SLA Definition for Network Intrusion Detection Systems in IaaS Clouds

Amir Teshome Wonjiga, Louis Rilling, Christine Morin

SLA Definition for Network Intrusion Detection Systems in IaaS Clouds

Amir Teshome Wonjiga
Univ Rennes, Inria, CNRS, IRISA
Rennes, France
amirteshome.wonjiga@gmail.com

Louis Rilling
DGA
Bruz, France
louis.rilling@irisa.fr

Christine Morin
Inria
Rennes, France
christine.morin@inria.fr

ABSTRACT

Migrating to the cloud results in losing full control of the physical infrastructure as the cloud service provider (CSP) is responsible for managing the infrastructure including its security. To solve the trust issue that this raises, CSPs provide tenants with guarantees through Service Level Agreements (SLA). However no such SLA addresses the security monitoring aspect of tenants' information systems. Moreover, security monitoring services should be configured according to the tenant's specific requirements. In this paper, we propose a method allowing CSPs to define SLAs providing each tenant with guarantees about the performance of a security monitoring probe, specifically a Network Intrusion Detection System (NIDS), configured according to the tenant's requirements. This method is based on an enhanced cloud SLA language and an efficient SLA template preparation method allowing a CSP to estimate the performance of an NIDS for any possible set of tenant's requirements at reasonable costs. Experimental evaluations show the feasibility of our approach.

CCS CONCEPTS

• **Security and privacy** → **Intrusion detection systems**; **Intrusion detection systems**; **Vulnerability management**; **Security services**; • **Networks** → **Cloud computing**; **Network monitoring**; • **General and reference** → *Metrics*;

KEYWORDS

SLA, SLA language, NIDS performance, rule interference

ACM Reference Format:
Amir Teshome Wonjiga, Louis Rilling, and Christine Morin. 2021. SLA Definition for Network Intrusion Detection Systems in IaaS Clouds. In *The 36th ACM/SIGAPP Symposium on Applied Computing (SAC '21), March 22–26, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, Article 4, 10 pages. https://doi.org/10.1145/3412841.3441885

1 INTRODUCTION

Compared to on-premises infrastructures the cloud introduces a new type of system ownership. In the cloud, depending on the service model, the tenant and the cloud service provider (CSP) own different parts of the system. For instance, in an IaaS cloud the provider owns the physical infrastructure and the virtualization layer while the tenant owns its information system deployed on top of the virtualization layer. When migrating to the cloud the tenant loses full control of the physical infrastructure. The CSP is responsible for managing the infrastructure and thus monitors every aspect from resource allocation, performance to security. This creates a trust issue between CSPs and tenants.

CSPs acknowledge the trust issue and endeavor to provide assurance through an agreement called *Service Level Agreement (SLA)*. SLAs help in building a trustworthy relationship between CSPs and tenants. An SLA provides a tenant with a guarantee, as it includes a reward (a penalty for the CSP) whenever the service is not rendered correctly. SLAs are negotiated, hence tenants can specify their requirements to get a service well tailored to their needs. The SLA document describes the provided service, the rights and obligations of all participants and a quantitative description of the targeted quality of service, called Service Level Objective (SLO), using Key Performance Indicators (KPI). However, to the best of our knowledge existing cloud SLAs do not provide a guarantee regarding the security monitoring aspect of the hosted information system.

Security monitoring is the collection, analysis, and escalation of indications and warnings to detect and respond to intrusions [3]. The goal of collecting and analyzing events and generating indicators is to *detect* and prevent intrusions. In addition, when prevention eventually fails, the goal is to *respond* to incidents as quickly as possible and understand how the intruder achieved its attack and what damage it made. Different types of security monitoring devices and techniques are used for various components. Notable examples of such devices are Intrusion Detection Systems (IDSs) [21].

The SLA considered in our study guarantees the performance of signature-based Network Intrusion Detection Systems (NIDSs) managed by the CSP. While tenants can have a self-configured NIDS in their own environment, usually cloud tenants are not experts in system security. Moreover, an NIDS managed by the CSP may have a broader visibility of the traffic that is potentially malicious for the tenant. Thus, the SLA can achieve mutual responsibility with a trade-off between tenant's private information disclosed and the monitoring service offered. We also focus on signature-based NIDS as they are more efficient in detecting exploits of already-known vulnerabilities than anomaly-based NIDSs [12].

Guaranteeing the performance of a monitoring probe is a complementary approach to the traditional security SLA (SecSLA) [10, 17] which aims to guarantee traditional security properties (e.g. confidentiality, integrity, privacy…). Finding an applicable KPI for such properties is still an open research problem. Our approach improves the state of the art of security SLAs by providing a guarantee for security monitoring probes. To this end, in this paper we introduce a practical method to define model-based KPIs for NIDSs. Moreover,

SAC'21, March 22-March 26, 2021, Gwangju, South Korea

ACM ISBN 978-1-4503-8104-8/21/03…$15.00
https://doi.org/10.1145/3412841.3441885

since such devices oversee the security status of a given system and can detect violations of the security policies, guaranteeing their performance helps to guarantee the traditional security properties.

The life-cycle of an SLA can be separated in three phases called *SLA definition and negotiation*, *SLA enforcement* and *SLA verification* [20]. Methods were proposed for the *SLA verification* phase in the case of an NIDS [2, 18, 22]. In this paper, we show how we can achieve the *SLA definition and negotiation* phase, thanks to new constructs in a cloud SLA language and an efficient knowledge-base building method for NIDS performance.

The overall contribution presented in this paper is a method allowing CSPs to define SLAs providing each tenant with guarantees about the performance of an NIDS configured according to the tenant's requirements. This contribution is composed of the following sub-contributions:

- We propose ECSLA, an SLA language suitable for security monitoring and based on an SLA language for clouds.
- We propose a model-based dynamic performance metrics computation method allowing to use a KPI based on all the necessary parameters, including the impossible-to-know-online rate of occurrence of attacks (base rate). To our knowledge, our work is the first showing how to use such security monitoring metrics in practice.
- We propose an efficient knowledge-base building method for NIDS performance on large numbers of vulnerabilities. This method makes it possible to prepare SLO templates despite the combinatorial explosion resulting from tenants having highly different needs and being concerned by only a few vulnerabilities among a large number of possible ones.
- We conducted an experimental evaluation validating our theoretical results.

The rest of the paper is organized as follows. Section 2 details the problem addressed. Related works are presented in Section 3. The components of security monitoring SLAs are described in Section 4. We present a method to include the base rate in the SLA definition in Section 5. A clustering method to reduce the required number of evaluation tests in SLO templates preparation is introduced in Section 6. In Section 7 we present an experimental evaluation. Finally, we conclude and present future works in Section 8.

2 PROBLEM DESCRIPTION

Our work aims to include security monitoring terms into IaaS cloud SLAs. To achieve this tenants are provided with guarantees on the performance of security monitoring tools and devices. In this paper we focus on NIDSs and on the first step in the SLA life-cycle, which is to define the agreement.

The SLA definition incorporates components such as *Service Description*, *SLO*, *Parties* and *Penalties*. For security monitoring terms, tenants are expected to describe their needs in terms of vulnerabilities. Thus, the service description should be able to express *vulnerabilities*. Vulnerabilities represent the finest granular concept to express tenant's specific needs. In addition, a vulnerability should be related to other services like products (software) where the vulnerability exists and infrastructure where the product is running. Vulnerabilities are also associated with previously known attacks. i.e., monitoring a vulnerability implies detecting and responding to the occurrence of attacks that are associated with the vulnerability.

A metric is required in order to express the performance of NIDSs. For a metric to be used in an SLO, it should satisfy two characteristics. (i) The metric should take the rate of occurrence of attacks (i.e. *base rate*) into consideration. Axelsson [1] showed the base rate fallacy problem for NIDSs and indicated the importance of the base rate parameter. The non-deterministic nature of the base rate parameter makes it challenging to define security monitoring SLAs. An SLA should not be defined for specific values of base rate because, most of the time, every occurrence of an attack has a different rate, i.e., a different base rate. (ii) It should be a single valued metric i.e., not a combination of two or more metrics. Indeed, comparing coupled metric values from different providers requires deep analysis and finding trade-offs. Hence, from the tenants perspective, it is easier to have a single unified metric.

From the providers' perspective, two facts make the SLA definition process challenging. First, there are thousands of public vulnerabilities [9] and a tenant is usually interested in only a few of them, typically the vulnerabilities related to the application(s) she deploys. Second, the number of rules in an NIDS affects its performance [6], which incites providers to minimize the set of rules configured. This is all the more critical because many such sets of rules are required to address simultaneously the needs of many tenants.

To solve this issue CSPs should know the performance of their monitoring devices on every possible combination of vulnerabilities. However the combinatorial explosion makes it not practical to test an NIDS for every possible configuration. Hence, CSPs need an efficient performance estimation method to prepare SLA templates.

3 RELATED WORKS

The challenge of including security services in SLAs has been a subject of study before the advent of cloud. With the development of clouds there has been different attempts to include security terms in cloud SLAs [4, 5, 13, 17, 20].

Bernsmed et.al [5] discussed the need to include security in SLAs and described security in SLAs as *Quality of Protection (QoP)*, which comprises the ability of a CSP to deliver a service according to a set of specific security requirements. QoP is described with one or more security mechanisms. In another paper [4] a similar method is presented for composing security SLAs for federated cloud services. These works are very first steps and lack quantitative descriptions of security services, enforcement and verification mechanisms.

To address the issue of verifiability, Da Silva et.al [10] proposed an approach to build a Security-SLA. They used two classes of metrics to define security SLAs (i) infrastructure metrics and (ii) services metrics. Monitoring agents run both inside and outside the tenant environment to collect security metrics. Collected metrics are mapped to scale values from [0-4] representing the range of security values as [Critical, High, Medium, Normal, Zero]. However, the proposal did not consider all critical security aspects and services in the cloud. It is not clear which metrics are being collected and how the critical level is assigned. Moreover, other than running monitoring agents in different locations of the cloud, it is not clear

how the actual measurement is performed and responsibilities of tenants, CSPs or third parties are not specified.

D. Petcu et.al [17] studied conceptual and technical barriers to implement a Sec-SLA monitoring service. Some barriers include the lack of acceptance and maturity of SLA management systems, the difficulty of mapping high level security properties to low level monitoring parameters, the lack of deployment-layer agnosticism, and extra complexity introduced by virtualization. This study was used as an input for a European project called SPECS [20].

The SPECS project focused on securing cloud resource provisioning using SLA management [8] and an SLA-based approach is used to have a common language between providers and tenants. It extends a language called WS-Agreement to describe security SLAs [7]. The core of the SPECS framework consists of three components: *negotiation*, *enforcement* and *monitoring*. Users specify security requirements using provided templates. A feasibility plan is generated by formulating an optimization problem aiming to find the minimum number of resources required to implement the SLA. The SLA is implemented if there is a solution satisfying the objectives of the problem.

Kaaniche *et al* [13] presented security SLA monitoring mechanisms. They extend a language called rSLA and its runtime service, named Sec-rSLA, to support the description of security requirements. However, both SPECS and Sec-rSLA miss quantifiable metrics to describe the security of tenants' environments. Use cases presented in [7, 8] use security policy constraints as an input for building SLAs (e.g. the load balancer should not be collocated with a server or the IDS needs exclusive use of machines…). Similarly, Sec-rSLA uses policies like the number of replicas for a given data as SLO in security SLAs. This mostly results from difficulties to quantitatively measure security.

SLAs are described using standard languages. The advantage of formalizing the SLA definition is manyfold including to automate the life-cycle of SLA, to pass SLA as an input for an autonomic system and comparability i.e., comparing offers from different providers. There are a number of standard languages proposed to describe cloud SLAs. CSLA [14] is one of them designed specifically for cloud SLAs. CSLA introduces novel properties for SLA metrics called *fuzziness* and *confidence ratios*. The former defines the acceptable margin around the threshold value of an SLO parameter, whereas the latter defines the percentage of compliance of SLO clauses.

The fuzziness and confidence ratios properties of CSLA are interesting for cloud security SLAs. Indeed, as the environment is very dynamic respecting a fixed SLO for every fraction of seconds may not be feasible. Especially, assuming complex metrics measuring different properties of the cloud (e.g., performance of a security monitoring device), achieving a fixed target SLO is challenging.

However CSLA misses features to describe vulnerabilities to be monitored. Indeed CSLA was designed to describe functional properties (e.g. response time) of cloud services, whereas defining a security monitoring service requires to link vulnerabilities to be monitored with software products and their location in the virtual infrastructure. Thus, we propose to extend the CSLA language with constructs allowing to describe these links (see Section 4.1).

4 COMPONENTS OF SECURITY MONITORING SLAS

Besides the anatomy similarity with other cloud SLAs the content of security monitoring SLAs differs from others in a few ways. In this section we focus on two components of security monitoring SLAs: the KPI used to describe the performance of NIDSs in SLOs and classes of SLO that we consider. Next, we present an SLA language called *Extended CSLA (ECSLA)*, which is an extension to the CSLA language. In addition, we present an example of security monitoring SLA described using ECSLA.

SLAs include:

- *Service Description:* The security monitoring service is used to monitor existing vulnerabilities in a product. Hence, the service definition correlates three components namely *product, vulnerabilities* and *infrastructure*.
- *Service Level Objectives (SLO):* SLOs describe the guaranteed level of performance. For an NIDS, it shows how much a given NIDS is effective in detecting the attacks (correlated with a given vulnerability) listed in the service description.

4.1 ECSLA

We propose *Extended CSLA (ECSLA)*, an extension to the CSLA language, to formalize our security monitoring SLA definition.

ECSLA inherits from CSLA the fuzziness and confidence ratio properties of an SLO. As a result of the dynamic nature of the cloud, such additional margins are useful while dealing with security. ECSLA extends the original CSLA language with new features, including a new generic service, a structure to define security vulnerabilities, and a definition of security monitoring service.

4.2 KPI for Security Monitoring SLO

The KPI we use to describe the performance of an NIDS is the *Intrusion Detection Capability (C_{ID})* [11]. For a metric to be used in a security monitoring SLO two features are required, namely being a single unifying metric and taking the base rate into account. C_{ID} satisfies both of these features.

Gu *et al* introduced the C_{ID} metric in [11]. Let X be the random variable representing the IDS input as either "attack" or "legitimate" packets (thus the base rate $B = P(X = \text{"attack"})$), Y the random variable representing the IDS output where a packet can be detected as intrusive or non-intrusive, $H(X)$ be the entropy of X as defined in information theory, and $I(X;Y)$ the mutual information which measures the amount of information shared between the two random variables. The C_{ID} is defined as: $C_{ID} = \frac{I(X;Y)}{H(X)}$. The value of C_{ID} ranges in $[0, 1]$ and increases with the IDS ability to accurately classify the input packets. This unified metric can be expressed in terms of *True Positive Rate* (TPR), *False Positive Rate* (FPR) and *base rate* (B). TPR and FPR, as well as their complement FNR (False Negative Rate) and TNR (True Negative Rate) are derived from the basic metrics *True Positive (TP), False Positive (FP), True Negative (TN)* and *False Negative (FN)*: for instance $TPR = \frac{TP}{TP+FN}$.

The SLA describes how to compute the expected C_{ID} value from TPR, FPR and B. Since it is not possible to know the values of B in advance, the SLA definition does not contain the exact B values.

Application	Version	Attacks
Apache	Apache/2.4.7 (Ubuntu)	DoS and port scan
Mysql	14.14 Distrib 5.6.31	Brute force access
WordPress(WP)	V. 4.4.5	None
Instalinker (WP plugin)	V. 1.1.1	Cross site scripting (XSS)
Custom Contact Forms	V. 5.1.0.2	SQL injection

Table 1: Example of considered services and attack types

Instead, providers use a model-based approach to offer an SLO (see Section 5).

4.3 Class of SLOs Considered

In the SLA definition phase providers prepare SLO templates. These templates describe the expected effectiveness of NIDSs using the C_{ID} metric while they are configured to monitor vulnerabilities. Providers prepare templates taking only *known* vulnerabilities into account. Tenants may require to be monitored against new or emerging vulnerabilities, but from the providers perspective, it is risky to promise the performance of monitoring devices for unknown vulnerabilities. Since signature-based NIDSs are more efficient in monitoring known vulnerabilities than anomaly-based NIDSs (e.g. they produce less false positives), they should suit better CSPs to offer security monitoring SLAs.

Tenants describe their needs by listing the vulnerabilities of their interest from shared databases like the Common Vulnerabilities and Exposures (CVE) database [9] and CSPs state the set of matching attacks to be monitored using NIDS rules. We believe that mutual responsibility can be achieved through an SLA, which makes a trade-off between tenant's private information disclosed and the monitoring service offered. In our work each tenant provides the set of services to be monitored, allowing the provider to offer security monitoring SLOs with clear KPIs.

CSPs associate NIDS rules with the corresponding vulnerabilities in advance. The final SLA contains the list of vulnerabilities that are at the intersection between tenants requirements and security monitoring services proposed by the provider. The agreement describes the list of services to be monitored, the list of their known vulnerabilities, and expected performance. Once the agreement is signed, the NIDS is configured with the rules associated with those vulnerabilities and the tenant information system is thus monitored against attacks which can exploit the listed vulnerabilities.

Examples of services considered to be included in a security monitoring SLA are described in Table 1. The table shows a list of vulnerable applications and attacks to be monitored. Section 7 presents a KPI example using these services.

4.4 Security Monitoring SLA Example

We present an example of SLA based on ECSLA and considering the services described in Table 1. The SLA sections shown are *service description*, *parameters* and *guarantee*.

4.4.1 Service Description. This section contains three sub sections describing (i) vulnerabilities to be monitored (ii) software where the vulnerability exists and (iii) the location in the infrastructure where the software is running.

Listing 1 shows an example of service description section. Four security monitoring services are defined: one for each application with known attacks listed in Table 1. The attacks in the table exploit the vulnerabilities listed in the service description. Each service contains one vulnerability except Apache, which is monitored for two vulnerabilities. The services are running in three different servers. The infrastructure subsections detail which servers host each service (Lines 9, 20, 30 and 41).

```
1  <cloudServices>
2  <cloudService>
3  <macro>
4  <securityMonitoring id="Mysql-SM-ID" description="...">
5    <software id="Mysql-ID" name="Mysql" version="14.14"
         distribution="5.6.31" license="GPL" mode="mode" />
6    <vulnerabilities>
7      <vulnerability id="Mysql-V-1" name="Brute force access" cve_Id
           ="" description="..." />
8    </vulnerabilities>
9    <infrastructure id="DBserver-ID" description="Database server ">
10     <compute id="DB-VM-1" name="DBserver" architecture="" hostname
           ="" cores="" speed="" memory="" />
11   </infrastructure>
12 </securityMonitoring>
13
14 <securityMonitoring id="Apache-SM-ID" description="">
15   <software id="Apache-ID" name="Apache" version="4.4.11"
         distribution="" license=" Apache License" mode="mode" />
16   <vulnerabilities>
17     <vulnerability id="Apache-V-1" name="Port scanning" cve_Id=""
           description="..." />
18     <vulnerability id="Apache-V-2" name="Denial of service" cve_Id
           ="" description="..." />
19   </vulnerabilities>
20   <infrastructure id="Websever-ID" description="Web server ">
21     <compute id="WEB-VM-1" name="Webserver" architecture=""
           hostname="" cores="" speed="" memory="" />
22   </infrastructure>
23 </securityMonitoring>
24
25 <securityMonitoring id="IL-SM-ID" description="">
26   <software id="IL-ID" name="InstaLinker" version=" &lt;= 1.1.1"
         distribution="" license="GPLv2" mode="mode" />
27   <vulnerabilities>
28     <vulnerability id="IL-V-1" name="Cross-Site Scripting (XSS)"
           cve_Id="8382 in WPVDB_ID" description="..." />
29   </vulnerabilities>
30   <infrastructure id="CM-ID" description="Content management
         server ">
31     <compute id="CM-VM-1" name="Cmserver" architecture="" hostname
           ="" cores="" speed="" memory="" />
32   </infrastructure>
33 </securityMonitoring>
34
35 <securityMonitoring id="CCF-SM-ID" description="">
36   <software id="CCF-ID" name="Custom Contact Forms" version=" &lt
         ;= 5.0.0.1"
         distribution="" license="GPLv2" mode="mode" />
37   <vulnerabilities>
38     <vulnerability id="IL-V-1" name="SQL injection" cve_Id="7542
           in WPVDB_ID" description="..." />
39   </vulnerabilities>
40   <infrastructure id="CM-ID" description="Content management
         server ">
41     <compute id="CM-VM-1" name="Cmserver" architecture="" hostname
           ="" cores="" speed="" memory="" />
42   </infrastructure>
43 </securityMonitoring>
44
45 </macro>
46 </cloudService>
47 </cloudServices>
```

Listing 1: Security monitoring service description in ECSLA

4.4.2 Parameters. This section lists metrics, monitoring, and schedule parameters. Parameters are either simple or complex. Simple parameters are measured directly by counting (or other methods) while complex parameters are computed from simple parameters.

In Listing 2, Lines 2–17 define four simple metrics, namely TP, FP, TN, FN, and a complex metric C_{ID}. For the formulation of C_{ID} (Line 16), a reference is added to its description in Section 4.2. In practice, the service provider can build a resource file describing a metric computation process, and the SLA definition can refer to this document for metric computation. Such practice (referring to other official documents for support) is not uncommon. For instance, Amazon SLA refers to a customer agreement document to exclude some cases from the SLA.

In this example, verifications are scheduled three times per 24 hrs, and the minimum of the three values should satisfy the expected SLO (Line 18). A restricted time window per day is also specified to perform verifications (Line 19).

```
1  <parameters>
2     <metric id="TP" name="True positive" unit="count" type="simple
          ">
3        <description description="The number of correctly detected
             attacks" />
4     </metric>
5     <metric id="FP" name="False positive" unit="count" type="
          simple">
6        <description description="The number of legitimate inputs
             mistakenly classified as attacks" />
7     </metric>
8     <metric id="FN" name="False negative" unit="count" type="
          simple">
9        <description description="The number of attacks that are
             not detected" />
10    </metric>
11    <metric id="TN" name="True negative" unit="count" type="simple
          ">
12       <description description="The number of legitimate that
             are classified as legitimate" />
13    </metric>
14    <metric id="CID" name="Intrusion Detection Capability" unit=""
          type="complex">
15       <description description="..." />
16       <formulation> As described in Section 4.2 </formulation>
17    </metric>
18    <monitoring id ="Mon-1" statistic="min" window="24 hrs"
          frequency="3"/>
19    <schedule id ="Sch-1" start="8:00pm" end="8:00am"/>
20 </parameters>
21 /* End of Parameter section, start Guarantee section */
22 <guarantees>
23   <guarantee id="G-1">
24     <scope id="Sc1">
25       <service id="Mysql-SM-ID" subid="Mysql-SM-ID-mode"/> <
             service id="Apache-SM-ID" subid="Mysql-SM-ID-mode"/>
26       <service id="IL-SM-ID" subid="Mysql-SM-ID-mode"/> <service
             id="CCF-SM-ID" subid="Mysql-SM-ID-mode"/>
27     </scope>
28     <requirements>
29       <Requirement id="R1">
30         <Specification id="Sp1" policy="Required"> Base Rate(B), B
               >= 10^(-7) and B <= 0.1 </Specification>
31       </Requirement>
32     </requirements>
33
34     <terms>
35       <term id="T1" operator="">
36         <item id="CIDTerm"/>
37       </term>
38
39       <objective id="CIDTerm" priority="1" actor="provider">
40         <precondition policy="Required">
41           <description> The threshold value should be computed with
                 a function as described in Section IV.5 </
                description>
42         </precondition>
43         <expression metric="CID" comparator="gt" threshold="" unit
               ="" monitoring="Mon-1" schedule="Sch-1" confidence="
               95" fuzziness_value="0,05" fuzziness_percentage="5"/>
44       </objective>
45
46     </terms>
47   </guarantee>
48 </guarantees>
```

Listing 2: Example Parameters and Guarantees in ECSLA

4.4.3 Guarantees. In Listing 2, Lines 22–48 show the guarantee section that contains three subsections. (i) The *scope* (Lines 24–27) shows services that are covered under this guarantee (all four services in our example). (ii) The *requirements* (Lines 29–32) define base rate boundaries. The guarantee is for base rate values greater than 10^{-7} and less than 10^{-1}. If the base rate is not in this range, the SLO may not be achieved and such an incident is not an SLO violation. (iii) The *terms* (Lines 34–46) describe the expected metrics value, with their fuzziness and confidence ratio as well as the monitoring frequency and schedule.

In the example, the only term is the expected C_{ID} value. Accordingly, the expected C_{ID} value should be computed using the formula presented in Section 5. The next section describes why a formula is given, rather than an actual C_{ID} value. It also presents the process used for generating such a formula. To show an example usage for such a formula, let us assume that the NIDS is evaluated (or an attack occurs) with a base rate value of 7×10^{-2}. Putting this value into the formula gives an estimated value of $TPR \approx 0.71046$ and $FPR \approx 0$, and from these values, we can compute the expected C_{ID} value: $C_{ID} \approx 0.7162$.

The fuzziness value of 0.05 (5%) and confidence ratio of 95% are interpreted as: from 100 verification tests with a base rate value of 7×10^{-2}, in at least 95 of the tests, the configured NIDS must perform with $C_{ID} >= 0.7162$ and the remaining tests must perform with $C_{ID} >= 0.6562$. In other words 5% of the verification tests are allowed to perform to a 5% extent below the guaranteed performance level without violating the SLO. Permitting such limited fluctuations allows providers to promise their best-quality security monitoring at a reasonable risk.

5 INCLUDING UNKNOWN BASE RATE VALUES IN SLO

Axelsson [1] showed the importance of the base rate in evaluating the performance of NIDSs. However, predicting or calculating the exact value of the base rate in advance is difficult if not impossible. As a result, the SLA definition does not include base rate values to describe the performance of NIDSs. Instead, the SLA contains a model function which takes the base rate as an input to calculate the expected TPR and FPR values. This way we avoid an SLO which would be defined for a specific base rate and could thus be almost

never applicable. In addition, such a model gives guarantees even for previously unknown base rate values.

We use the C_{ID} metric in our SLO. The factors affecting the performance of an NIDS (i.e. the C_{ID} value) can be grouped into two categories: (1) *external factors* like the rate of the inputs (throughput), the base rate and available resources for the NIDS; (2) *internal factors* like the number of rules and the number of services that are configured to be monitored. Before describing the model generation method, we present the assumptions and challenges for designing such a method.

5.1 Assumptions and Challenges

If we are given a fixed number of services and vulnerabilities to be monitored, it is safe to assume that the internal factors affecting the performance of the NIDS are constant values. Indeed, since we have a fixed number of vulnerabilities to be monitored, we assume that the rules configured in an NIDS are constant. In addition, we assume that there are enough resources for the NIDS to perform the monitoring task. Hence, changes in the rate of the inputs (throughput) won't affect the performance of the NIDS by creating resource scarcity. The remaining factor affecting the performance of the NIDS is the attack rate or base rate.

In realistic production sites the value of the base rate is very small (i.e. close to zero). This means an attack packet happens very rarely compared to legitimate packets. When testing an NIDS, achieving such very low base rates is challenging and it requires an enormous amount of resources. For instance in [22], to achieve low base rates, attacks are injected very rarely and to have statistically sound results the injection is performed multiple times. Hence, the test takes a long time. In addition, logging packets for such a long time requires large disk space. The challenge exacerbates as the test needs to be performed for a large number of vulnerabilities.

We also assume that the SLA specifies the lowest guaranteed base rate value. In other words, the SLO may not be violated for attack occurrences below the specified base rate. However, such lower bounds should be as realistic as possible. Finally, we assume that the provider can perform the evaluation at the lowest guaranteed base rate value at least once.

5.2 Metrics Estimation Method

Figure 1 shows the correlation between B and C_{ID} for different FPR and FNR values. The plot assumes FPR and FNR as constant values. A close look at Figure 1 shows a trend between the C_{ID} value and different B values. Nevertheless, in practice configuring an NIDS to generate constant FPR and FNR values is difficult. This is because in a realistic operational environment the values of FPR and FNR are affected not only by its internal configuration but also by external non-deterministic factors, e.g. the base rate.

Following this observation, we performed experiments to identify if there was a correlation between B and (TPR, FPR). If such a correlation exists, we can have a model showing the correlation from known values of B and (TPR, FPR). Then we can use this model to estimate TPR and FPR values for unknown base rates. The estimated values can be used to compute the C_{ID} metric for those unknown base rates.

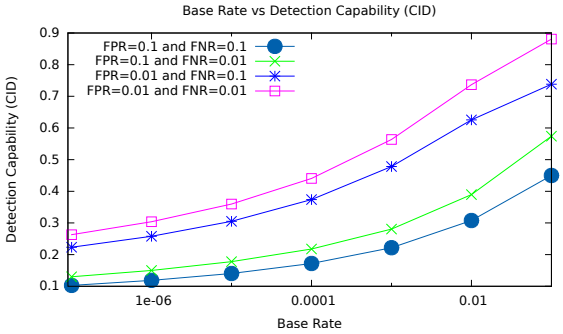


Figure 1: Base rate (B) vs C_{ID}

To generate the model we propose an interpolation-based method. The method takes known values of TPR and FPR which are computed for different B values. These values are used to generate a fitting function that can approximate the points. The generated function takes B as an input and produces TPR and/or FPR values. The function is used in the SLAs to estimate the TPR and FPR values for previously unknown B values and allows to compute C_{ID} values. In practice, such an estimation method may provide only an approximation of the exact value. Moreover, the margins described in ECSLA help to tolerate some degree of variation in the SLO.

Given a security monitoring configuration, to generate a representative formula which can be used to estimate the TPR and FPR values, we follow the following steps:

(1) Compute the performance of the NIDS on a given configuration taking a known base rate value as an input. For example, the procedure presented in [22] can be used. The performance evaluation should include the lowest guaranteed base rate and as many other points as possible. The accuracy of the model increases with the number of evaluation points.
(2) Using results from the performance test, find a function f of B that produces TPR and/or FPR.

This way we can generate an equation to be used in the SLA definition. The function f can be used to estimate the expected TPR and FPR values for new B values. The function f may not represent the exact relationship, but it is derived from the best information available. Section 7 presents an experimental evaluation showing the feasibility of our metric estimation method.

6 NIDS PERFORMANCE WITH A LARGE NUMBER OF VULNERABILITIES

We aim at defining SLAs to guarantee the performance of signature-based NIDSs. CSPs need to prepare SLA templates that will be offered to potential tenants at the start of the negotiation process. To achieve this, CSPs need to build a knowledge base on the performance of their security monitoring ability. Indeed, in a multi-tenant cloud setup each tenant has a separate virtual network and the NIDS applies a dedicated set of rules for each virtual network.

As discussed in Section 2, preparing SLO templates is challenging because there are thousands of public vulnerabilities and a tenant may choose any combination of those vulnerabilities.

Moreover, the set of services to monitor affects the effectiveness of the security monitoring process. In practice, signature-based

NIDSs use rules to detect attacks and more vulnerabilities means more rules to be added in the NIDS. Rules may interfere with each other and thus the performance of the NIDS may be lower than just summing the performance of each rule.

Given an NIDS to be configured with a set of rules, we propose a performance estimation method based on the pair-wise interference evaluated between rules. However, in practice evaluating the pair-wise interference between thousands of rules requires a large number of resource-costly NIDS tests. To reduce the required number of tests, we propose a clustering method which groups rules based on a given criterion. To our knowledge, this clustering method is the first attempt to evaluate an NIDS performance for large numbers of rules and any possible set of rules.

6.1 The Effect of Rule Interference on an NIDS Performance

The performance of an NIDS varies depending on the set of configured rules. Assuming all other factors affecting the performance of an NIDS are constant, the performance can be assumed as decreasing when the number of rules increases. Adding rules may lead to more false positives (e.g. a rule detects an attack in place of another one) as well as false negatives (e.g. the NIDS becomes too slow to process every input).

Testing an NIDS to measure the achievable performance for all the different sets of rules asked in SLA negotiation is infeasible. Indeed, for a given SLA negotiation, this process may last a long time and require significant resources, which both delay the negotiation and make it costly. The combinatorial explosion makes it also infeasible to anticipate and run such tests for every possible set of rules.

For these reasons, to get an accurate performance estimate in the process of SLA definition, we propose to rely on a quantified level of interference between rules, which the provider can measure without knowing the future requirements of the tenants and with a feasible number of tests. If a tenant selects k vulnerabilities out of n, the SLO offered for that tenant is the resulting estimated performance of the NIDS configured with the rules for those k vulnerabilities.

6.2 Rule Interference on NIDS

The interference between rules refers to the effect of one rule on the effectiveness of another rule. Before formally defining this notion, we explain the assumptions made.

6.2.1 Assumptions. NIDSs take rules as an input. The rules are mechanisms to tell the NIDS what to look for in the input packets. The detection engine of an NIDS applies the rules on each packet. If the packet matches a rule, the specified action of that rule is taken, and log(s) and/or alert(s) will be generated. However, if the packet matches more than one rule, the NIDS behaves in one of two ways:

- It generates an alert for all the rules matching;
- It generates an alert for few of the matches based on some heuristics (e.g. the first match, the most severe vulnerability…).

Usually, the second case is the default property, but even if it is not optimal, NIDSs can be configured to behave according to the first case. To simplify the formal definition of rule interferences we assume an NIDS configured to generate an alert for all matches (the first case).

6.2.2 Formal definition of interference. Let us assume we have two vulnerabilities to be monitored (V_i and V_j) with a set of rules for each of them to be configured in an NIDS. For a given base rate (B), an NIDS configured to monitor V_i will generate (TP_i, FP_i, TN_i, FN_i). Similarly for V_j, it generates (TP_j, FP_j, TN_j, FN_j). These are the basic metrics and they are measured by counting the number of attacks (or legitimate traffic) that are correctly (or incorrectly) classified by the NIDS from a given input. From these basic metrics we can calculate TPR and FPR for V_i and V_j as:

$$TPR_x = \frac{TP_x}{TP_x + FN_x} \text{ and } FPR_x = \frac{FP_x}{FP_x + TN_x} \tag{1}$$

Where 'x' is 'i' or 'j'. There is an interference between the rules related to vulnerabilities V_i and V_j (expressed simpler as "interference between V_i and V_j" thereafter) if the NIDS configured with V_i gives a quadruple value (TP_i, FP_i, TN_i, FN_i), the NIDS configured with V_j gives a quadruple value (TP_j, FP_j, TN_j, FN_j), and the NIDS configured to monitor both V_i and V_j gives a quadruple value $(TP', FP', TN', FN') \neq (TP_i{+}TP_j, FP_i{+}FP_j, TN_i{+}TN_j, FN_i{+}FN_j)$.

Note that an interference is just the change in values, it can be an increase or decrease in any of TP', FP', TN' and FN' from the sum of (TP_i, FP_i, TN_i, FN_i) and (TP_j, FP_j, TN_j, FN_j). We call an increase in TP' or TN' a *positive interference*. It indicates better effectiveness of an NIDS and such type of changes increases the C_{ID} value. However positive interferences are unlikely to happen.

Therefore, we are interested in *negative interferences* which indicate a degradation in the effectiveness of an NIDS and decrease the C_{ID} value. Negative interferences are errors made by the NIDS which result in increases in the value of FP or FN.

We can represent the interference between V_i and V_j as $(FP, FN)_{ij}$, defined as follows:

$$FP_{ij}{=}FP'{-}(FP_i{+}FP_j) \text{ and } FN_{ij}{=}FN'{-}(FN_i{+}FN_j) \tag{2}$$

With this definition TP', FP', TN', and FN' can be expressed as follows:

$$\begin{aligned} TP'{=}TP_i{+}TP_j{-}FN_{ij},\ FN'{=}FN_i{+}FN_j{+}FN_{ij} \\ FP'{=}FP_i{+}FP_j{+}FP_{ij},\ TN'{=}TN_i{+}TN_j{-}FP_{ij} \end{aligned} \tag{3}$$

Note that these formulas show the impact of increases of FP and FN on decreases of TN and TP.

If there are n vulnerabilities, the interference between V_i and the other $n-1$ vulnerabilities can be described with an *interference vector (IV)*:

$$IV{:}\ V_i = \{(FP, FN)_{i1}, (FP, FN)_{i2}, \ldots (FP, FN)_{in}\}$$

Interference vectors of n vulnerabilities form an *interference matrix* as shown in Table 2. An entry in the matrix shows an interference between the vulnerabilities in the corresponding column and row. The matrix is triangular and there is no interference for a vulnerability with itself.

Given the performance of an NIDS configured for each vulnerability alone and the interference matrix, we can estimate the performance of an NIDS for any combination of those vulnerabilities. Next, we present how to compute such values.

Vuln.	V_1	V_2	V_3	...	V_n
V_1	0	$(FP,FN)_{1,2}$	$(FP,FN)_{1,3}$	...	$(FP,FN)_{1,n}$
V_2		0	$(FP,FN)_{2,3}$	...	$(FP,FN)_{2,n}$
V_3			0	...	$(FP,FN)_{3,n}$
...				...	...
V_n					0

Table 2: Interference Matrix (IM)

6.3 Computing aggregated metrics

Let us assume we have an NIDS configured to monitor two vulnerabilities, V_i and V_j. We represent the TPR' and FPR' values for this NIDS as TPR_{agg} and FPR_{agg}. They are values computed from (TP', FP', TN', FN') and $(FP, FN)_{ij}$. We have two cases:

6.3.1 No interference between V_i and V_j (i.e., $(FP,FN)_{ij}=(0,0)_{ij}$). Using Equation 1 and Equation 3 TPR_{agg} and FPR_{agg} can be computed as follows:

$$TPR_{agg}=\frac{TP'}{TP'+FN'}=\frac{TP_i+TP_j}{TP_i+TP_j+FN_i+FN_j} \quad and$$
$$FPR_{agg}=\frac{TP'}{TP'+FN'}=\frac{FP_i+FP_j}{FP_i+FP_j+TN_i+TN_j} \tag{4}$$

6.3.2 Interference between V_i and V_j (i.e., $(FP,FN)_{ij}\neq(0,0)_{ij}$). I.e., there is a difference between the output of an NIDS configured to monitor both V_i and V_j and the sum of the separate outputs from V_i and V_j. We can compute TPR_{agg} and FPR_{agg} as:

$$TPR_{agg}=\frac{TP'}{TP'+FN'}=\frac{TP_i+TP_j-FN_{ij}}{TP_i+TP_j+FN_i+FN_j} \quad and$$
$$FPR_{agg}=\frac{FP'}{FP'+TN'}=\frac{FP_i+FP_j+FP_{ij}}{FP_i+FP_j+TN_i+TN_j} \tag{5}$$

If we have n vulnerabilities, we can generalize TPR_{agg} and FPR_{agg} as shown in Equation 6. In the equation, in addition to the assumptions described above, we assume that interferences are limited to pairwise interferences, i.e. the only interferences in the group are the ones in the pairs of rules.

$$TPR_{agg}=\frac{\sum_{i=1}^{n} TP_i-\sum_{i=1,j=1}^{n} FN_{ij}}{\sum_{i=1}^{n} TP_i+\sum_{i=1}^{n} FN_i} \quad and$$
$$FPR_{agg}=\frac{\sum_{i=1}^{n} FP_i+\sum_{i=1,j=1}^{n} FP_{ij}}{\sum_{i=1}^{n} FP_i+\sum_{i=1}^{n} TN_i} \tag{6}$$

It is important to note that we still have the issue to cope with a large number of vulnerabilities. The method described above may reduce the number of required tests to prepare SLO templates to $O(n^2)$ instead of $O(2^n)$ but $O(n^2)$ is still infeasible for thousands of vulnerabilities.

Moreover, this process is not a one time task: it may be required to do the test on different occasions. For instance, as new vulnerabilities are discovered the NIDS needs to be tested on those vulnerabilities. Hence, further reducing the number of performance tests is required for the feasibility of security monitoring SLAs.

6.4 Building a Knowledge Base by Clustering Rules

In order to reduce the required number of performance tests, we propose to cluster rules. The idea is to perform the evaluation test once per cluster of rules rather than once for the rules related to each vulnerability. The clustering mechanism is applied to the NIDS rules based on heuristics, and the resulting clusters are used to build a knowledge base for the performance of an NIDS.

To make it sufficient to measure the NIDS performance only per cluster, there should be only *negative interferences* between the rules of the cluster. With this assumption the result of a measurement is a *lower bound* for the performance of an NIDS configured with any subset of rules of the cluster. If there are *positive interferences* in the cluster, there may be a subset of rules in the cluster with which the NIDS could perform worse than with the whole cluster. For this reason, even if the chance of having a *positive interference* is very small, the provider should be careful while building the clusters and setting the lower bound values.

Subsequently, Table 2 will have a smaller height and width. The columns and rows will represent clusters of rules (C_i) instead of rules related to single vulnerabilities (V_i). CSPs can choose appropriate heuristics to cluster rules considering different factors. For example heuristics can cluster rules based on related applications (e.g. rules for OS vulnerabilities, browser vulnerabilities...) or based on applications that are used together (e.g. a single cluster with rules for the LAMP stack (Linux, Apache, Mysql, and PHP)) or based on the location where the application is deployed (e.g. clustering rules for login nodes together, storage nodes...). Having a minimal number of clusters results in meager SLO offers and having a large number of clusters requires huge amounts of tests. Section 7 presents a clustering example using real NIDS rules.

Having such clusters and a minimized interference matrix, it is possible to estimate a minimum expected performance for any given set of vulnerabilities. For a given set of vulnerabilities, either (i) the rules related to the selected vulnerabilities are in the same cluster and the performance of the NIDS on that cluster is offered as an SLO or (ii) the rules span several clusters and we compute the expected C_{ID} value using Equation 6.

It is possible to cluster rules using multiple heuristics. Each heuristic produces a matrix similar to Table 2 and, for a given set of vulnerabilities, the matrix which leads to the best performance estimate can be used to offer the SLO.

7 EVALUATION

We performed experiments to validate the proposed metrics estimation and rule clustering methods. In order to observe the realistic performance and behavior of given NIDS configurations, the experiments are based on real exploit payloads injection. To this end we use the measuring technique used to verify the correctness of an NIDS configuration in [22].

We present the setup for the experiments (Section 7.1) and the results, counted as basic metrics, to show the process of SLO template creation (Section 7.2). We also present an example of clustering based on some heuristics (Section 7.3).

7.1 Experimental Setup

We build a cloud infrastructure using the OpenStack [15] cloud software and Open vSwitch(OvS) [16] as a virtual switch. Figure 4(A) shows a high-level architecture of the experimental setup. The tenant infrastructure is configured to run the services described in Table 1. For the experiments, we configured a single tenant environment with three virtual machines (VMs) representing real

production VMs, running Apache, Mysql server and the WordPress content management system. To prevent injected attacks from disrupting the production VMs, a fourth VM is used as a *target* for the injected attacks. The target VM exhibits similar properties as the production VMs by running all the three services. The virtual switch is configured to forward all injected attack packets only to the target VM.

Snort [21], the most widely deployed IDS, is used as NIDS. It is deployed on a separate physical node and connected to the virtual switch through a mirror port. Snort is configured to look for instances of attacks listed in Table 1 and to output an alert for each matching rule. A traffic injector machine is located outside of the cloud and is used to inject attacks and legitimate traffic. We use real exploit programs to inject attacks against vulnerabilities in the target VM. The attacks are interlaced with legitimate traffic according to the base rate. In addition, we know the number of packets sent by each legitimate request and attack.

To measure the performance of a configured NIDS, we record all the injected packets. At the end of the attack campaign, we get the output of the NIDS (alerts). Using recorded inputs and the output alerts, we count the number of inputs that are correctly or wrongly classified as attacks or legitimate requests by the NIDS. This way we can count the TP, TN, FP, and FN values. More details about the setup and its justification are presented in [22].

7.2 Collecting Data Points and Generating a Model

Using the setup described above we performed experiments and Table 5 shows the results as *TPR, FPR, TNR, FNR*, and C_{ID} values for a given B. The values shown are averages over three rounds for each computation.

The experiment is performed for two sets of base rate values: small B values in $(0.001 - 0.01)$ and slightly higher B values in $(0.06 - 0.1)$. With such values a provider would assume a lower bound base rate value of 10^{-3}, i.e., the SLOs would not be violated if the NIDS underperforms for occurrences of attacks with $B < 10^{-3}$. It is important to note that in practice 10^{-3} is a relatively large lower bound. However, in our experimental environment, $B = 10^{-3}$ is the lowest achievable base rate value for the reasons described in Section 5.1.

Figure 2 shows the plot of TPR and FPR as functions of B for B in $(0.06 - 0.1)$. We use Table 5 and the corresponding graphs to model the relation between B and (TPR, FPR). From the plots, we observe that the FPR values are very small (≈ 0). Indeed the rules in the NIDS are carefully crafted for the set of considered attacks, which results in very few false positives. This property is consistent for different base rate values. Other studies [19] showed that given the attacks to be monitored, it is possible to craft an NIDS rule with a minimal FP value. Taking this into consideration, we could use a constant $FPR = 0$ value in SLAs. However, when negotiating an SLA providers should take the highest FPR value, as C_{ID} is sensitive to changes in FPR. Taking the highest FPR means promising a lower C_{ID} value, which puts the CSP in a better position to avoid violating the SLO.

For the TPR, we can observe that its value is increasing with B. To model their relation, we use the TPR values for B in $(0.06 - 0.1)$,

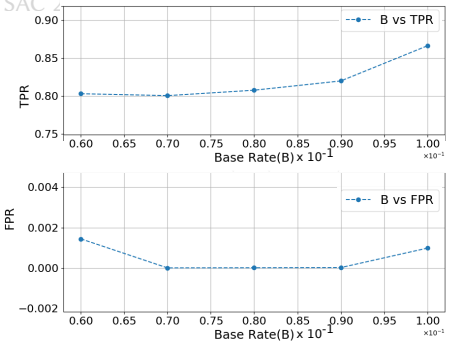


Figure 2: Plot of B vs *TPR* and B vs *FPR* for B in $(0.06 - 0.1)$

and a TPR at the lowest guaranteed base rate, i.e., $B = 10^{-3}$. We fit these points using a quadratic polynomial function, because there is only one local extremum value in the given range of B. The quadratic function f which best approximates these points is:

$$f(B) = 0.7008827 + 1.357906 * B + 1.447843 * B^2 \quad (7)$$

Using this function, we can estimate the values of the expected TPR for other base rate values. As an example, Figure 3 shows the estimated and actual measured TPR values for B in $(0.003 - 0.009)$. From this data, we can observe that our metric estimation method produces results which are close to the actual values.

7.3 Clustering NIDS Rules

In Section 6.4 we presented a clustering technique that can be used to reduce the dimensions of an interference matrix, thus reducing the number of required evaluation tests to prepare the template SLOs. In this section, we present an example using rules from the Snort NIDS.

We used the official registered rule set version 2.9.8.0 [21], which contains a total of 10628 rules in 53 files, to demonstrate the proposed clustering technique. Several properties can be used to cluster the rules, including the *application type*, the *attack class type*, the *severity of the vulnerability*, and *applications working together*. We present a clustering using the *application type* property and other examples can be found in [23].

Clustering based on application type is a method of putting applications with a similar type of functionalities into the same cluster, e.g., putting rules for vulnerabilities in browsers and operating systems together in one cluster. By default, rules in the Snort repository are grouped per application, e.g., Firefox rules, Internet Explorer rules, office file rules, etc. This default grouping can be used as

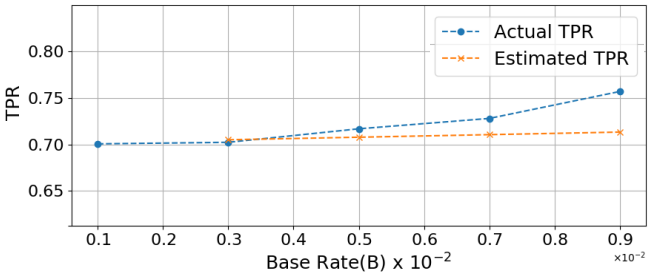
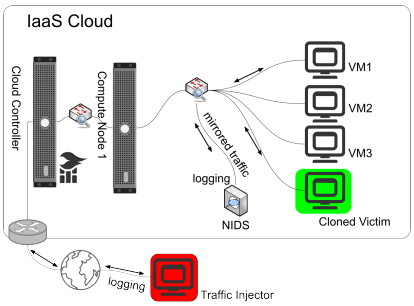


Figure 3: Plot of the estimated and actual TPR values for B in $(0.001 - 0.009)$

(A)



(B)

Application Group	Applications (number of rules)	Total
Browsers	Firefox (15), Internet explorer (1286), Webkit (2), Others (8), Plugins (31)	1339
Files	Executable (17), Flash (1506), image (125), java (110), multimedia (55), office (462), pdf (506), others (331), file-identifier (979)	4091
OS	Windows (388), Solaris (3), mobile os (3), others (36), netbios (22), Linux (15)	467
Protocols	imap (2), scada (10), telnet (3), dns (1), pop (1), snmp (1), voip (2), others(4)	24
Servers	Apache (27), mail (9), mysql (1), oracle (2), samba (6), web-apps(273), mssql(1), others (223)	542
Potentially Unwanted Application (PUA)	Adware(30), p2p(3), toolbars (4), others (21)	58
Malware	Backdoor (110), CNC (3053), malware-tools (14), others (336)	3513
Others	app-detector (2), exploit-kit (492), indicator-compromise (30), indicator-scan (2), policy-social (2), policy-others (15), sql (16), deleted(3), indicator-obfuscation (32)	594

Figure 4: (A) Experimental setup and (B) Rule clustering based on application types

	B	TPR	FPR	TNR	FNR	CID
Small B values	0.001	0.7005733333	0.000035352	0.9999566667	0.2994266667	0.6453533333
	0.003	0.7022433333	1.23E-07	0.9999966667	0.2977566667	0.6493066667
	0.005	0.7167533333	0.00000026	0.9999966667	0.26614	0.66004
	0.007	0.7278733333	8.63E-07	0.9999966667	0.2721233333	0.66845
	0.009	0.7567866667	0	1	0.2432133333	0.6968933333
	0.01	0.7338044444	8.12E-07	0.999996667	0.23843	0.700916667
Slightly higher B values	0.06	0.80297	0.001438933	0.99856	0.197023333	0.704433333
	0.07	0.80046	0	1	0.1995366667	0.71623
	0.08	0.8076033333	6.53E-06	0.9999966667	0.19239	0.73025
	0.09	0.8198066667	1.59E-05	0.9999833333	0.18019	0.76141
	0.1	0.8661833333	0.0009814231	0.9990166667	0.13381	0.7737933333

Figure 5: TPR, FPR, TNR, FNR values of an NIDS from our experiment and calculated C_{ID} value for varying B

a base for our clustering purpose and our clustering method reduces further the number of groups. Figure 4(B) shows an example of clustering created using the application type which requires a maximum of 64 experiments.

8 CONCLUSION AND FUTURE WORK

We presented a method allowing CSPs to define SLAs providing each tenant with guarantees about the performance of an NIDS configured according to the tenant's requirements. The first key component of this method is ECSLA, an SLA language suitable for security monitoring and based on an SLA language for clouds. ECSLA introduces a new generic service, a structure to define security vulnerabilities, and a definition of security monitoring service. The KPI in our SLO takes the base rate parameter into account. Moreover, to avoid restricting SLOs to fixed base rate values, we presented a novel approach for KPIs with unknown parameter values, consisting in defining model-based KPIs instead of scalar values. To facilitate this process we presented a method to quantify the interference between NIDS rules. The second key component is an efficient knowledge-base building method for the performance of an NIDS. To our knowledge, this method is the first attempt to evaluate an NIDS performance for large numbers of rules and any possible set of rules. Finally, we showed an experimental evaluation on how to prepare the model and vulnerability clustering method in order to reduce the number of required NIDS tests to build the knowledge base.

As future work, first we should address limitations in this paper. We did our experiments only on single tenant setups. In multi-tenant use cases we should study the performance issues, especially the effects of resource contention like memory. We also plan to experimentally validate our theoretical work on rule interference. Second, our approach should be extended in two directions. Users are expected to describe their needs in terms of software vulnerabilities. However, tenants have no detailed information about the vulnerabilities of their software. A higher level language to describe tenants needs is required. Finally, to address use-cases of full-featured security monitoring systems, the SLA definition mechanism should be studied for monitoring devices other than NIDSs.

REFERENCES

[1] Stefan Axelsson. 1999. The base-rate fallacy and its implications for the difficulty of intrusion detection. In *6th ACM CCS*.
[2] Pierre-Marie Bajan et al. 2019. Methodology of a network simulation in the context of an evaluation: application to an IDS. In *Proc. ICISSP*.
[3] Richard Bejtlich. 2004. *The Tao of network security monitoring: beyond intrusion detection*. Pearson Education.
[4] Karin Bernsmed, Martin Gilje Jaatun, Per Hakon Meland, and Astrid Undheim. 2011. Security SLAs for federated cloud services. In *IEEE ARES*.
[5] Karin Bernsmed, Martin Gilje Jaatun, and Astrid Undheim. 2011. Security in service level agreements for cloud computing. In *CLOSER*.
[6] Waleed Bul'ajoul et al. 2015. Improving network intrusion detection system performance through quality of service configuration and parallel technology. *J. Comput. System Sci.* (2015).
[7] Valentina Casola et al. 2017. Automatically enforcing security SLAs in the cloud. *IEEE TSC* (2017).
[8] Valentina Casola, Alessandra De Benedictis, and Massimiliano Rak. 2015. Security monitoring in the cloud: an SLA-based approach. In *IEEE ARES*.
[9] CVE 2020. Common Vulnerabilities and Exposures. https://cve.mitre.org/ accessed July 2020.
[10] Carlos Alberto Da Silva and Paulo Licio de Geus. 2014. An approach to security-SLA in cloud computing environment. In *IEEE LATINCOM*.
[11] Guofei Gu et al. 2006. Measuring intrusion detection capability: an information-theoretic approach. In *ACM ASIACCS*.
[12] Hannes Holm. 2014. Signature based intrusion detection for zero-day attacks:(not) a closed chapter?. In *47th HICSS*. 4895–4904.
[13] Nesrine Kaaniche, Mohamed Mohamed, Maryline Laurent, and Heiko Ludwig. 2017. Security SLA based monitoring in clouds. In *IEEE EDGE*.
[14] Yousri Kouki and Thomas Ledoux. 2012. CSLA: a Language for improving Cloud SLA Management. In *CLOSER*. 586–591.
[15] Openstack 2020. Open source software for creating private and public clouds. https://www.openstack.org/ accessed July 2020.
[16] OVS 2020. Open vSwitch is a production quality, multilayer virtual switch. https://www.openvswitch.org/ accessed July 2020.
[17] Dana Petcu and Ciprian Craciun. 2014. Towards a Security SLA-based Cloud Monitoring Service.. In *CLOSER*. 598–603.
[18] Thibaut Probst et al. 2015. Automated Evaluation of Network Intrusion Detection Systems in IaaS Clouds. In *Proc. EDCC*.
[19] Elias Raftopoulos and Xenofontas Dimitropoulos. 2013. A quality metric for IDS signatures: in the wild the size matters. *EURASIP* (2013).
[20] Massimiliano Rak et al. 2013. Security as a service using an SLA-based approach via SPECS. In *CloudCom*.
[21] Snort 2020. An open source intrusion detection and prevention system. https://www.snort.org/ accessed July 2020.
[22] Amir Teshome, Louis Rilling, and Christine Morin. 2018. Verification for security monitoring SLAs in IaaS clouds: The example of a network IDS. In *NOMS IEEE/IFIP*.
[23] Amir Teshome Wonjiga, Louis Rilling, and Christine Morin. 2019. *Defining Security Monitoring SLAs in IaaS Clouds: the Example of a Network IDS*. Research Report RR-9263. Inria Rennes Bretagne Atlantique.