Building a Compliance Vocabulary to Embed Security Controls in Cloud SLAs

Matthew L. Hale Rose Gamble
Tandy School of Computer Science
University of Tulsa
Tulsa, OK, USA
{matt-hale, gamble}@utulsa.edu

***Abstract*—Mission critical information systems must be certified against a set of security controls to mitigate potential security incidents. Cloud service providers must in turn employ adequate security measures that conform to security controls expected by the organizational information systems they host. Since service implementation details are abstracted away by the cloud, organizations can only rely on service level agreements (SLAs) to assess the compliance of cloud security properties and processes. Various representation schema allow SLAs to embed service security terms, but are disconnected from documents regulating security controls. This paper demonstrates an extensible solution for building a compliance vocabulary that associates SLA terms with security controls. The terms allow services to express which security controls they comply with and enable at-a-glance comparison of security service offerings so organizations can distinguish among cloud service providers that best comply with security expectations. To exemplify the approach, we build a sample vocabulary of terms based on audit security controls from a standard set of governing documents and apply them to an SLA for an example cloud storage service. We assess the compatibility with existing SLAs and calculate the computational overhead associated with the use of our approach in service matchmaking.**

Keywords-cloud; security; web services; service level agreement; certification; compliance; xml;

I. INTRODUCTION

Cloud computing and service oriented architectures (SOA) provide a number of opportunities for advancing organizational goals of increased service delivery efficiency, dynamic allocation of resources, and decreased IT costs [1]. Operating mission critical systems, such as military and health care systems, requires strict security considerations. Such systems must comply with a set of *security controls* [2-6] that implement information security policies, govern system operations, and protect sensitive data and functionality. Care must be taken when moving resources to the cloud to ensure that cloud service providers (CSPs) have adequate security services in place that conform to the expected security control implementations.

Service Level Agreements (SLAs) provide sets of contractual *service terms* [7, 8] that describe the functionality and security offerings of a service without the blackboxed implementation details, which are privy only to the service provider. Although various approaches have improved the ability of SLAs to expose and advertise their security services [9-13], selecting the provider that is most compatible with organizationally selected security controls is still ad-hoc [14]. The main problem is that security controls lack representation in cloud services [14, 15], making it difficult to connect organizational certification efforts to the services and security features being used by the cloud.

In this paper we define a *compliance vocabulary* that creates a set of security SLA terms that are derived from security controls in governance documents such as the NIST-SP800-53 [2], the Common Criteria Part 2 [3], the DoD 8500 [4], the DISA Secure Application Security Technical Implementation Guide (STIG) [5], and the Cloud Security Alliance Cloud Control Matrix (CCM) [6]. Each compliance *term* consists of an identifier and one or more controls that substantiate it. Existing services would rely on the compliance vocabulary to represent the controls it must satisfy and embed the corresponding terms in its SLA. Advertising security control compliance services can be economically advantageous for CSPs to attract organizations with strict compliance requirements that may not otherwise use cloud services. For the organization, the compliance vocabulary allows for at-a-glance comparison of service offerings and shifts some certification burden to the CSP based contractual SLA terms.

Our approach constructs an XML schema to define a structure for identifying vocabulary terms based on established security controls. Using the schema, we produce a set of terms and *categories* of similar terms to form an *ontology*. Ontologies [7, 8, 16-18] are re-usable structures that allow services to communicate the semantics of their SLAs [7, 16-18] facilitating automated selection and negotiation [7, 8] during service matchmaking. We define a second XML schema that dictates how terms are structured and plugged into existing security SLAs. The resulting structure is easily parseable, allowing matchmaking algorithms to determine which elements of the ontology (i.e. which vocabulary terms) are being offered by the CSP. Our approach is demonstrably compatible with existing SLA frameworks including WS-Agreement [9, 14, 19] and WSLA [13, 14, 20, 21].

The rest of the paper is organized as follows. Section II reviews relevant background on compliance in cloud services, security SLAs, schemas, and ontologies. Section III describes the compliance vocabulary, how vocabulary terms are used in services, and a running example based on audit controls from the governance documents. In Section IV we assess the compatibility of our approach with existing security SLAs, discuss the economic incentives for service providers to provide certification information, and calculate the computational overhead associated with parsing and comparing vocabulary terms. Section V concludes the paper.

978-0-7695-5024-4/13 $26.00 © 2013 IEEE
DOI 10.1109/SERVICES.2013.27

II. BACKGROUND

A. Compliance and Certification

Compliance and certification is an important consideration in organizational selection of cloud services [22]. Cloud service compliance research has focused on SLAs [9, 14, 15, 22] and service matchmaking [9, 14], security requirement assessment of CSPs [15], and the automation and validation of service configurations [22]. These approaches underscore the importance of service compliance with sets of security requirements, which may be expressed as security controls or formal statements.

In this work, we examine standard governing documents: NIST-SP800-53 [2], the Common Criteria (CC) Part 2 [3], the DoD 8500 [4], the DISA Secure Application Security Technical Implementation Guide (STIG) [5], and the Cloud Security Alliance Cloud Control Matrix (CCM) [6]. Each document contains a representation of security controls. The NIST sp800-53 [2] is the most relied on. It structures the controls across three levels, starting at the top with the *control family*. There are 18 families, such as audit, access control, and contingency planning. Each family contains uniquely numbered *security controls* describing a particular security requirement. *Control enhancements*, at the lowest level, decompose the control content. For example, AU-5.E1 identifies control 5, enhancement 1 in the audit family (AU). The CCM [6] is based on the SP800-53. The DoD 8500 [4] and the DISA STIG [5] identify *subject areas* and outline guidelines for each. The CC Part2 [3] follows a component-based representation with *functional classes* (similar to NIST families), *functional families* (similar to NIST controls), and *components* which are akin to control enhancements.

In previous requirements engineering research [23], we construct a hierarchical compliance model that allowed security controls from all of the governing documents to be extracted, formally expressed, and reasoned over. We structure controls into equivalence classes based on a common *security intent*. All of the controls in a single intent are semantically related according to the core security requirements relevant for certification efforts. One important result of our study shows a dominant control emerges among controls grouped according to their security intent. The expression of this control is related directly or transitively to all other controls in the group. We rely on this result by including a field for the dominant control in our schema.

Spanning the various regulatory documents is a security life cycle outlined in the NIST SP800-37 [24] and SP800-53 [2] that directs organizational security certification efforts. The first step in the lifecycle classifies organizational information systems according to their assets and goals. The second step selects applicable security controls from governing security documents based on the system goals and types of assets. In step three, the security controls are implemented in the system and then assessed, in step four. Steps five and six authorize the deployment of the system and then monitor it for continuing compliance. Given the transition of information systems to the cloud, some of the certification effort falls to CSPs. Thus, adapting the lifecycle to the cloud requires services to advertise their certification criteria as SLAs, which is currently ad-hoc at best [14].

B. Security SLAs

Security in SLAs started with identified types of quantifiable security metrics applicable to SLAs [25]. Irvine and Levin [26] expanded on these metrics coining the term 'QoSS' for quality of security service. Based on QoSS, Lindskog [12] defined four dimensions that characterize a "tunable security service": type of protection service (e.g. confidentiality), protection level (e.g. number of assets that must be encrypted), protection level specification (i.e. the security policy), and adaptiveness (i.e. the ability of a service to change protection levels at run-time). Bernsmed, et al., [11] examine SLA management for federated cloud services by developing a framework that supports a security SLA lifecycle with 6 phases: *publishing, negotiation, commitment, provisioning, monitoring, and termination*. Their negotiation architecture consists of a customer's security requirements, an initial CSP with a security offer, and third party CSPs that may meet one or more of the terms of the security offer [11].

C. XML Schema and Ontologies for SLAs

Several frameworks exist for specifying security SLAs. WS-Agreement standards [19] define an XML schema and protocol to advertise service capabilities and create SLAs. It provides a set of XML terms (e.g. the core *wsag:Template*) for specifying *service terms*. *Service Description Terms* (SDTs) are service terms that can contain Web Service Definition Language (WSDL) statements or XML to define the offered or requested operations that the service can perform. A set of *service properties* are defined over the SDTs to quantify the service performance. *Guarantee Terms* (GTs) contain *Service Level Objectives* (SLOs) that define quality assurances based on the service properties. *Business Value Lists* qualify the SLOs from a business perspective and may include actions to be taken if an SLO is not met.

Another XML-based framework called WSLA [13, 20] defines SDTs on top of a service's WSDL. WSLA allows a set of *SLAParameters, resource metrics* and *composite metrics* to be defined over the service to quantify the quality-of-service (QoS) constraints that it should satisfy. The *SLA parameters* represent hard numbers quantifying the service performance, in terms the customer can understand, such as "meet/exceed/fail" [13]. Finally, *business metrics* attach financial valuations to SLA parameters when customers establish the contractual parameters that bind to CSP performance. These business metrics come in the *obligations* section which allows SLOs and *action guarantees* to be defined that assure the customer of a certain level of service with defined financial penalties should the SLA be violated.

In our previous work [9, 10], we advance SLA research by developing a schema, based on WS-Agreement [19], that is capable of expressing security requirements in a standard way across service providers. We define an associated matchmaking algorithm that compares security requirements to identify the security offerings best suited to a particular organization. In [10], we improve risk evaluations in the schema and develop a renegotiation protocol capable of

renegotiating security SLA terms if service providers fail to meet their contractual guarantees specified in their SLAs. We show that the algorithm can minimize the risk to the organization by selecting alternate services compatible with the failing service. Missing in these approaches are direct compliance assurances connected to the service terms that provide certification guarantees for specific governing documents. This paper will address this need in a way that is both compatible with our previous work, commonly used WS-Agreement [19], WSLA-based approaches [13, 20, 21] and framework agnostic approaches such as in [14].

Research has examined the usefulness of ontologies for structuring service terms for matchmaking [7, 8], service interface compatibility [16, 17], and standard QoS expression [18] in the cloud. For instance, Dobson, et al., [18] assess various WSLA and WS-Agreement based ontologies to form a unified ontology of QoS and SLA terms that facilitate service selection based on available QoS offerings. Belhajjame, et al., [16] use an ontological approach to verify service semantic annotations to ensure that input and output parameters are correctly designed. Most relevant is work by Redl, et al., [7] on automatic service and CSP matching. Their approach semantically compares SLA elements using an ontology-like method to parse, map, and translate SLA elements into an interchangeable format amenable to element-by-element comparison. The goal is to understand each service offering based on its SLA terms and select the one that best fits a set of input.

III. Compliance Vocabulary

To improve the state of cloud security compliance we introduce a *compliance vocabulary* that defines a set of security SLA terms based on security controls from the governing documents. Each SLA *term* consists of an identifier and one or more controls it reflects. The next sections discuss how vocabulary terms are constructed and arranged hierarchically into multiple trees of an *ontology* and how the ontology can be used in SLAs to express service security control compliance.

A. Building a Compliance Vocabulary

The first step in constructing an ontology for security compliance involves the use of control families in NIST sp800-53, such as access control or audit, as a basis for segregating the high level controls across governing documents. Within each family, the controls can be further partitioned into *control groups* such as least privilege and separation of duties in access control or auditable events and audit failure in audit. Within each group, individual controls are segregated into *control categories,* which provide a finer granularity of qualification within the group.

Fig. 1 shows the basic structure of a compliance vocabulary for a tree representing a particular control group. The ontology is formed by a set of these trees. The root of each tree, represented as a bold and rounded black box in Fig. 1, is a *security control group term*, referred to simply as a *group term*, that is singularly associated with that control group. The leaf nodes of each tree are called *control terms*. Between the root and the leaves are various *category terms* that correspond to the control categories and decompose the group ontologically. Terms in the compliance ontology trees are referenced by their unique *identifiers*. Each identifier is a short phrase with no spaces, such as *has-audit-system,* that expresses the overall description of the group (*group identifier*), category (*category identifier*), or control (*control identifier*) it represents.

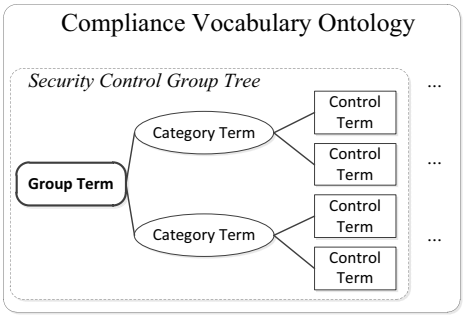

Figure 1: Structure of a Compliance Vocabulary

A *control identifier* is associated with a control term node in the security control group tree, as seen in Fig. 1. Control terms represent decomposed requirements of one or more security controls. Its identifier denotes one or more governing documents and associated controls to indicate the basis for the requirement that becomes the control term. For instance, the control AU-5.E3 (control ID) in the SP800-53 (document ID) [2] states that the system "[Selection: rejects or delays] network traffic above (defined) thresholds" as related to audit failure. This suggests the creation of three control terms, *accept-network-traffic* (by default allows traffic), *prioritize-network-traffic* (delays traffic), and *reject-network-traffic* (outright rejects traffic). These control terms are grouped into the *network-actions* category of the *audit-processing-failure* control group (shown later in Fig. 3).

C*ategory terms,* corresponding to the control categories, structure control terms ontologically and describe a particular facet, e.g. *network-actions,* of the control group. A category term is defined by its *category identifier* and may appear in only one group. At the root of each tree, a *group term* consists of a *group identifier* and an optional dominant security control [23] that may be representative of the group. The dominant security control is encoded as a requirement within the group term. Fig. 2 shows the graphical representation of the different terms.

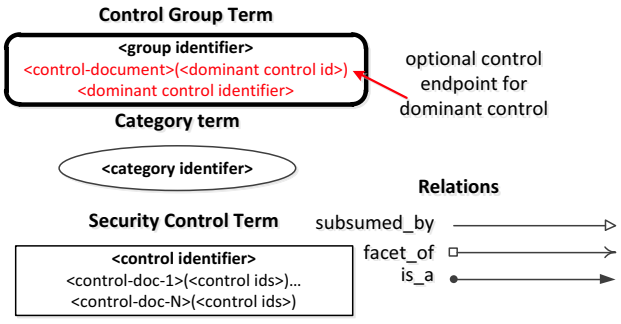

Figure 2: Ontological Building Blocks

Three semantic relations are allowed between terms (Fig. 2). The first two, *is_a* and *subsumed_by*, reflect their common use in ontologies with respect to the term relationships. For example, whenever a service advertises a control term that subsumes another term in the ontology, it can safely be assumed that the service satisfies the subsumed security control(s) in addition to the control(s) associated with the advertised term. The relation *facet_of* is a type of 'part-of' relationship in which a category represents a portion of the concerns related to a group term. A legend identifying the arrow used for each relation is included in Fig. 2. As an illustration, we show a sample compliance vocabulary in the ontology with two groups of controls previously analyzed [23] for auditable events and audit failure respectively.

For the first control group, the identifier is *auditable-events*. AU-2.G.a, from the SP-800-53, is a dominant control and three main categories best partition the security controls in the group. AU-2.G.a broadly states that the overall system needs an audit system, inducing the control term *has-audit-system* to represent this requirement. The three categories of this group are *auditable-actions,* which combines security controls that define particular system actions that must be audited, *log-properties*, which denotes the various attributes (such as user ID and IP) that must be recorded in the audit log, and *available-audit-detail-levels* which qualifies the amount of information to be captured in the audit log. A subset of the controls related to log-properties specifically focus on login, causing the creation of a sub-category *login-information* to classify these controls. Each category is linked to the group term using *facet_of*. Control terms are all grouped into a category via the *is_a* relation. Since *audit-detail-level* is one of (*not-specified, minimum*, *basic*, *detailed*) *subsumed_by* conveys the connection among the property ranges. Though all levels derived from a single control, *subsumed_by* can be applied between control terms relating to any control endpoint. The resulting *auditable-events* tree appears in Fig. 3a.

To consolidate the controls related to audit failure, we created the control group term *audit-processing-failure*. A dominant control for this group is AU-5.G.b in the SP-800-53 [23]. It requires the system to deploy an audit failure system, inducing the control term *has-audit-failure-system* to denote this core requirement. There are two main categories in this group: *handled-failures,* to denote the audit failure events that can be handled by the audit failure system, and *performs-failure-actions,* which identifies the available actions that can be performed by the failure system when one of the handled failures occurs. A subcategory called *network-actions* identifies actions the audit failure system can take to handle network failure events. Fig. 3b shows the *audit-processing-failure* tree of vocabulary terms.

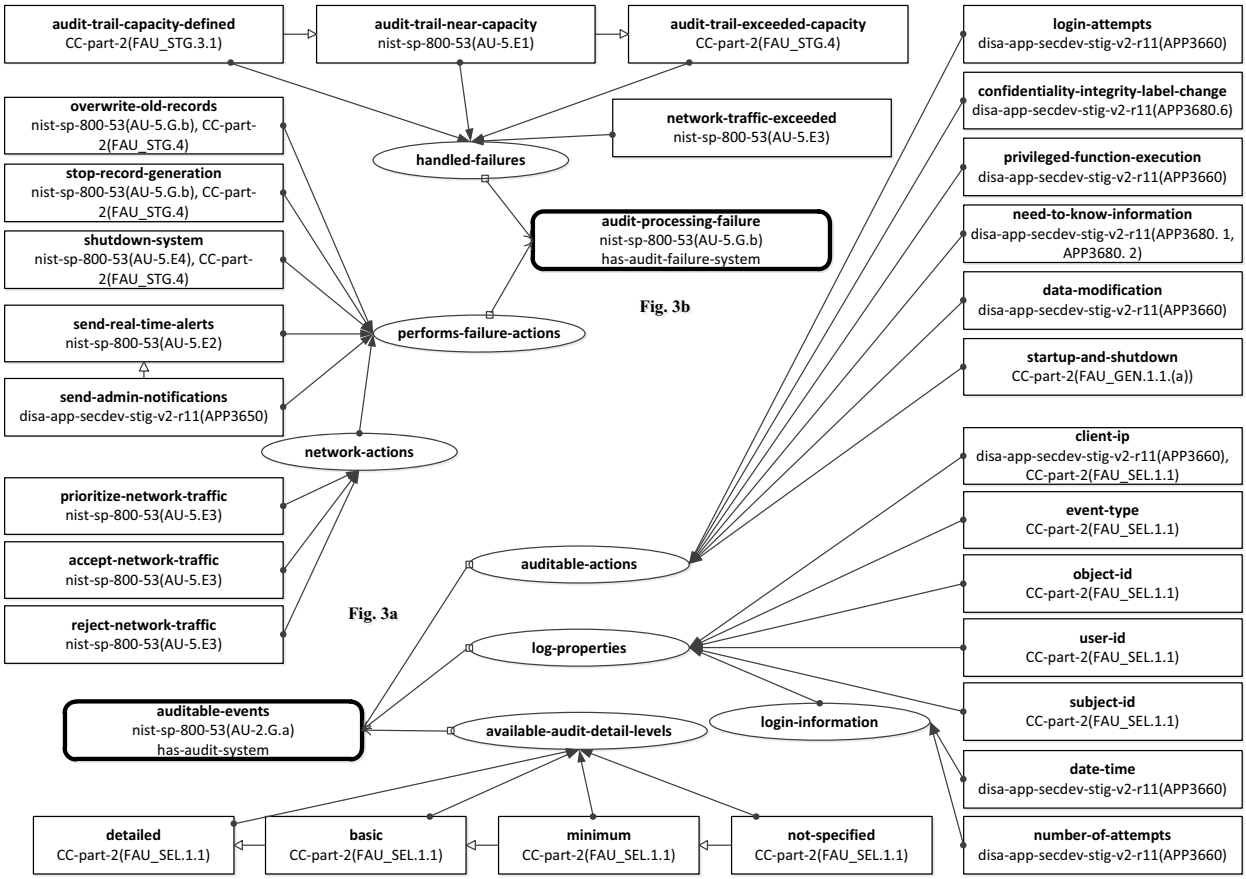

Figure 3: Compliance Vocabulary of Terms including 3a) *auditable-event* tree, 3b) *audit-processing-failure* tree

B. XML Representation

Underlying the compliance vocabulary is an XML schema that defines the formation of XML constructs to house the ontology data. The top level structural construct is the *compliance-vocabulary* XML `xs:element` [27]. The *compliance-vocabulary* holds one or more *control-group* trees that identify a control group term and express all of its related control terms. A control-group element has an `xs:complexType` [27] that consists of a *term,* which is an `xs:attribute` [27] matching the group identifier, a *dominant-control*, which is an `xs:element` that identifies the dominant control, a set of *category* elements, which are of type `xs:element` and match each of the category terms in the tree, and a set of *control* elements, identifying any uncategorized security controls. Note that there must be at least one category or control element in a *control-group* .

```
<compliance-vocabulary>
  <control-group
    term="group identifier">
      <dominant-control/> ?
      (<category/> | <control/> |<controls/>) +
      <relations>
        <relation/> +
      </relations>
  </control-group> +
</compliance-vocabulary>
<category term="category identifier">
    (<control/> | <controls/>) +
</category>
<dominant-control
  term="control identifier">
    <doc>dominant control document</doc>
    <id>dominant control id</id>
    <desc>optional term commentary</desc> ?
</dominant-control>
<control
  term="control identifier">
    <doc>control document</doc>
    <id>control id</id>
    <desc>optional term commentary</desc> ?
</control>
<controls
  term="control identifier">
    <control/> +
    <desc>optional term commentary</desc> ?
</controls>
<relation
  type="control identifier"
  from ="originating element"
  to ="destination element/>
```

Figure 4: XML Structure of Compliance Vocabulary

The *control* and *dominant-control* elements both have an `xs:complexType` that consists of a *term* `xs:attribute`, matching the control identifier in the tree, a *doc* `xs:element` that holds the document identifier, and an *id* `xs:element` that holds the control identifier. An optional *desc* field is included for additional commentary about the security requirements the term represents. This optional field can facilitate at-a-glance comparison by service providers, since they do not have to look back at the control documentation to understand the embedded operational information in the requirements. Another `xs:element` called *controls* is used to group multiple controls into a single term. A *category* element has an `xs:complexType` that consists of a *term* `xs:attribute` matching the category identifier in the tree, and one or more *control* or *controls* `xs:elements`.

A *relation* is a *complex empty element* [27] that contains three attributes: *type* for a relation type, *from* for the originating entity, and *to* for the destination entity. Fig. 4 displays the structure of the XML generated by the schema. We adhere to traditional XML schema notation, where '+' means "one or more," '?' means "at most one or none," '*' means "zero or more," '|' means "or," and parenthetical infix notation must be evaluated before any postfix notation. That is, (item1 | item2) + means one or more of item 1 or item 2, e.g. the elements item1, item1, item2.

The schema generates valid XML to form the ontology based on our control group trees. Given space constraints, we show a snippet of XML for the *audit-processing-failure* tree, (Fig. 3). Fig. 5 includes the control group term followed by the dominant control AU-5.G.b, its two categories, and a sampling of the relationships between entities. We partially expand the *handled-failures* category to show how control terms are represented.

```
<control-group term="audit-processing-failure">
  <dominant control
    term="has-audit-failure-system">
      <doc>nist-sp-800-53</doc>
      <id>AU-5.G.b</id>
  </dominant control>
  <category term="handled-failures">
    <control term="audit-trail-near-capacity">
      <doc>nist-sp-800-53</doc>
      <id>AU-5.E3</ id>
    </control>
    <control term="audit-trail-capacity-exceeded">
      <doc>common-criteria-part2</doc>
      <id>FAU_STG.4.1</ id>
    </control>
    …(other controls)…
  </category>
  <category term="performs-failure-actions"/>
  <relations>
    <relation type="facet_of"
      from="handled-failures"
      to="audit-processing-failure"/>
    <relation type="is_a"
      from="audit-trail-near-capacity"
      to="handled-failures"/>
    <relation type="subsumed_by"
      from="audit-trail-near-capacity"
      to="audit-trail-capacity-exceeded/>
    …(other relations)…
  </relations>
</control-group>
```

Figure 5: Snippet of XML for *audit-processing-failure* ontology

C. Using the Compliance Vocabulary in Services

Expressing the vocabulary in XML is not enough to host a secure service on the cloud. In order to make the cloud know what is needed, the vocabulary terms must be deployed into SLAs for direct comparison among CSPs. Determining which terms a service satisfies and should include in its SLA requires examining the compliance vocabulary and then assessing the service offering to

determine if it satisfies the control requirements associated with each term. The *desc* and term names can also direct the assessment process without looking at the controls themselves. Deploying the terms to service SLAs requires an additional XML schema to encode and include vocabulary terms in SLAs. Such a schema need only provide the group, category, and control identifiers since their connotation can be understood by parsing the compliance vocabulary to find the associated security requirements information. To ensure the schema produces XML compatible with WSLA and WS-Agreement SLAs, we structure it as a self-contained block that can be plugged into SDTs, to identify the security control compliance of a certain component in the service. It can also be included in the SLA as a top-level SLO, denoting that the all service components comply with the control requirements associated with the compliance terms. Fig. 6 shows the XML structure produced by our schema.

```
<compliance-vocabulary-terms>
  <con-group-term
    term="group identifier">
      <dom-control-term/> ?
      (<cat-term/> | <control-term/>) +
  </con-group-term> +
</compliance-vocabulary-terms>

<dom-control-term
  id="dominant control identifier"/>

<cat-term id="category identifer">

<control-term id="control identifier"/>
```

Figure 6: Schema for expressing Compliance Vocabulary in Services

This structure simply lists the terms from the vocabulary that the service complies with. At the top is the `compliance-vocabulary-terms` element. This element is the top level wrapper that houses the vocabulary terms from all of the control group trees. Each ontology used by the service is identified by a `sec-group-term` element that contains a term identifier, an optional dominant control (i.e. `dom-control-term`) and one or more category terms (i.e. `cat-term`) or control terms (i.e `control-term`).

```
<compliance-vocabulary-terms
  <con-group-term id="auditable-events">
    <dom-control-term id="has-audit-system"/>
    <cat-term id="available-audit-detail-levels">
      <control-term id="detailed"/>
    </cat-term>
    <cat-term id="auditable-actions">
      <control-term id="login-attempts"/>
      <control-term id="data-modification"/>
      <control-term id="startup-and-shutdown"/>
    </cat-term>
    <cat-term id="log-properties">
      <cat-term id="login-information">
        <control-term id="date-time"/>
        <control-term id="number-of-attempts"/>
      </cat-term>
      <control-term id="host-ip"/>
      <control-term id="event-type"/>
      <control-term id="user-id"/>
      <control-term id="object-id"/>
    </cat-term >
  </con-group>
</compliance-vocabulary>
```

Figure 7: *auditable-event* terms included in an SLA by an example CSP

Each term id must reference a valid entity in one of the security-group trees in the compliance vocabulary ontology. The XML snippet in Fig. 7 identifies particular terms from the *auditable-events* tree that an example CSP has decided comply with its service offering. Specifically, after examining its service offering (not shown) the CSP has decided that the service: has an audit system, provides detailed audit records, can audit login attempts, data modification, and startup and shutdown of the system, and provides log properties that include the host IP, type of event, user ID, and object ID for all events and additionally the date/time and number of attempts for logins.

IV. EVALUATION

A. Compatibility with Existing Security SLAs

Security SLAs come in three forms. The first form, the static freeform plaintext documents, factually state the SDTs and SLOs of a cloud web service [7, 16-18]. Despite the fact that such SLAs are not meant to be machine readable, the graphical nature of our approach can still be used. Rather than including a set of XML elements in the SLA, a CSP using a static plaintext SLA could simply include a graphic of the compliance terms that it satisfies. The other two forms of SLAs are WS-Agreement and WSLA, which are based in XML and constitute the vast majority of research and industry interest.

WSLA defines SDTs directly on top of a service's WSDL [13, 20] and SLOs in the *obligations* section [13, 20]. The `compliance-vocabulary-terms` block can be directly inserted into the WSDL to identify component-specific compliance needs and into an SLO to quantify the expected range of compliance. More specifically following the WSLA schema [1], the `compliance-vocabulary-terms` would become a *SLAParameter* in WSLA. Every SLAParameter has the associated attributes name (`xs:string`), type (`wsla:Type`), and unit (`xs:string`) and relates to a *Metric*. A `wsla:Type` can be any numeral type, string or various others. We propose a SLAParameter as shown in Fig. 8 to be used in WSLA-based SLAs to express the compliance vocabulary on specific WSDL SDTs.

```
<wsla:SLAParameter
  name="comp-vocab-terms-<component name>"
  type="XML-ontology" unit="">
  <wsla:Metric>
    <compliance-vocabulary-terms>
      …(compliance vocabulary content)…
    </compliance-vocabulary-terms>
  </wsla:Metric>
</wsla:SLAParameter>

<wsla:ServiceLevelObjective>
…(other wsla:SLO fields)…
<expression><Predicate wsla:type="True">
  <SLAParameter>comp-vocab-terms</SLAParmeter>
</Predicate></expression>
…(other wsla:SLO fields)…
</wsla:ServiceLevelObjective>
```

Figure 8: WSLA Interface for including Compliance Vocabulary

[1] WSLA-Schema: http://www.xmlns.me/?op=visualize&id=834

The bottom portion of Fig. 8 applies the SLAParameter as a `wsla:ServiceLevelObjective`. This block guarantees that the item is published with the rest of the SLOs in the service offering. This element contains several fields, but the interesting one is the *expression* field. An expression is a `wsla:LogicExpressionType` that is a combination of `wsla:PredicateType` elements, which are either numerical, logical operators, or SLAParameters. To accommodate service-wide compliance declarations we simply rename the SLAParameter to "comp-vocab-terms-system" indicating that the compliance vocabulary applies globally.

```
<wsag:ServiceDescriptionTerm
  wsag:Name="xs:string"
  wsag:ServiceName="xs:string">
  <xs:any>
    <compliance-vocabulary-terms>
      …(compliance vocabulary content)…
    </compliance-vocabulary-terms>
    <xs:any>…service WSDL or XML …</xs:any>
  </xs:any>
</wsag:ServiceDescriptionTerm>

<wsag:ServiceLevelObjective>
  <wsag:CustomServiceLevel>
    <compliance-vocabulary-terms>
      …(compliance vocabulary content)…
    </compliance-vocabulary-terms>
  </wsag:CustomServiceLevel>
</wsag:ServiceLevelObjective>
```

Figure 9: WS-Agreement Interface for Including Compliance Vocabulary

Since WS-Agreement is more flexible and extensible than WSLA, including the compliance vocabulary is straightforward. WS-Agreement abstracts the specification of the SLA away from the WSDL, while retaining its semantics. A `wsag:ServiceDescriptionTerm` contains two attributes, a `wsag:Name` that identifies the SDT and a `wsag:ServiceName` which identifies the service and groups SDTs, allowing multiple service offerings to be described in the same SLA. In addition, a SDT includes an `xs:any` element for extensions. Typically the WSDL with which the SDT is associated is inserted into this field. Given this extensibility, we directly include the compliance vocabulary block in an SDT. WS-Agreement also provides a flexible structure for its SLOs, which are expressed as `wsag:ServiceLevelObjective` elements with `wsag:CustomServiceLevel` elements to allow for the insertion of `xs:any` content. Thus, expressing the vocabulary terms globally simply requires insertion in this block. Fig. 9 shows the structure of our WS-Agreement interface.

We previously constructed SecAgreement [9] to extend WS-Agreement to embed risk-related SDTs and SLOs in a format amenable to risk-based matchmaking. Since, SecAgreement is based on WS-Agreement the interface can be adapted for compliance by changing the *wsag* elements in Fig. 9 to *secag* [9] elements. Each secag element has the same name as the wsag form, but includes additional structure for embedding the risk terms. These terms may be combined with compliance terms to provide matchmaking algorithms security information from both a risk-based perspective and a compliance perspective. The three interfaces shown in this section demonstrate how our compliance vocabulary can be pushed to existing service specifications.

B. Computational Analysis: Parsing and Comparing Terms

For XML-based SLAs, such as WSLA or WS-Agreement, comparisons are typically performed by a service matchmaker [11] which takes in a set of *requested terms*, examines available services, picks the one that most matches the requested terms, and negotiates [13, 19] the terms of service with the service provider. Including the compliance vocabulary in SLAs to augment the assessment process with compliance reasoning naturally entails overhead for parsing and comparing additional SLA terms. We characterize this overhead in terms of two metrics 1) number of term identifiers that must be parsed and 2) number of operations required to compare two SLAs from different providers to determine which SLA offers superior compliance terms.

The first metric is derived from the XML schema specifying the structure of the *compliance-vocabulary-terms* block. Eq. 1 describes the total number of elements and attributes to be parsed for a service using N control group terms. Each group term includes a dominant control d_i, x_i category terms, and y_i control terms.

$$termsParsed = N + \sum_{i=1}^{N} d_i + x_i + y_i \qquad (1)$$

The value of d_i is bounded as $0 \leq d_i \leq 1$ meaning that either the i-th group term includes a dominant control or not. The number of terms to be parsed to assess a service's security offering grows linearly as a service includes additional control group, category, and control term identifiers.

The second metric characterizes the number of operations needed to compare SLAs to assess which has superior compliance terms. We compare SLAs pairwise, term by term, to determine which terms are met, subsumed, or not met by terms from the other SLA. Let *SLA1* have *N1* control group terms, *X1* category terms, and *Y1* control terms. Similarly, SLA2 has *N2*, *X2*, and *Y2*. Assume *SLA1* and *SLA2* both use a compliance vocabulary with R relations between elements. Two *equals* operations are required to compare two group terms, i.e. one for the group term and one for the dominant control term. Category terms can be compared using one *equals* operation. Comparing two control terms can be done with one *ifequal* operation if there are no *subsumes* relations in R. Otherwise the presence of *subsumes* in R means that control terms could potentially *subsume* one another, meaning that $1 + R$ operations are required. Eq. 2 states the number of operations required in the worst case to determine if each service term is *met*, *subsumed-by*, or *not-met* by a term from the other service.

$$ops = 2(\mathrm{N1} * \mathrm{N2}) + (X1 * X2) + (1 + R)(Y1 * Y2) \quad (2)$$

This formulation assumes a worst case of an uninformed search where each type of term in SLA1 is compared against every term in SLA1 of the same type. A realistic algorithm would iterate over all group terms *n1* in *N1* and determine if

N2 also contained the same term. If not, *n1* would be *not-met* by SLA2 and thus, all of the categories and controls in *n1* would automatically be *not-met* by *SLA2*. A similar type of pruning could be applied at the category level preventing additional analysis of control terms, greatly reducing the number of operations to compare two SLAs. Another optimization would be to remove all *n* from *N1* or *N2, x* from *X1* or *X2,* and *y* from *Y1* or *Y2* if they have already been listed as meeting a term in the other SLA. This prevents redundant term checking.

V. CONCLUSION

Security and loss of system control are key issues [1] inhibiting many businesses and government organizations from moving to the cloud. Government agencies have been specifically challenged by the Obama administration to "think cloud first" [1], but worries exist about the security threats that might result from the loss of control over system designs and overall lack of compliance guarantees. In this paper, we propose a compliance vocabulary that creates security SLA terms and associates them with security controls from governance documents. By enhancing their SLA with certification information, cloud service providers can open themselves to entirely new markets of would-be service consumers, particularly in the government sector. From an organizational perspective, having cloud based services with certification guarantees means increased service delivery, efficiency through a better economy of scale, and ultimately bottom-line savings without loss of critical mandated compliance guarantees.

ACKNOWLEDGMENT

This material is based on research sponsored in part by the Air Force Office of Scientific Research (AFOSR), under agreement number FA-9550-09-1-0409.

REFERENCES

[1] A. Minkiewicz, "Cloud Nine, are we there yet?," *Journal of Software Technology,* vol. 14, pp. 4-8, 2011.

[2] NIST, "Special Publication 800-53 Recommended Security Controls for Federal Information Systems," 2009.

[3] "Common criteria (Part 2) for IT Security Evaluation V3.1," 2009.

[4] DoD, "Directive 8500.1: Information Assurance," 2002.

[5] DISA, "Application Security and Development STIG V3 R4," 2011.

[6] CSA, "Cloud Controls Matrix " 2012.

[7] C. Redl, I. Breskovic, I. Brandic, and S. Dustdar, "Automatic SLA Matching and Provider Selection in Grid and Cloud Computing Markets," in *International Conference on Grid Computing*, 2012.

[8] G. Modica, G. Petralia, and O. Tomarchio, "A Business Ontology to Enable Semantic Matchmaking in Open Cloud Markets," in *Eighth International Conference on Semantics, Knowledge and Grids*, 2012.

[9] M. Hale, and R. Gamble, "SecAgreement: Advancing Security Risk Calculations in Cloud Services," *World Congress on Services*, 2012.

[10] M. Hale, and R. Gamble, "Risk Propagation of Security SLAs in the Cloud," in *Proceeding of the workshop on Management and Security technologies for Cloud Computing 2012, IEEE GLOBECOM*, 2012.

[11] K. Bernsmed, M. Jaatun, P. Meland, and A. Undheim, "Security SLAs for Federated Cloud Services," in *Sixth International Conference on Availability, Reliabilty and Security*, 2011.

[12] S. Lindskog, "Modeling and Tuning Security from a Quality of Service Perspective," Ph.D, Dept. of Comp. Sci. and Engineering, Chalmers University Of Technology Goteborg, Sweden, 2005.

[13] A. Keller, and H. Ludwig, "The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services.," *Journal of Network and Systems Management,* vol. 11, 1, pp. 57-81, 2003.

[14] M. Anisetti, C. Ardagna, E. Damiani, and J. Maggesi, "Security Certification-Aware Service Discovery and Selection," in *Proceedings of Fifth IEEE International Conference on Service-Oriented Computing and Applications*, 2012.

[15] N. Bhensook, and T. Senivongse, "An Assessment of Security Requirements Compliance of Cloud Providers," in *4th International Conference on Cloud Computing Technology and Science*, 2012.

[16] K. Belhajjame, S. Embury, and N. Paton, "Verification of Semantic Web Service Annotations Using Ontology-Based Partitioning," in *IEEE Transactions on Services Computing (preprint)*, 2013.

[17] W. Zhu, "Semantic Mediation Bus (TM): An Ontology-Based Runtime Infrastructure for Service Interoperability," in *Proceedings of 16th IEEE International Enterprise Distributed Object Computing Conference Workshops*, 2012.

[18] G. Dobson, and A. Sanchez-Macian, "Towards Unified QoS/SLA Ontologies," in *Proceedings of the IEEE Services Computing Workshops* 2006.

[19] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. (2007). *Web Services Agreement Specification (WS-Agreement)*.

[20] H. Ludwig, A. Keller, A. Dan, and R. King, "A Service Level Agreement Language for Dynamic Electronic Services," in *Proceedings of the 4th IEEE Int'l Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems*, 2002.

[21] M. Klusch, P. Kapahnke, and I. Zinnikus, "Hybrid Adaptive Web Service Selection with SAWSDL-MX and WSDL-Analyzer " in *Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications*, 2009.

[22] T. C. Chieu, M. Singh, C. Tang, M. Viswanathan, and A. Gupta, "Automation System for Validation of Configuration and Security Compliance in Managed Cloud Services," in *Proceedings of Ninth IEEE International Conference on e-Business Engineering*, 2012.

[23] M. Hale, and R.Gamble, "Hierarchically Modeling Security Control Compliance" Report No. SEAT-UTULSA-012013," 2013(under review).

[24] NIST, "Special Publication 800-37, Guide for Applying the Risk Management Framework to Federal Information Systems A Security Life Cycle Approach," 2010.

[25] R. R. Henning, "Security Service Level Agreements: Quantifiable Security for the Enterprise?," in *Proceedings of the New Security Paradigms Workshop*, 1999.

[26] C. Irvine, and T. Levin, "Quality of Security Service," in *Proceedings of the New Security Paradigms Workshop*, 2000.

[27] W3C, "Extensibile Markup Language (XML) 1.0 Fifth Edition.," 2008. Available: http://www.w3.org/TR/REC-xml/.