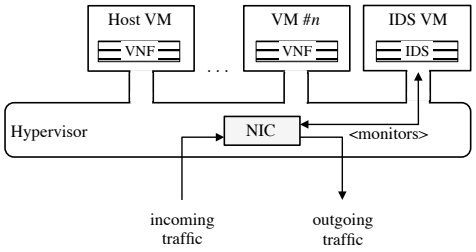


Fig. 22.2: A network-based IDS deployed as a VNF

in Chapter 14. When calculating the values of the expected cost metric we assumed a cost ratio C of 10 (i.e., the cost of not responding to an attack is 10 times higher than the cost of responding to a false alert). The base rate B is 0.10.

Once we calculated the cost associated with each operating point, we were able to identify an optimal operating point, i.e., the operating point that has the lowest C_{exp} associated with it compared to the other operating points (see Chapter 14). The optimal operating point is $(0.0026, 0.958)$, which has an estimated cost of 0.044 associated with it. Based on our findings, we conclude that Snort operates optimally in terms of cost when configured with its default settings.

In Figure 22.1b, we depict the values of the intrusion detection capability metric (C_{ID}) [5] for the different operating points of Snort. The C_{ID} curve depicted in Figure 22.1b enables the identification of an optimal operating point of Snort in terms of intrusion detection capability (i.e., the point that marks the highest C_{ID}). The optimal operating point is $(0.0026, 0.958)$, which marks a C_{ID} of 0.883. As a result, we conclude that Snort operates optimally in terms of intrusion detection capability when configured with its default settings.

22.3 Issues and Relevant Phenomena

In Section 22.1, we discussed the practice of deploying IDSes in virtualized environments; that is, we discussed IDSes specifically designed to operate in virtualized environments and conventional IDSes deployed as VNFs. In Figure 22.2, we depict the deployment of a conventional network-based IDS as a VNF. The IDS, deployed in a designated VM (*IDS VM* in Figure 22.2), taps into the physical network interface card (NIC) managed by the hypervisor in order to monitor incoming and outgoing traffic. Other VNFs (*VNF* in Figure 22.2), such as routing or firewalling, may be deployed in co-located VMs (*Host VM*, *VM #n* in Figure 22.2). In this section, we discuss relevant issues related to the use of conventional IDS evaluation metrics (see Section 22.2) for evaluating IDSes deployed in virtualized environments, for example, as depicted in Figure 22.2.

In Section 22.1, we mentioned that a common aspect of current IDS evaluation metrics is that they are defined with respect to a *fixed* set of hardware resources available to a given IDS under test. This is contrary to what elasticity of modern virtualized environments enables — flexible, on-demand provisioning of resources to VMs where IDSes may be deployed (see Figure 22.2 and Section 22.1). Mell et al. [13] and Hall et al. [6] confirm that conventional IDS evaluation metrics express the attack detection accuracy of an IDS only for a specific hardware environment in which the IDS is expected to reside during operation.

Based on the above, we argue that the use of conventional IDS evaluation metrics may lead to inaccurate measurements in cases where the elastic behavior of a given virtualized environment has a significant impact on the attack detection accuracy exhibited by an IDS deployed in the environment by 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 attack detection accuracy exhibited by the IDS. For example, the attack detection accuracy exhibited by a network-based IDS may be correlated to the number of dropped packets by the IDS in the time intervals when attacks have been performed. Large amounts of dropped packets in such intervals due to lack of resources may manifest themselves as low IDS attack detection accuracy. Not quantifying the impact of the former on the latter may lead to incomplete, inaccurate observations about the accuracy of the IDS.

In a scenario 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 introduces the following inter-related issues:

- *challenging 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 and false 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 that purpose, such a correlation would be approximative, which may lead to inaccurate observations;
- *inaccurate comparisons of IDSes*: The approximative nature of the correlation mentioned above rules out accurate comparisons of the attack detection accuracy of multiple IDSes by taking elasticity into account. Note that IDS comparisons are a common goal of IDS evaluation studies [15]. Comparing IDSes requires precise measurement of considered metric values so that the comparisons are accurate and fair.

22.3.1 Transient IDS Behaviors and IDS Attack Detection Accuracy

In this section, we discuss and demonstrate the impact of relevant transient IDS behaviors on IDS attack detection accuracy. We consider the case of a network-based IDS deployed as a VNF (see Figure 22.2). The transient IDS behavior of interest is amount of dropped packets over time (see Section 22.3).

There are many factors influencing the amount of packets dropped by an IDS over the duration of an evaluation experiment. These include characteristics of processed workload (e.g., the speed the network traffic processed by the IDS), and the configuration and design of the IDS (e.g., use of multithreading for processing network packets in an efficient manner). If the IDS under test is deployed in a virtualized environment, an additional important factor is the hypervisor, which may provision on-demand resources to the VM where the IDS operates (see Section 22.3).

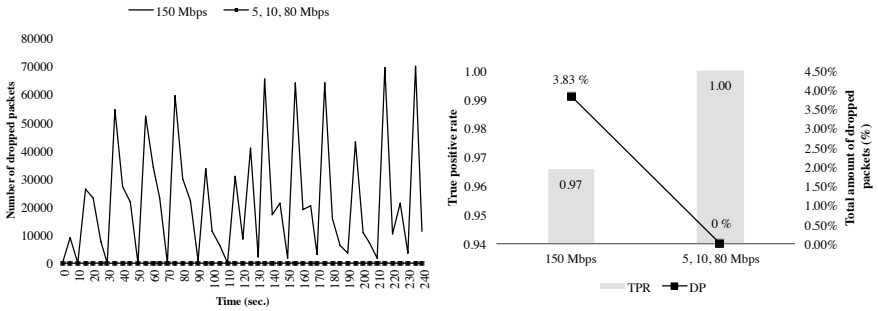
In this work, we focus on the hypervisor as a factor influencing transient IDS behaviors, which we discuss in detail in paragraph 'the hypervisor'. However, we note that having an overview of the other factors influencing a given transient IDS behavior of interest is beneficial when it comes to quantifying IDS attack detection accuracy by taking elasticity into account. This is because it allows for the precise measurement of the impact of the hypervisor on IDS attack detection accuracy by varying relevant configuration points between measurements (e.g., enforcing different resource provisioning policies), not changing characteristics of the other factors (e.g., the speed of the network traffic used as workload). Note that these factors may also have a significant impact on transient IDS behaviors and therefore on IDS attack detection accuracy. Next, we demonstrate how the speed of the network traffic processed by an IDS impacts the amount of packets dropped by the IDS over time, which, in turn, impacts the attack detection accuracy exhibited by the IDS.

We deployed Snort 2.9.7.0 in a paravirtualized VM with an Ubuntu 14.04 operating system running on top of a Xen 4.4.1 hypervisor. We allocated 2 CPUs of 2.6 GHz, 4GB of main memory, and a NIC with a maximal data transfer rate of 1 Gbit/second to the VM where Snort was deployed.[8] We replayed over 240 seconds a trace file from the 1998 DARPA datasets.[7] We configured Snort to use a database of rules dated 10th April 2015. All other configuration options of Snort were set to their default values.

We performed four separate experiments such that we replayed network traffic at the speed of 5, 10, 80, and 150 Mbps. We repeatedly executed each experiment 30 times and we averaged the results. In Figure 22.3a, we depict the number of packets dropped by Snort over 240 seconds for each considered network traffic speed. In Figure 22.3a, one can observe that Snort dropped a certain amount of packets when we replayed network traffic at the speed of 150 Mbps.

In Figure 22.3b, we depict the true positive rate exhibited by Snort in relation to the total amount of dropped packets at the considered network traffic speeds. One can observe a decline of 0.03 of the measured true positive rate when network traffic

[8] We used this testbed environment for all experiments presented in this paper.



(a) (b)

Fig. 22.3: Relevant transient behavior and attack detection accuracy of Snort: a) number of dropped packets over time, and b) measured true positive rate (TPR) in relation to the total amount of dropped packets (DP)

is replayed at the speed of 150 Mbps. This can be attributed to the larger amount of packets dropped by Snort (i.e., 3.83% of all replayed network packets), some of which are malicious.

The hypervisor We now discuss and demonstrate through case studies the impact that the hypervisor may have on relevant transient behaviors (i.e., number of dropped packets over time) of an IDS deployed in a virtualized environment (i.e., a network-based IDS deployed as a NVF). Modern hypervisors feature on-demand provisioning of CPU and memory resources, performed by hotplugging virtual CPU(s) and memory on running VMs (see Section 22.1). We investigate here the impact of two relevant characteristics of CPU and memory hotplugging on the number of packets dropped by an IDS over time and therefore on its attack detection accuracy: hotplugging *intensity* (i.e., amount of hotplugged resources) and hotplugging *speed* (i.e., the hypervisor's speed at provisioning resources with respect to resource demands).

Case Study #1: CPU hotplugging intensity We demonstrate through this case study the impact of CPU hotplugging intensity on IDS attack detection accuracy. We deployed the IDSes Snort 2.9.7.0 and Suricata 2.0.6 in our testbed environment. We allocated one virtual CPU to the VM where Snort and Suricata were 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 (i.e., in scenarios where a VM on which CPU is hotplugged is under CPU pressure). We replayed over 240 seconds, at the speed of 150 Mbps, a trace file from the 1998 DARPA datasets.[7] All configuration options of Snort and Suricata were set to their default values.

By experimenting with both Snort and Suricata, we demonstrate how CPU hotplugging affects the number of packets dropped by IDSes with different designs. Therefore, besides the hypervisor, we also demonstrate the impact of IDS design as a factor influencing transient IDS behaviors (see Section 22.3). Note that Suricata

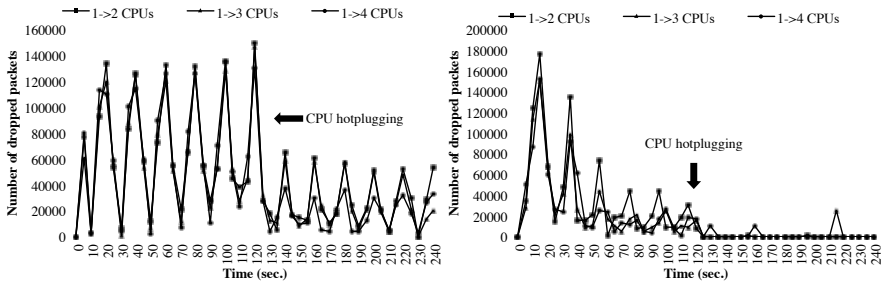


Fig. 22.4: Number of packets dropped over time: a) by Snort, and b) by Suricata

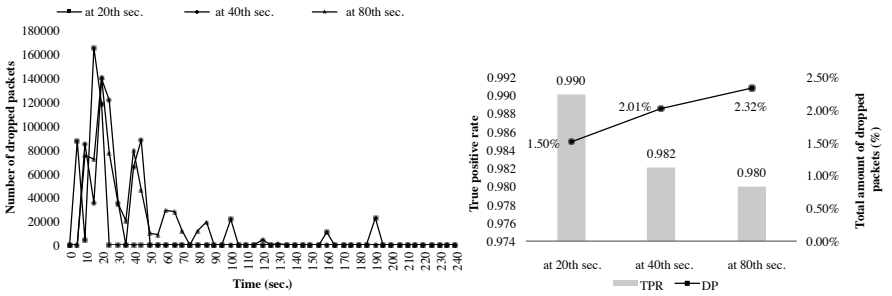
features multi-threading and is therefore utilising multiple CPUs more effectively than Snort.

We performed six separate experiments; that is, three experiments for each considered IDS such that we hotplugged one, two, and three additional virtual CPUs on the VM where Snort/Suricata was deployed, at the 120th second of each experiment. We repeated each experiment 30 times and we averaged the results. In Figure 22.4a and Figure 22.4b, we depict the number of packets dropped by Snort and Suricata over 240 seconds for each considered CPU hotplugging scenario (1→2 CPUs, 1→3 CPUs, and 1→4 CPUs in Figure 22.4). In Table 22.3, we present the attack detection accuracy of Snort and Suricata we measured for each hotplugging scenario, that is, exhibited true and false positive rate, in relation to the total amount of dropped packets (expressed in percentage in Table 22.3).

Table 22.3: Attack detection accuracy of Snort and Suricata [TPR — true positive rate; FPR — false positive rate; DP – dropped packets]

CPU hotplugging scenario	Snort			Suricata		
	TPR	FPR	DP (%)	TPR	FPR	DP (%)
1 → 2 CPUs	0.924	0.0000034	8.26	0.967	0.0000098	3.37
1 → 3 CPUs	0.929	0.0000031	7.71	0.972	0.0000109	2.79
1 → 4 CPUs	0.932	0.0000032	7.48	0.975	0.0000108	2.73

As expected, the true positive rates exhibited by Snort and Suricata increase as more CPUs are hotplugged on the VM where the IDSes are deployed. This is due to the decrease of the number of packets dropped by the IDSes after CPUs have been hotplugged (see Figure 22.4a and Figure 22.4b). The false positive rates exhibited by Snort and Suricata vary with respect to the accuracies of the IDSes as well as the particular packets dropped by the them (i.e., some of the dropped packets would



(a) (b)

Fig. 22.5: Relevant transient behavior and attack detection accuracy of Suricata: a) number of dropped packets over time, and b) measured true positive rate (TPR) in relation to the total amount of dropped packets (DP)

have been falsely considered malicious if they had been processed by the IDSes). We do not elaborate further on the false positive rate.

In Figure 22.4a and Figure 22.4b, one can observe the impact of CPU hotplugging intensity on the number of packets dropped over time by IDSes with different designs; that is, in contrast to Snort, Suricata dropped significantly less packets (i.e., 3.37%, 2.79%, and 2.73%, see Table 22.3) due to its ability to effectively utilise multiple CPUs. This results in Suricata exhibiting higher true positive rates than Suricata.

Case Study #2: CPU hotplugging speed We demonstrate through this case study the impact of CPU hotplugging speed on IDS attack detection accuracy. We deployed Suricata 2.0.6 in our testbed environment and we allocated one virtual CPU to the VM where Suricata was deployed so that the VM is under CPU pressure when workloads are run. We replayed over 240 seconds, at the speed of 150 Mbps, a trace file from the 1998 DARPA datasets.[7] All configuration options of Suricata were set to their default values. We performed four experiments such that we configured the hypervisor to hotplug three CPUs at the 20th, 30th, 40th and the 80th second of the experiment. We repeated each experiment 30 times and we averaged the results.

In Figure 22.5a, we depict the number of packets dropped by Suricata over 240 seconds for each considered hotplugging scenario (at 20th sec., at 40th sec., and at 80th sec. in Figure 22.5a). In Figure 22.5b, we depict the true positive rate exhibited by Suricata in relation to the total amount of dropped packets for each considered hotplugging scenario. As expected, one can observe in Figure 22.5b that CPU hotplugging speed has significant impact on the attack detection accuracy exhibited by Suricata. For instance, there is a decrease of 0.01 of the true positive rate exhibited by Suricata when the hypervisor provisioned CPUs at the 80th instead of the 20th second of the experiment. This is due to the loss of additional 0.82% of all replayed packets because of a delay in CPU provisioning of 60 seconds.

Case Study #3: Memory hotplugging intensity We demonstrate through this case study the impact of the memory hotplugging intensity on IDS attack detection accu-

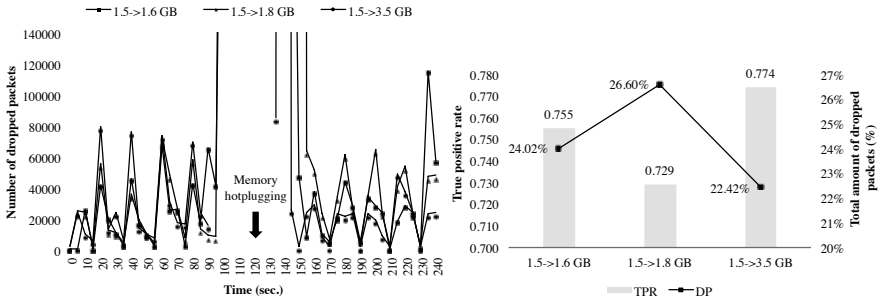
racy. We deployed Snort 2.9.7.0 in our testbed environment and we allocated 1.5 GB of memory to the VM where Snort was deployed so that the VM is under memory pressure when workloads are run. This enabled us to observe the impact of memory hotplugging on the attack detection accuracy exhibited by Snort in scenarios where such a hotplugging is normally performed (i.e., in scenarios where a VM on which CPU is hotplugged is under memory pressure). We replayed over 240 seconds, at the speed of 150 Mbps, a trace file from the 1998 DARPA datasets.[7] We set all configuration options of Snort to their default values. We performed three separate experiments such that we allocated additional 0.1 GB, 0.3 GB, and 2 GB of memory to the VM where Snort was deployed at the 120th second of the experiment. We repeated each experiment 30 times and we averaged the results.

In Figure 22.6a, we depict the number of packets dropped by Snort over 240 seconds for each considered hotplugging scenario (1.5→1.6 GB, 1.5→1.8 GB, and 1.5→3.5 GB in Figure 22.6a). In Figure 22.6b, we depict the true positive rate exhibited by Snort in relation to the total amount of dropped packets for each considered hotplugging scenario.

The results from this study show that on-demand resource provisioning may have diverse impacts on transient IDS behaviors and therefore on IDS attack detection accuracy. This further emphasizes the need of novel metrics and measurement methods for quantifying these impacts (see Section 22.3). For instance, in Figure 22.6a, one can observe a significant increase of packets dropped by Snort when memory is hotplugged. This is followed by a stable transient behavior of the IDS, which, as expected, is dropping less packets than before the memory hotplugging action. Depending on the amount of hotplugged memory, this normally leads to an improvement of the exhibited true positive rate to a certain extent (see in Figure 22.6b the true positive rate exhibited by Snort when 0.1 GB and 2 GB are hotplugged). However, when 0.3 GB of memory is hotplugged, a significant amount of packets is dropped, which leads to the lowest true positive rate we measured (i.e., 0.729, see Figure 22.6b). This is because of many factors involved, for example, the way in which the IDS under test and/or the operating system where the IDS is deployed have been designed to handle various amounts of newly allocated memory.

22.4 Metric and Measurement Methodology

In this section, we present our preliminary work on a novel metric and measurement methodology that take elasticity of virtualized environments into account (see Section 22.1). The metric and methodology we propose enable the measurement of the attack detection accuracy of an IDS deployed in a virtualized environment featuring on-demand resource provisioning; that is, they enable the evaluation of the attack detection accuracy of such an IDS with respect to the impact that on-demand resource provisioning performed by the underlying hypervisor has on the attack detection accuracy exhibited by the IDS (see for example Section 22.3.1, paragraph ‘the hypervisor’). The metric and measurement methodology we propose aim to



(a) (b)

Fig. 22.6: Relevant transient behavior and attack detection accuracy of Snort: a) number of dropped packets over time, and b) measured true positive rate (TPR) in relation to the total amount of dropped packets (DP)

address the issues related to the use of conventional IDS evaluation metrics — possibility of inaccurate observations about the accuracy of an IDS, challenging metric value correlation, and inaccurate comparisons of IDSes (see Section 22.3). We stress that the metric and measurement methodology we propose are meant to complement the conventional ones and are to be used only when it comes to evaluating IDSes deployed in virtualized environments featuring on-demand resource provisioning. 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 (see Section 22.3.1).

Quantifying the impact of the hypervisor on IDS attack detection accuracy calls for a novel definition of the boundaries of a system-under-test (SUT) in the area of IDS evaluation. In the area of system 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, since it is an important factor impacting transient IDS behaviors, which, in turn, impact IDS attack detection accuracy. In Figure 22.7, 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 (see Figure 22.2).

22.4.1 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 re-

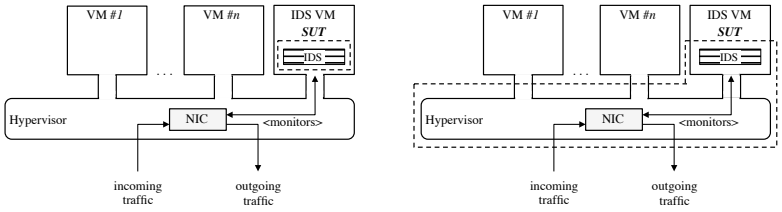


Fig. 22.7: Boundaries of: a) the conventional SUT, and b) novel SUT in the area of IDS evaluation

sources 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 22.3). 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 22.3). 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 22.2). 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 22.2). 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 22.7b).

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 22.4.3, 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 22.4.3.

22.4.2 Metric construction

We present here the main principles of construction for the HF metric. Similar to Gaffney et al. [3], we construct the HF metric using a construct from decision theory — a decision tree — as a basis. In Figure 22.8, we depict the decision tree that we use for constructing the HF metric. The tree shows the sequence of uncertain events (circles) that describe:

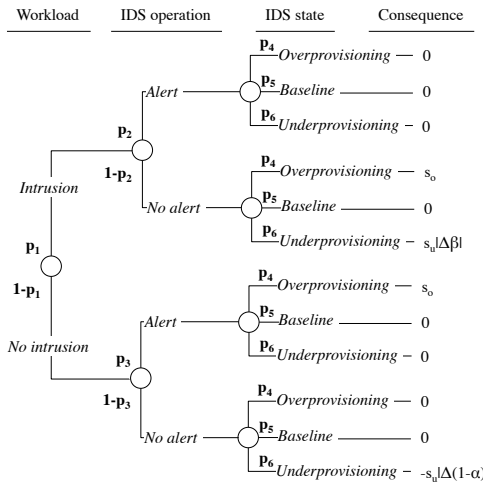


Fig. 22.8: The decision tree used for constructing the HF metric

• 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 22.2);

• 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 22.2);

• the *state* of the IDS (i.e., baseline, overprovisioned, or underprovisioned IDS state, see Section 22.4.1) 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 22.8 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 22.4.1), 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. β_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 (α) 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 22.8; 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:

$$\begin{aligned} HF &= B[\beta(\frac{T_o}{T_{max}}s_o + \frac{T_u}{T_{max}}s_u|\Delta\beta|)] + (1-B)[\alpha\frac{T_o}{T_{max}}s_o - (1-\alpha)\frac{T_u}{T_{max}}s_u|\Delta(1-\alpha)|)] \\ &= \frac{T_o}{T_{max}}s_o[B + (1-B)\alpha]\beta + \frac{T_u}{T_{max}}s_u[B\beta|\Delta\beta| - (1-B)(1-\alpha)|\Delta(1-\alpha)|] \end{aligned} \tag{22.1}$$

If the values of s_o and s_u are set to 1, Equation 22.1 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.

22.4.2.1 On baseline IDS state

Calculating values of the HF metric requires calculating T_o, T_u, and T_b, and, in addition, β_b and $(1-\alpha)_b$ (see Equation 22.1). 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 22.4.1); 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.

22.4.3 Properties of the HF metric

In this section, we show how the HF metric satisfies each of the design criteria we presented in Section 22.4.1:

Criterion C_1: For a given T_{max}, the value of the HF_o component of the HF metric (see Section 22.4.2) 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 (α), and the false negative rate (β);

Criterion C_2 ***and*** C_3: For a given T_{max}, the value of the HF_u component of the HF metric (see Section 22.4.2):

- 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 22.1);

Criterion C_5: In Definition 22.1, we define an operating point of an SUT (see Figure 22.7b).

Definition 22.1. 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 α_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 α (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 22.1 and Definition 22.1). 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.

22.5 Conclusion

In this chapter, we elaborated on evaluating in an accurate manner attack detection accuracy of IDSes deployed in virtualized environments featuring on-demand resource provisioning. We demonstrated through case studies the impact of such a provisioning on IDS attack detection accuracy. We surveyed conventional metrics for quantifying IDS attack detection accuracy observing that they do not express this impact, which may lead to inaccurate assessments when using these metrics to evaluate an IDS deployed in a virtualized environment. We presented a novel metric — the *HF metric* — and a measurement methodology, which capture the impact of on-demand resource provisioning on IDS attack detection accuracy and therefore contribute towards addressing the previously mentioned issue.

We designed the HF metric with respect to several criteri, such as SUT penalizing and rewarding criteria, and expression of the base rate. We consider these criteria crucial for the credible assessment of the attack detection accuracy of an IDS deployed in a virtualized environment.

The metric and measurement methodology we presented in this chapter are in their preliminary forms. This work can be continued in several directions. For instance, in-depth analysis of various properties of the HF metric is needed (e.g., analysis on how values of the HF metric relate to base rate measures). In addition, different SUT penalizing and rewarding criteria may be considered. Further, we plan to conduct realistic case studies involving evaluation of single or multiple SUTs in order to demonstrate the practical usefulness of the HF metric.

We stress that rigorous metrics are essential not only for the accurate evaluation of IDSes, but also as a driver of innovation by enabling the identification of issues and the improvement of existing IDSes.

Acknowledgements This research has been supported by the Research Group of the Standard Performance Evaluation Corporation (SPEC, http://www.spec.org, http://research.spec.org). The authors would like to thank Alexander Leonhardt for providing experimental data.

References

1. Tcpreplay.
2. Wesam Dawoud, Ibrahim Takouna, and Christoph Meinel. Elastic Virtual Machine for Fine-Grained Cloud Resource Provisioning. In P.Venkata Krishna, M.Rajasekhara Babu, and Ezendu Ariwa, editors, *Global Trends in Computing and Communication Systems*, volume 269 of *Communications in Computer and Information Science*, pages 11–25. Springer, 2012.
3. Jr. Gaffney, J.E. and J.W. Ulvila. Evaluation of intrusion detectors: a decision theory approach. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, pages 50–61, 2001.
4. Frank Gens, Robert Mahowald, Richard L. Willards, David Bradshaw, and Chris Morris. Cloud computing 2010: An idc update, 2010.
5. Guofei Gu, Prahlad Fogla, David Dagon, Wenke Lee, and Boris Skorić. Measuring intrusion detection capability: an information-theoretic approach. In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security (ASIACCS)*, pages 90–101, New York, NY, USA, 2006. ACM.
6. Mike Hall and Kevin Wiley. Capacity verification for high speed network intrusion detection systems. In *Proceedings of the 5th International Conference on Recent Advances in Intrusion Detection (RAID)*, pages 239–251, Berlin, Heidelberg, 2002. Springer-Verlag.
7. J. Hancock and P. Wintz. *Signal Detection Theory*. McGraw–Hill, New York, 1966.
8. Evangelos Kotsovinos. Virtualization: Blessing or curse? *Queue*, 8(11):40:40–40:46, November 2010.
9. Sajib Kundu, Raju Rangaswami, Ajay Gulati, Ming Zhao, and Kaushik Dutta. Modeling virtualized applications using machine learning techniques. In *Proceedings of the 8th ACM SIGPLAN/SIGOPS conference on Virtual Execution Environments*, VEE '12, pages 3–14, New York, NY, USA, 2012. ACM.
10. Flavio Lombardi and Roberto Di Pietro. Secure virtualization for cloud computing. *Journal of Network and Computer Applications*, 34(4):1113–1122, July 2011.
11. Neil MacDonald. Yes, Hypervisors are vulnerable. http://blogs.gartner.com/neil_macdonald/2011/01/26/yes-hypervisors-are-vulnerable/, 2011.
12. R. A. Maxion and R. R. Roberts. Proper Use of ROC Curves in Intrusion/Anomaly detection. Technical Report CS-TR-871, School of Computing Science, University of Newcastle upon Tyne, November 2004.

13. Peter Mell, Vincent Hu, Richard Lippmann, Josh Haines, and Marc Zissman. An Overview of Issues in Testing Intrusion Detection Systems, 2003.
14. Yuxin Meng and Wenjuan Li. Adaptive Character Frequency-Based Exclusive Signature Matching Scheme in Distributed Intrusion Detection Environment. In *IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 223–230, June 2012.
15. Aleksandar Milenkoski, Marco Vieira, Samuel Kounev, Alberto Avrtizer, and Bryan D. Payne. Evaluating Computer Intrusion Detection Systems: A Survey of Common Practices. *ACM Computing Surveys*, 2015. To appear.
16. N. Mohammed, H. Otrok, Lingyu Wang, M. Debbabi, and P. Bhattacharya. Mechanism Design-Based Secure Leader Election Model for Intrusion Detection in MANET. *IEEE Transactions on Dependable and Secure Computing*, 8(1):89–103, January-February 2011.
17. Diego Perez-Botero, Jakub Szefer, and Ruby B. Lee. Characterizing hypervisor vulnerabilities in cloud computing servers. In *Proceedings of the 2013 International Workshop on Security in Cloud Computing*, Cloud Computing '13, pages 3–10. ACM, 2013.
18. Martin Roesch. Snort - Lightweight Intrusion Detection for Networks. In *Proceedings of the 13th USENIX conference on System Administration (LISA)*, pages 229–238. USENIX Association, 1999.
19. Karen Scarfone and Peter Mell. Guide to Intrusion Detection and Prevention Systems (IDPS), 2007. NIST Special Publication 900-94.
20. Sushant Sinha, Farnam Jahanian, and Jignesh M. Patel. WIND: Workload-aware INtrusion Detection. In *Proceedings of the 9th International Conference on Recent Advances in Intrusion Detection (RAID)*, pages 290–310, Berlin, Heidelberg, 2006. Springer Verlag.
21. S. Spinner, S. Kounev, Xiaoyun Zhu, Lei Lu, M. Uysal, A. Holler, and R. Griffith. Runtime Vertical Scaling of Virtualized Applications via Online Model Estimation. In *IEEE Eighth International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, pages 157–166, 2014.
22. Jing Xu, Ming Zhao, José Fortes, Robert Carpenter, and Mazin Yousif. Autonomic Resource Management in Virtualized Data Centers Using Fuzzy Logic-based Approaches. *Cluster Computing*, 11(3):213–227, 2008.