End-to-End Security Formalization and Alignment for Federated Workflow Management

Matthew Dickinson, Saptarshi Debroy, Prasad Calyam,
Samaikya Valluripally, Yuanxun Zhang, Trupti Joshi, Dong Xu
University of Missouri-Columbia, USA
Email: {*dickinsonmg, debroysa, calyamp*}@missouri.edu, {*svbqb, yzd3b*}@mail.missouri.edu, {*joshitr, xudong*}@missouri.edu

***Abstract*—Traditionally, the allocation and dynamic adaptation of federated cyberinfrastructure resources residing across multiple domains for data-intensive application workflows have been performance or quality of service-centric (i.e., *QSpecs*); often compromising the end-to-end security requirements of scientific workflows. Lack of standardized formalization methods of the workflows' end-to-end security requirements, and diverse/heterogenous domain resource and security policies make inter-conflict characterization between application's security and performance requirements non-trivial, and leads to sub-optimal resource allocation. In this paper, we present a joint security and performance-driven federated resource allocation and adaptation scheme to define and characterize a data-intensive scientific application's security specifications (i.e., *SSpecs*). In order to aid security-driven resource brokering among domains with diverse security postures, we describe an alignment technique inspired by Portunes Algebra to combine domain-specific resource policies (i.e., *RSpecs*) along the application workflow life cycle. We use standardized guidelines that help in compute/storage resource domain/location selection as well as network path selection based on both application *QSpecs* and *SSpecs*. We implement our security formalization and alignment methods as a framework, viz., "OnTimeURB" and apply it on an exemplar Distributed Computing workflow to show the benefits of joint *QSpecs*-*SSpecs*-driven, *RSpecs*-compliant federated workflow management.**

***Index Terms*—Data-intensive application, End-to-end security requirement formalization, Cross-domain policy alignment, Dynamic resource provisioning.**

I. Introduction

Data-intensive science applications (e.g., in areas of high-energy physics, bioinformatics) often require specialized instruments and cyberinfrastructure (CI), i.e., compute/networking/storage resources (e.g., supercomputers, federated data repositories, public clouds) that do not always reside at the data generation sites on researcher campuses [1]. This leads to researchers relying on geographically distributed and federated resources connected with multi-domain physical networks equipped with software-defined networking (SDN) [2] capabilities. A growing trend in multi-domain federations can be seen in exemplar application communities such Large Hadron Collider for physicists [3] and iPlant Collaborative for informaticians [4], and thus the need to facilitate cross-campus data-intensive science collaborations for these researchers is becoming increasingly critical.

As shown in Fig. 1, provisioning of federated CI resources is generally based on applications' performance and quality of service (QoS) requirements i.e., *QSpecs* which can be viewed as a metaphorical gear. This *QSpecs* gear's cogs in most cases conflict with the application's key security and privacy requirements as well as remote instrument security, i.e., the metaphorical *SSpecs* gear cogs. Additionally, strict security and privacy requirements may require extensive firewall policies or selection of resources with scheduling policies or resource capacity (e.g., number of licenses for analytics packages) that may limit the peak performance expected in the application workflows, as shown for the conflict case with the metaphorical *RSpecs* gear cogs. These challenges in capturing distributed application requirements and conflict phenomena mitigation are compounded with the inherent complex nature of distributed application workflows that demand different Service Level Objectives (SLOs) (i.e., varying definitions of *QSpecs*, *SSpecs*) for their life cycle stages across federated resources (with unique *RSpecs*).

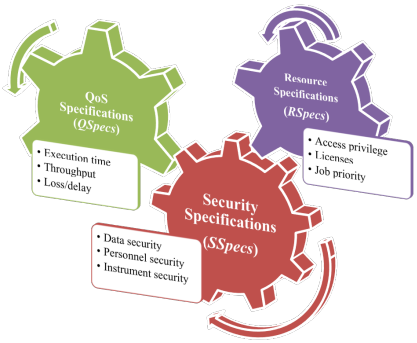


Fig. 1. Inter-conflicts between *SSpecs*, *QSpecs*, and *RSpecs* of a data-intensive application represented as metaphorical gears

Therefore, we argue that there is a need to enhance the ability of such distributed applications to securely use federated resources within predictable performance bounds. This in turn will require a workflow-centric CI resource management paradigm instead of the traditional strictly-*QSpecs*-driven and/or strictly-*SSpecs*-driven approaches. This next-generation federated CI resource management approach first needs to universally define and formalize data-intensive application *QSpecs* and related *SSpecs* from its security requirements along the different stages of its workflow life cycle. This new paradigm also needs to align the diverse domain security postures and resource policies (i.e., *RSpecs*) into homogenous policy statements that are comparable with application *SSpecs*. Hence, solving the end-to-end security alignment problem will lay the foundation of a workflow-centric joint *QSpecs*-*SSpecs*-driven, *RSpecs*-compliant CI resource management that minimizes the inter-conflict among the 3 metaphorical gears while managing application workflows on federated resources. The outcome of such end-to-end security alignment and corresponding resource selection can be performed seamlessly with programmable technologies for computing (e.g., OpenStack [5]), networking (OpenFlow [6] with SDN), storage (iRODS [7]), and security (Shibboleth [8]).

This work was partially supported by the National Science Foundation under award: ACI-1440582. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

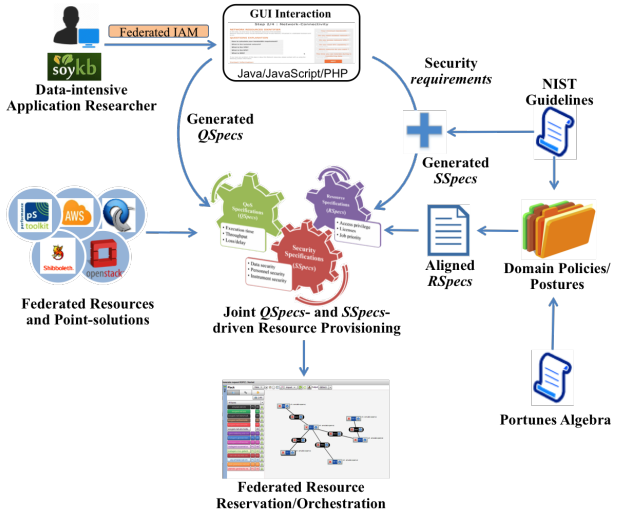


Fig. 2. Steps involved in a joint *QSpecs*-*SSpecs*-driven, *RSpecs*-compliant federated CI resource management scheme

In this paper, we present novel methods for end-to-end security formalization and security-posture alignment (as illustrated in Fig. 2) within a next-generation CI resource provisioning environment that leverages private cloud (local) and remote (public cloud) resources. The novelty of our approach is in our formalization of *SSpecs* of a data-intensive distributed application for different stages of its life cycle utilizing resources across federated domains. For this, we extend the National Institute of Standards and Technology (NIST) guidelines [9] to define specific security categories that are relevant to the 'Data security' and other 'Auxiliary security' requirements for data-intensive applications. The resultant formal *SSpecs* data structure is simple to create, easy to understand and comprehensive enough to account for a wide range of security requirements pertaining to data-intensive application workflows.

Building upon the *SSpecs* formalization, we next propose a novel security alignment method to align heterogenous domain security postures into a homogeneous set of formal statements in order to aid in resource provisioning driven by both application *SSpecs* and *QSpecs*. The novelty of our alignment method is in its use of a predicate logic, viz., "Portunes Algebra" [10] for automated drill-down of domain-specific security and resource policies from security documents into homogenous policies. The resulting portunes statements are compared with the NIST guidelines and specifications [9] to organize domain resources (network/compute/storage) into "High", "Moderate", and "Low" levels based on those combined policies. The output of such a formalization and subsequent alignment process is then fed into a joint *QSpecs*-*SSpecs*-driven federated CI resource management algorithm for network path and compute location selection at each stage of application life cycle in a *RSpecs*-compliant manner. The resulting CI resource provisioning thus ultimately ensures our goal of delivering predictable application performance without sacrificing the security requirements.

We implement our proposed security formalization and alignment schemes in a *unified resource broker* framework viz., "OnTimeURB" that performs joint *QSpecs*-*SSpecs*-driven, *RSpecs*-compliant federated CI resource management for an exemplar application, viz. SoyKB [11]. Our experiments on a real-world testbed demonstrate how OnTimeURB helps in aligning the domain security postures and pertinent resource domain/location and network path selection amongst three candidate (federation) domains along the SoyKB workflow lifecycle. The domains refer to the resources at the following institutions: University of Missouri (MU), University of Texas at Austin (UT), and University of Southern California (USC). By comparing results of our joint *QSpecs*-*SSpecs*-driven approach against only *QSpecs*-driven and only *SSpecs*-driven approaches for federated CI resource management, we show efficient resource allocation that satisfies SoyKB application's *SSpecs*, and *QSpecs* without overriding intermediate-domain *RSpecs* through pertient policy alignment.

The remainder of the paper is organized as follows: Section II presents related work. Section III describes the process to define and generate *SSpecs*. Section IV discusses the formal method of aligning domain policies and security postures. Section V discusses the OnTimeURB implementation and results of SoyKB application's CI resource provisioning. Section VI concludes the paper and suggests future work.

II. Related Work

In terms of scheduling data-intensive workflows using traditional QoS-driven resource provisioning, different approaches are proposed in [12], [13]. Authors in [12] show how QoS requirements are determined by the application traffic type, such as multimedia or file transfer. In our earlier work on ADON (application-driven overlay network-as-a-service) [13], we used SDN within hybrid cloud computing architectures for on-demand and concurrent application handling for accelerated performance of data-intensive application workflows. In all of these works, security among domains is not explicitly stated within the SLOs, when choosing amongst a set of resource domains or interconnecting paths. Other works such as [14], [15] extend QoS-driven approaches to include security requirements. However, security alignment as a requirement for resource provisioning has not been addressed in an end-to-end manner. They opt to instantiate point solutions such as SDN [16] and Network Function Virtualization (NFV) [17] along with methods to test the solutions' impact on federated cloud security. In contrast, our work is unique because we address the end-to-end multi-domain security design by defining and aligning *SSpecs* of a data-intensive application along an application workflow life cycle.

With regards to aligning diverse domain-security postures using a standardized technique, the current literature lacks foundational methods to define and formalize *SSpecs*. Although the National Institute of Standards and Technology (NIST) [18] has provided guidelines for Security and Privacy Controls for Federal Information Systems and Organizations [9], the security postures of institutions such as e.g., University of Missouri (MU) [19], University of Texas at Austin (UT) [20], and University of Southern California (USC) [21] can, and are heterogenous, disproportionate ranging from 5 to 150 pages of documents with limited, and in some cases - no alignment with the NIST guidelines.

In related works for security formalization and alignment, domain security assessment has generally been performed by quantification [22], or risk analysis as suggested by NIST. The current literature lacks any definitive solutions to directly compare different domains' security requirements. In [10], a formal approach is proposed to align security policies within an organization between physical, digital, and social domains, where the security profiles are updated as wrappers along the organization hierarchy. The alignment of domain security

policies is performed informally by authors in [23] to assess organizational goals. Work in [24] discusses policies in terms of sequences of actions; their framework allows refinement of both systems and policies based on UML specifications. However, they do not explicitly address end-to-end security policies spanning multiple domains that affect application performance as we do in our novel scheme that uses portunes algebra and NIST guidelines in order to convert diverse or even mutually orthogonal domain security postures into a more homogenous set of formal statements.

Existing works pertaining to security and dependability for federated CI resources in data-intensive research communities mostly deal with security measures and point solutions to counter confidentiality, availability, and integrity threats rather than designing an end-to-end security design that helps in dynamic provisioning and adaptation using such measures. Exemplar solutions to Big Data transfer in a federated environment include Globus that provides the ability to use point solutions such as InCommon [25], OpenID [26] and X.509 [27] to access resources [28]. The Large Synoptic Survey Telescope (LSST) community on the other hand provides detailed guidelines for multi-domain cybersecurity compliance with a list of threat mitigating capabilities at involved domains [29]. These communities can benefit from our formal approach of resource provisioning based on multi-domain security requirements, and augment their current approach of manual co-ordination of policies to find security alignment in workflow management across federated resources.

III. Security Specifications Formalization

In this section we will first present a distributed computing use case that can motivate the joint *QSpecs*-*SSpecs*-driven, *RSpecs*-compliant federated workflow management. Next we will propose our NIST inspired *SSpecs* formalization scheme.

A. Distributed Computing Use Case

Knowledge Base (KB) approaches, in areas such as transactional genomics and breeding, allow users and researchers to have a single point of information source. An exemplar in this area is the SoyKB, which is a comprehensive web resource developed at MU for soybean translational genomics and breeding. The SoyKB application handles the management and integration of soybean genomics and multi-omics data along with gene function annotations, biological pathway and trait information. The SoyKB has been featured as a model use case for distributed computing in the systems biology area within the Open Science Grid community [30].

The SoyKB requires large data sets from data archives being transferred to MU for pre-processing and subsequently being transferred to either local private cloud at MU or remote public cloud HPC sites, viz., ISI (Information Sciences Institute) at USC, and TACC (Texas Advanced Computing Center) at UT for analysis as shown in Fig. 3. Following computation completion, the resultant processed data will be transferred to MU and also to the iPlant Collaborative [4] storage. For different stages of the cross-domain transfer, the data can take either dedicated high-speed Layer 2 connectivity or regular Layer 3 Internet path depending on the QoS requirements and availability of network resources.

As the first step towards joint *QSpecs*-*SSpecs*-driven resource management for SoyKB application, we define and formalize a data-intensive application's *SSpecs*. Although existing environments, such as the Global Environment for Network Innovations (GENI) [31], make use of *RSpecs* and *QSpecs* to request available resources, definition and characterization

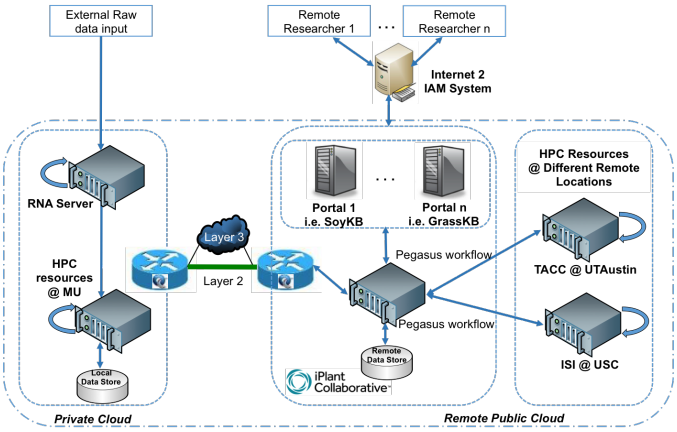


Fig. 3. SoyKB user workflows with federated CI resource requirements

of *SSpecs* from application's security requirements is not addressed directly in the orchestration of the workflow life cycle. In the following, we list the key challenges in formalizing of security requirements to *SSpecs*.

B. Challenges in formalizing SSpecs

- As the size, nature, and structure of data for a data-intensive application change along the involved workflow's life cycle, so does change the security requirements associated with the data. Representing such change in a standardized manner becomes even more complicated when the different stages of the workflow life cycle involves multiple domains.
- Not all these participating domains will treat data classification with a unified standard, because domains carry their own classification of data in their security postures. And not all domains will be willing to accept an overriding security classification standard on top of own their domain posture.
- A data-intensive application's data security classification is generally defined according to the originating domain's security posture. However, due to the diversity of different domain postures, in most cases such classifications become reduced to a conservative setup, or the security compliance from both data-side and domain-side becomes complicated.
- Although security requirements of a data-intensive application are predominantly data-driven, they directly correspond to CI (network/compute/storage) resources where the data resides.
- For any given stage in a workflow's life cycle, additional security requirements may be added on top of any initial security requirements, such as scientific instruments generating the data, software analysis tools, and personnel associated and their privileges. The formal security requirements definition needs to have such provisions.

Keeping in mind such challenges and the basic need for a formalization of security requirements, we define *SSpecs* of a data-intensive application as:

Definition 1 (SSpecs): A formal data structure to describe the security requirements of the application workflow for any state during the workflow life cycle in terms of data, CI resources handling the data, instruments, software tools, and personnel involved.

Example: Figure 4 shows a pictorial representation of our proposed *SSpecs* formalization for a typical data-intensive application. The data structure is divided into life cycle stages with each life cycle stage further divided into Data and Auxiliary security requirements. The Data requirements are again

divided in to specific CI resource requirements in terms of network, compute and storage resources in order to facilitate joint *QSpecs*-*SSpecs*-driven, *RSpecs*-compliant resource brokering. Both Data and Auxiliary requirements are expressed using security requirement categories that follow NIST guidelines, the description of the requirement for that category and finally the level (High/Medium/Low) of that requirement. Details about the *SSpecs* data structure are discussed next.

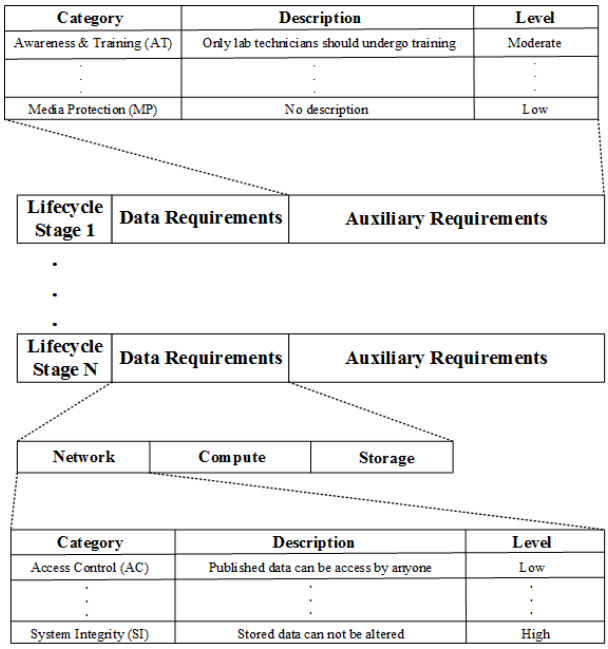


Fig. 4. A data-intensive application's *SSpecs* format in terms of Data and Auxiliary security requirements

C. Application workflow life cycle

Every data-intensive application workflow can be viewed as collection of life cycle stages, such as acquisition, preprocessing, processing, analysis, and collaboration. While going through these stages, data will be frequently changing in form and size depending upon the scientific protocol in the research experiment. In most cases, such life cycle stage division is also associated with diverse CI, software, or hardware resource requirements that essentially become part of *QSpecs*. This in turn helps in compute/storage resource selection and network path selection between domains for distributed resource provisioning. As our objective in defining and characterizing *SSpecs* is to include an application's security requirements as a factor for such provisioning, it is all but natural that the *SSpecs* of that application should also be defined in terms of life cycle stages.

Defining *SSpecs* on the basis of the application workflow life cycle serves two purposes that are directly linked to the challenges described in Section III-B. Firstly, a life cycle based approach obviates the need for having the application user(s) original site's security posture as an overarching influence on the application *SSpecs*. This in turn removes any chance of potential conflict between the data classification rules used to define the *SSpecs* and any other domain's (along the life cycle) security posture. Secondly, such a life cycle based characterization makes the addition of *SSpecs* into any resource provisioning algorithm that uses *QSpecs* as the differentiating factor in choosing ideal domains and paths.

TABLE I
NIST SP 800E GUIDELINES COMPATIBLE DATA AND AUXILIARY SECURITY CATEGORIES AND FAMILY NAMES

Data Requirements		Auxiliary requirements	
ID	**Family**	**ID**	**Family**
AC	Access Control	AT	Awareness and Training
AU	Audit and Accountability	CM	Configuration Management
CA	Security Assessment and Authorization	CP	Contingency Planning
IA	Identification and Authentication	IR	Incident Response
SA	System and Services Acquisition	MA	Maintenance
SC	System and Communication Protection	MP	Media Protection
SI	System and Information Integrity	PE	Physical and Environmental Protection
		PL	Planning
		PM	Program Management
		PS	Personnel Security
		RA	Risk Assessment

D. Data and Auxiliary Security Requirements

For any given stage of the workflow life cycle, application *SSpecs* is divided into Data and Auxiliary security requirements. The Data requirements are based upon the factors (categories) that directly relate to the data security involving the CI resources. We use NIST [32] guidelines for the comprehensive list of 18 security and privacy control categories for federal information systems. We further divide them into Data and Auxiliary categories in accordance with our *SSpecs* data structure. Table I shows the list of 18 categories divided into Data and Auxiliary security requirements and their corresponding NIST descriptions.

The category description in the *SSpecs* data structure (as shown in Fig. 4) differs from that of the description in Table I. Descriptions in the data structure represents the security requirements that are specified by the application for that particular category. Depending upon the description, the corresponding category at a particular life cycle stage is given either a High/Moderate/Low level ranking based on the NIST guidelines. As an example, in Fig. 4, data accessed by anyone in a particular domain/stage is deemed as Low level security requirement by NIST guidelines. In our opinion, Table I represents the comprehensive list of security related categories/issues that are relevant for a data-intensive application. In general, applications do not specify any particular requirements for most categories at different life cycle stages. Further, if security requirements are not mentioned for any category for a particular life cycle stage or even for the entire workflow, the security requirement level is as Default.

The Auxiliary requirements are the security specifications imposed on a workflow on top of Data requirements involving CI resources. These mainly deal with security requirements that can not be categorized into network/compute/storage resources, rather they involve scientific instruments, analysis tools and expensive software, and users/personnel related social requirements, or a combination of all these at different stages of the workflow life cycle. We associate 11 out of 18 NIST security control categories with Auxiliary security requirements of a data-intensive application. The method of assigning levels for such Auxiliary requirement categories is the same as in the case of the Data requirements discussed in the previous section and shown in Fig. 4. For example, a scientific instrument at life cycle stage 1 of a data-intensive application might be a very sophisticated, exclusive and secured equipment that requires expert handling by technical staff with

sufficient training. However, the security requirements considers faculty and graduate students sufficiently trained and only requires technician staff in the lab to be adequately trained. According to the NIST security descriptions from [32], such requirements fall under category "Awareness and training" (AT) and can be ranked in the Moderate level as shown in Fig. 4.

Similarly, user and personnel related security requirements are also included into the Auxiliary requirements of *SSpecs*. The reputation/trust of inter- or intra-domain users associated with the workflow is an important factor in deciding whether the security (in terms of confidentiality, integrity and availability) of the overall workflow is being compromised or not. Thus, for each given user associated with any stage of the workflow, the history of the user using that application, or similar applications, or even resources associated with that application can be taken into account and specified as a part of security category description. It can subsequently be scored with a level according to the NIST guidelines. We identify, categories such as "Personnel Security" (PS) and partly "Risk Assessment" (RA) as the areas where such user related requirements can be specified. It is to be noted that such user related security requirements are not same as "Access Control" (AC) category, which mostly deals with access to data at different life cycle stages. In PS and RA categories, we rather associate users' access to elements other than data (e.g., instruments, hardwares, tools) and exception conditions on top of specified rules for access to such elements based on a user's history.

IV. Aligning Domain Security Policies

As stated earlier, one of the major barriers in joint security and performance-driven resource management is the fact that in most cases the domain security postures involving *RSpecs* are diverse and cannot be compared with application security requirements for compliance. This is especially true for many public and private universities in the United States who separate their security classification levels for different resources from anywhere between three to six levels that are difficult to align. For example, we specifically study the data classification policies of a selection of universities related to the SoyKB application: on one hand, universities such as, UT [20], USC [21] classify the data into three different categories. Whereas, MU [19] on the other hand, classify data into four different categories. Moreover, universities, such as Harvard [33] uses five categories. The resource security policies for each of these classifications vary between institutions, making security specification compliance while reserving federated resources a burden for both data-intensive application users and resource providers.

TABLE II
Comparison of Security Policy Levels of Different Universities

MU	UT	USC
Highly Restricted	Confidential	Highly Sensitive
Restricted	Controlled	Sensitive
Sensitive	Published	Private
Public	-	-

More specifically, inconsistencies between the classification levels of institutions can cause confusion in how the domain resources should be classified. UT for instance, does not classify any of its information as "Public", thus making all the resources apparently secured. Whereas the definition of "Sensitive" at MU is different than the same at USC. In addition to resource classification, the length and complexity

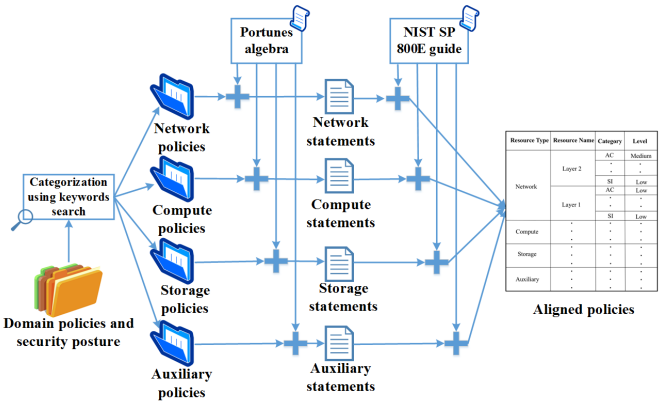


Fig. 5. Logical steps to align diverse domain security posture into standardized NIST compatible security policies

of the security posture differ greatly with one university having a 7 page generic posture with high-level policies and 7 different categories, whereas another university having a very detailed posture with intricate policies of over 160 pages and over 30 categories.

Thus, in order to align such diverse/heterogeneous security postures into homogenous policy statements that can make the domain *RSpecs* comparable to the application's *SSpecs* for resource brokering, we propose a 3-step security alignment scheme. Fig. 5 shows our proposed 3-steps scheme, where we first categorize the policies based on the type of resources (network/compute/storage), then drill down security policies pertaining to each of the resource types into homogenous formal policy statements using "Portunes Algebra" [10], and then assign security levels to each such resources by applying NIST SP 800E guidelines. The outcome of such a process is the homogenous categorization of different domain *RSpecs* that is comparable with a data-intensive application's *SSpecs*.

A. Portunes Algebra

We use portunes algebra [10] to perform the first step shown in Fig. 5 in order to convert diverse domain security postures into a more homogenous set of formal statements. Although, the authors in [10] proposed portunes algebra to align physical, digital, and social security polices within a domain to remove inconsistencies, we are the first to use the portunes to boil-down both generic postures (i.e., high level policies, e.g., MU) and fine-grained postures (i.e., low level policies, e.g., UT) into formal homogenous statements. Below, we discuss using examples, the relevant concepts in applying portunes algebra to represent simple and complex policies.

Definition 2 (Policy): A policy is a theory Θ in first-order predicate logic, with behaviors $T \in \mathcal{T}$, and $P(\_)$, a distinguished prefix-closed predicate over behaviors.

The formula $P(T)$ means that behavior T is permitted or possible; $\sim P(T)$ means that a behavior T is forbidden or impossible. If neither $P(T)$ nor $\sim P(T)$ can be derived from a policy, then the permissibility of T is undecided. For example, the policy $\{\sim P(Action1, Action2)\}$ would forbid all behaviors beginning with *Action1* followed by *Action2*.

Definition 3 (Simple policy): A simple policy is a set of sentences Θ of the form $P(T)$ or $\sim P(T)$, with T being a behavior.

A simple policy can be understood as assigning to each behavior a value among: don't care, permitted, forbidden, or contradiction. Many policies allow certain behavior, however

Resource Type	Resource Name	Category	Level
Network	Layer 2	AC	Medium
		. .	. .
		SI	Low
	Layer 1	AC	Low
		. .	. .
		SI	Low
Compute	. . .	. . .	. . .
Storage	. . .	. . .	. . .
Auxiliary	. . .	. . .	. . .

Fig. 6. Representation of aligned security posture of any domain

they require that a certain result can be achieved in relation to an institution goal. Often, it is not of essential importance how this result is achieved. For example, there should be at least one possible way to change the configuration of an e-mail server. This means that security policies can forbid all but one of the concerned behaviors, as long as this one behavior remains possible. We can thus have a situation where out of a set of behaviors, at least one should be possible.

Example 3.1: Remote access of public data is permitted by using Secure Shell (SSH).

The corresponding portunes representation of the policy will be $nc(publicData, ssh)$ where nc denotes generic *netcopy* action for remote access.

Definition 4 (Extended policy): An extended policy Θ is a set of sentences of the form $\phi_1 \vee \phi_2 \vee \cdots \wedge \phi_n$, where each ϕ_i is of the form $P(T)$ or $\sim P(T)$, with T being a behavior, and $\vee$ and $\wedge$ denote conjunction and disjunction of policies respectively.

The extended policies are only "extended" with respect to simple policies, not with respect to the general notion of policy. Extended policies are a subset of high-level policies to be found in generic postures, and simple policies are a subset of extended policies to be found in detailed postures.

Example 4.1: The server data should never leave a secure server.

The corresponding portunes representation of the policy will be $nm(serverData, secureServer) \wedge nc(serverData, secureServer)$ assuming *netmove* and *netcopy* being the only two actions permitted on secure server data. Portunes specifies another basic action *neteval* and expresses any simple or extended policies using these three fundamental actions.

B. NIST conversion

As shown in Fig. 5, once the security postures are homogenized into portunes statements, we apply NIST guidelines from NIST SP 800E document [9] to determine the security level (High/Medium/Low/Default) of each type of resource and for each of the 18 categories (discussed in Section III) based on the detailed descriptions provided in [9]. This way, the output of our proposed 3-step process is aligned *RSpecs* of a domain that looks Fig. 6, and can be easily compared with application *SSpecs* for joint *QSpecs*-*SSpecs*-driven, *RSpecs*-compliant resource provisioning.

V. Resource Provisioning Implementation and Case Study Results

We implement our proposed security formalization and alignment scheme, and the ensuing workflow management by developing algorithms and software elements as part of a *unified resource broker* framework, viz., OnTimeURB shown in Fig. 7. OnTimeURB intelligently use distributed CI resources, and point solutions to dynamically manage and adapt federated CI resources in an agile, timely and federation policy-compliant manner by implementing *SSpecs* formalization, *RSpecs* alignment, and 3-way gear optimization algorithm shown in Algorithm 1 and explained in Section V-D. OnTimeURB is built using RESTful APIs [34] that are modular, and interoperable with common data-intensive application deployments, and it enables OnTimeURB to be integrated with popular federated authentication and authorization frameworks (e.g., Shibboleth-based). We used the SoyKB application (discussed in Section III-A) as a use case for OnTimeURB evaluation and next will show SoyKB security formalization, policy alignment and federated resource allocation results using OnTimeURB.

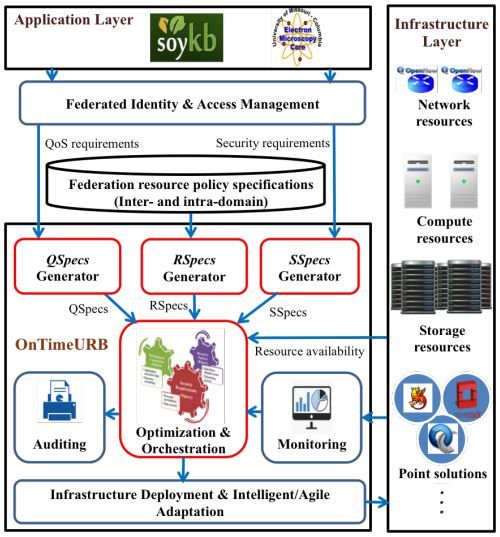


Fig. 7. OnTimeURB functional and logical architecture

A. SoyKB QSpecs

Fig. 8 shows SoyKB *QSpecs* generation using OnTimeURB that includes 1Gbps data throughput between MU and remote servers (i.e., within iPlant) with little or no packet loss in order to justify wide-area network path selection to move data to remote compute resources, instead of local computation. *QSpecs* also include provisioning of adequate HPC computing resources (12 cores, 12 GHz, 32 GB RAM) either at local or remote site for faster processing of raw genomic data of concurrent KB flows to meet strict QoS requirements in terms of execution times, depending on the characteristics of the different KB processing pipelines. Finally, the iPlant storage requires 1 TB persistent hard drive for processed data for remote collaboration. The data transfer from remote HPC sites to iPlant needs to have 200 Mbps throughput.

B. SoyKB SSpecs

Fig. 9 shows SoyKB *SSpecs* using OnTimeURB for all the life cycle stages from application security requirements. The requirements are collected through a careful and relevant

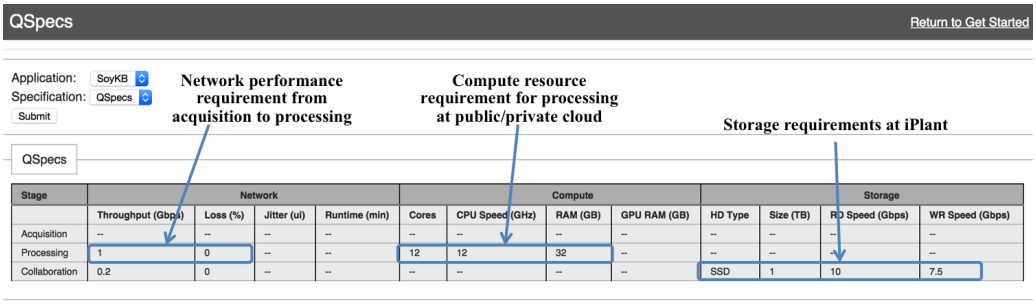


Fig. 8. SoyKB application *QSpecs* for different stages of application life cycle using OnTimeURB framework

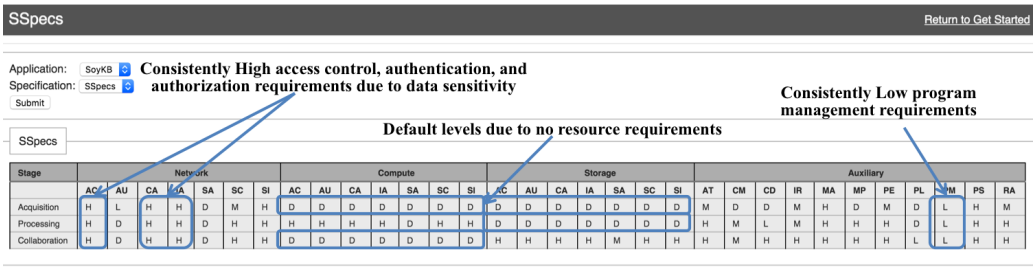


Fig. 9. SoyKB application *SSpecs* for different stages of application life cycle using OnTimeURB framework

questionnaire asking for network, compute, storage resource and auxiliary requirements for different stages of the workflow as part of our OnTimeURB framework. At the Acquisition stage, the security requirements mostly pertain to network resource, with compute and storage security requirements set to Default. Compute and storage security requirements are more elaborate for processing and collaboration stages, respectively. Whereas the auxiliary requirements are more all encompassing where requirements, such as awareness and training, personnel security, and risk assessment require consistently substantial protection through the application life cycle.

C. Converting TACC RSpecs into portunes statements

Next we demonstrate some example of TACC *RSpecs*, i.e., UT security policies (UT-IRUSP) statements being converted into portunes statements and categorized into network, compute, storage resources or as auxiliary policies using our OnTimeURB framework. Although the UT-IRUSP statement document is more than 150 pages long, below we will show policy statement alignment that is relevant to the SoyKB application.

UT-IRUSP 4.2.3 mandates *monitoring for identifying and disabling of unauthorized (i.e., rogue) wireless access points*. The corresponding portunes statement is:

$$\langle ne(\sim (nm(WAP, 1_l, Server) \ \vee \ nm(U\_WAP, 1_l, Server)))\rangle l_t$$

where *WAP* is Wireless Access Point and *U_WAP* is unauthorized WAP. The location predicate 1_l denotes the server can be located at any generic location, and the entire statement is true of all such generic locations signified by the expression l_t. This statement belongs specifically to network resource.

UT-IRUSP 4.3 recommends that *UT Austin must approve all network hardware connected to UT network system resource in order to ensure integrity*, the portunes statement is as follows:

$$\langle \sim nm(System, 1_l, UT\_SytemResources) \ \vee \ nm(auth\_sys, 1_p, UT\_SystemResources)\rangle tt$$

Similar to the location predicate, 1_p denotes the process predicate for any generic process, and tt signifies satisfaction of all kinds of predicate logic. This statement can also be categorized exclusively to network resources.

For compute resource specific policies, UT-IRUSP 17.1.3 states *vulnerability assessments are performed annually, at minimum, to identify software and configuration weaknesses within information systems*, yielding the portunes statement

$$\langle (nc(results, file, 1_l) \ \wedge \ ne(assess, software, InformationSystems))\rangle tt$$

A example of a specific policy UT-IRUSP 4.2.1 requires to *establish and communicate to Users the roles and conditions under which Remote or wireless Access to Information Resources containing Confidential Data is permitted*, leading to the portunes statement:

$$\langle \sim ne(person, 1_p, PrimaryServer) \ \vee \ nm(person, 1_p, server)\rangle tt$$

The above examples present only a small subset of portunes statement generated from TACC and ISI domain/resource security policies. For resource provisioning of SoyKB application using OnTimeURB, we converted the all the policies belonging to UT and USC that are relevant to SoyKB resource provisioning.

D. Resource provisioning

OnTimeURB 3-way gear optimization algorithm shown in Algorithm 1 takes an application a's *SSpecs* SS_a, *QSpecs* QS_a and different domain's resource availability RA_d, and aligned *RSpecs*/domain policies DP_d as inputs where $d \in D$, and D is the set of domains. The outcome of the algorithm is resource allocation $A_a = \{A_a^N, A_a^C, A_a^S\}$ in terms of network, compute, and storage resources that satisfy both the *SSpecs* and *QSpecs* of the application a.

Algorithm 1 follows a heuristic where the available resources at each domain is checked for satisfiability of the application QoS requirements. If satisfied, then each of the

network, compute, storage, and auxiliary security requirements of the application *SSpecs* is compared with the aligned *RSpecs* of the domain. If each category of all the resource types in the domain *RSpecs* have equal or higher security level than the corresponding application *SSpecs*, then that domain is considered to satisfy the application *SSpecs*. This was when both the *QSpecs* and *SSpecs* requirements satisfiability for all the domains are evaluated, only the domain offering maximum resources from the set of domains satisfying both *QSpecs* and *SSpecs* is chosen to be the destination domain. Next, we will use this algorithm to provision resources from UT, USC and MU for different life cycle stages of the SoyKB application.

Algorithm 1: 3-way gear Optimization Algorithm

Data: *SSpecs* $SS_a = \{S_a^N, S_a^C, S_a^S, S_a^A\}$ of application a
Data: *QSpecs* $QS_a = \{Q_a^N, Q_a^C, Q_a^S\}$ of application a
Data: Resource availability $RA_d = \{R_d^N, R_a^C, R_a^S\}$ of each candidate domain $d \in D$, with D domain set
Data: Aligned *RSpecs* or domain policies $DP_d = \{P_d^N, P_a^C, P_a^S, P_d^A\}$ of each candidate domain d
Result: Resource allocation A_a of application a
for *all candidate domains* $d \in D$ **do**
  if $\left(R_d^N \geq \textit{resourceEquv}(Q_a^N)\ \&\&\ R_d^C \geq \textit{resourceEquv}(Q_a^C)\ \&\&\ R_d^S \geq \textit{resourceEquv}(Q_a^S)\right)$ **then**
    if $\left(\textit{levelof}(S_a^N) \geq \textit{levelof}(P_d^N)\ \&\&\ \textit{levelof}(S_a^C) \geq \textit{levelof}(P_d^C)\ \&\&\ \textit{levelof}(S_a^S) \geq \textit{levelof}(P_d^S)\ \&\&\ \textit{levelof}(S_a^A) \geq \textit{levelof}(P_d^A)\right)$ **then**
      $A_d^N = \textit{resourceEquv}(Q_a^N)$;
      $A_d^C = \textit{resourceEquv}(Q_a^C)$;
      $A_d^S = \textit{resourceEquv}(Q_a^S)$;
    else
      $A_d^N = 0;\ A_d^C = 0;\ A_d^S = 0;$
  else
    $A_d^N = 0;\ A_d^C = 0;\ A_d^S = 0;$
  $A_d = \{A_d^N, A_d^C, A_d^S\};$
end
Return $A_a = \textit{maximize}(A_d)$;

E. OnTimeURB optimization outcome and results

We use our proposed OnTimeURB framework to provision resources for the SoyKB application, specifically for compute location selection for Processing stage and ensuing network path selection to the final storage facility at iPlant. This is because the SoyKB application mandates data to be collected at MU at the Acquisition stage and final public access from the iPlant data store.

The three choices for compute location selection were MU, TACC, and ISI HPC clusters. The outcome of OnTimeURB joint *QSpecs-SSpecs*-driven resource provisioning scheme is shown in Fig. 10 where TACC public cloud resources are chosen over MU and ISI for data processing. Fig. 10 also shows the outcome of our previous ADON [13] based resource allocation that uses a traditional *QSpecs*-driven allocation, and only *SSpecs*-driven allocation. The figure shows that in case of ADON, local MU private cloud resources are chosen over public cloud, and for only *SSpecs*-driven allocation remote public cloud at ISI is chosen over TACC. Next we discuss the reason behind such observations.

Figs. 11(a) and 11(b) show the reason behind the choice of MU private cloud over public cloud with ADON based *QSpecs*-driven allocation. Fig. 11(a) shows that due to local

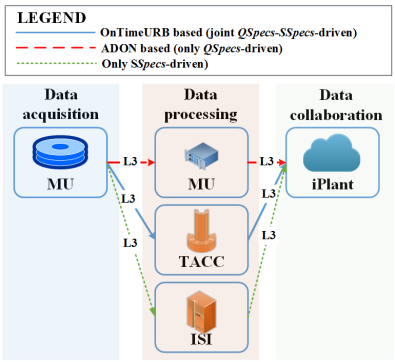
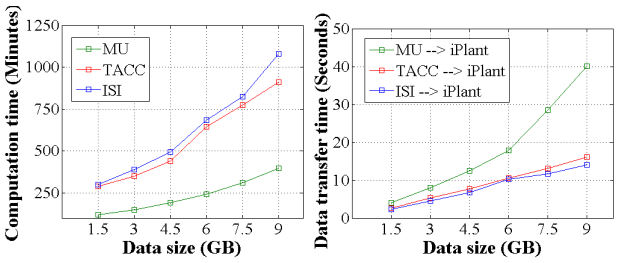


Fig. 10. Comparing network path and domain selection outcomes for different life cycle stages for SoyKB workflow between only *QSpecs*-driven, only *SSpecs*-driven and joint *QSpecs-SSpecs*-driven allocation

resource availability and high priority task scheduling at local site, the compute time for different sizes of soybean genomic data at MU is much less than that at TACC and ISI which closely comparable to each other. Although Fig. 11(b) shows longer transfer time from MU to iPlant due to lower available bandwidth, the scale of deterioration in transfer time (in seconds) is no match to the scale of improvement in compute time (in minutes). Therefore the overall end-to-end execution time remains better at MU and consequently the *QSpecs*-driven resource allocation chooses MU private cloud over public cloud at TACC or ISI.



(a) Compute time comparison (b) Transfer time comparison

Fig. 11. SoyKB application performance comparison at different processing locations for different sizes for genome data sizes

However, Fig. 12 shows a different story as far as satisfying the SoyKB *SSpecs* is concerned. The figure shows the three domains' aligned NIST-inspired *RSpecs* for network, compute and auxiliary security policies (no storage, as it is irrelevant for Processing stage) and their compliance with SoyKB *SSpecs* for Processing stage (as shown in Fig. 9). The levels representing lower and higher (or equal) than the SoyKB application's *SSpecs* levels for different categories are shown using annotations in Fig. 12. We see that MU, due to its very high level policies and small number of security APIs, exhibit less than required security standards. Whereas, public clouds, such as TACC and ISI that are designed for remote HPC collaboration, supports all the SoyKB security requirements. Therefore OnTimeURB resource provisioning algorithm (Algorithm 1) chooses TACC as the final Processing site due to its compliance with SoyKB *SSpecs* and marginally better performance than ISI in terms of end-to-end execution (compute and transfer time) time. However, only *SSpecs*-driven allocation chooses ISI over TACC as ISI *RSpecs* are more stringent in few categories than TACC (annotated in Fig. 12) making ISI marginally better option than TACC in terms of security.

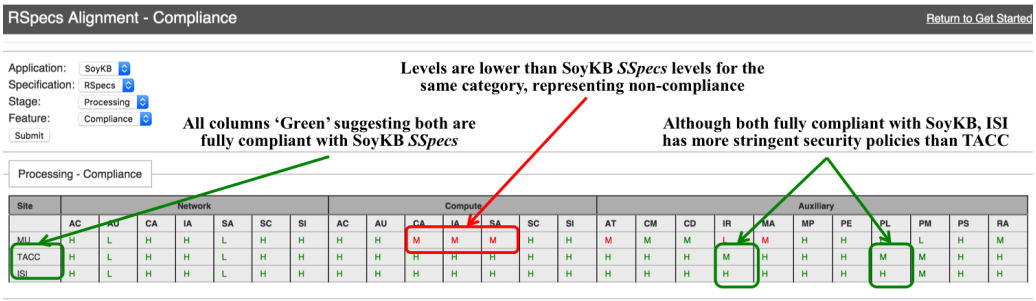


Fig. 12. *RSpecs* alignment outcome of SoyKB application candidate sites/domains and their annotated compliance and non-compliance with SoyKB *SSpecs*

VI. Conclusions

In this paper, we motivated the need to formalize *SSpecs* of distributed applications for more efficient workflow management across federated resources. We showed how a process of breaking down the security requirements across workflow life cycle stages and applying NIST based categorization can facilitate formalization of application *SSpecs*. Our unique use of portunes algebra to align diverse domain postures resulted in homogenizing domain *RSpecs* that is easily comparable with a data-intensive application's *SSpecs* to achieve joint *QSpecs-SSpecs*-driven, *RSpecs*-compliant resource provisioning. Our implementation of OnTimeURB and case study evaluation with the SoyKB application demonstrated the benefits of our proposed approach in ensuring both satisfaction of performance and security requirements by limiting friction across domains, without overriding any domain policies to gain performance advantages. Our work advances the current knowledge on how to intelligently perform resource allocations among public and private cloud locations to reduce turnaround times in a secured and policy-compliant manner. The data-intensive application communities can benefit from our novel approach of resource provisioning, and augment their current techniques of manual co-ordination of policies.

Our future work is to build upon the foundations presented in this paper, and comprehensively model the gear frictions for other distributed computing based data-intensive applications. We plan to create 3-way gear optimization algorithm extensions and evaluate them for various distributed applications and communities, such as, remote instrumentation for physicists, and in the context of other cloud resource providers (e.g., Amazon Web Services, Microsoft Azure, NSF CloudLab).

References

[1] R. Mount and D. Skinner, "Scientific collaborations for extreme-scale science workshop report," US Department of energy, Tech. Rep.

[2] I. Monga, E. Pouyoul, and C. Guok, "Software-defined networking for Big-Data science - architectural models from campus to the WAN," in *High Performance Computing, Networking, Storage and Analysis*, 2012.

[3] "Large Hadron Collider," `http://home.cern/topics/large-hadron-collider`.

[4] S. Goff, *et al., "The iPlant Collaborative: Cyberinfrastructure for Plant Biology"*. Frontiers in Plant Science, 2011.

[5] "OpenStack," `http://www.openstack.com`.

[6] N. McKeown, T. Anderson, and H. Balakrishnan, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, p. 2, 2008.

[7] "Integrated Rule-Oriented Data System," `http://irods.org/`.

[8] "Shibboleth," `https://shibboleth.net/`.

[9] "Security and privacy controls for federal information systems and organizations," NIST SP800-30 Technical Report, Tech. Rep., 2013.

[10] W. Pieters, T. Dimkov, and D. Pavlovic, "Security policy alignment: A formal approach," *Systems Journal, IEEE*, vol. 7, no. 2, June 2013.

[11] T. Joshi, M. Fitzpatrick, S. Chen, Y. Liu, H. Zhang, R. Endacott, E. Gaudiello, G. Stacey, and H. Nguyen, *D.* u, "Soybean Knowledge Base (SoyKB): A web resource for integration of soybean translational genomics and molecular breeding", *Nucl. Acids Res*, 2014.

[12] W. Kim, P. Sharma, J. Lee, S. Banerjee, J. Tourrilhes, S.-J. Lee, and P. Yalagandula, "Automated and scalable qos control for network convergence," in *Proceedings of the Internet Network Management Conference on Research on Enterprise Networking*, 2010.

[13] R. B. Antequera, P. Calyam, S. Debroy, L. Cui, S. Seetharam, M. Dickinson, T. Joshi, D. Xu, and T. Beyene, "ADON: Application-driven overlay network-as-a-service for data-intensive science," in *Cloud Computing, IEEE Transactions on*, 2016.

[14] C. Irvine and T. Levin, "Quality of security service," in *Proceedings of the Workshop on New Security Paradigms*, 2000.

[15] S. Lindskog, "Modeling and tuning security from a quality of service perspective," Ph.D. dissertation, Chalmers Univ. of Tech., 2005.

[16] A. Zaalouk, R. Khondoker, R. Marx, and K. Bayarou, "Orchsec: An orchestrator-based architecture for enhancing network-security using network monitoring and sdn control functions," in *Network Operations and Management Symposium (NOMS), 2014 IEEE*, May 2014, pp. 1–9.

[17] T. Wood, K. Ramakrishnan, J. Hwang, G. Liu, and W. Zhang, "Toward a software-based network: integrating software defined networking and network function virtualization," *Network, IEEE*, May 2015.

[18] R. S. Ross, "Guide for conducting risk assessments," *NIST SP800-30-Rev1 Technical Report*, 2012.

[19] "MU security posture," `http://infosec.missouri.edu/classification/definitions.html`.

[20] "UT security posture," `http://security.utexas.edu/policies/data_classification`.

[21] "USC security posture," `http://itservices.usc.edu/securityservices/`.

[22] A. Taha, R. Trapero, J. Luna, and N. Suri, "Ahp-based quantitative approach for assessing and comparing cloud security," in *IEEE TrustCom*, 2014.

[23] M. Corpuz and P. H. Barnes, "Integrating information security policy management with corporate risk management for strategic alignment," in *World Multi-Conference on Systemics, Cybernetics and Informatics*, 2010.

[24] B. Solhaug and K. Stølen, "Preservation of policy adherence under refinement," *Int. J. Software and Informatics*, vol. 5, pp. 139–157, 2011.

[25] "Incommon," `https://www.incommon.org/`.

[26] "OpenID," `http://openid.net/`.

[27] "X.509 Specification," `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf`.

[28] R. Ananthakrishnan, J. Bryan, K. Chard, I. Foster, T. Howe, M. Lidman, and S. Tuecke, "Globus Nexus: An identity, profile, and group management platform for science gateways and other collaborative science applications," in *IEEE CLUSTER*, 2013.

[29] B. Baker, K. Borne, T. Handley, J. Kantor, J. Hughes, R. Lambert, C. L. Lee, H. Larrieu, and R. Plante, "LSST data management cybersecurity draft plan," 2015.

[30] "Open Science Grid," `http://www.opensciencegrid.org/`.

[31] "GENI," `https://www.geni.net/`.

[32] "Minimum security requirements for federal information and information systems," NIST, Tech. Rep., 2006.

[33] "Harvard security posture," `http://sites.gse.harvard.edu/its/data-classification`.

[34] M. Masse, "REST API design rulebook," *O'Reilly Media ISBN: 978-1-4493-1050-9*, 2012.