Usage Control, Risk and Trust*

Leanid Krautsevich[1,2], Aliaksandr Lazouski[1,2], Fabio Martinelli[2], Paolo Mori[2], and Artsiom Yautsiukhin[2]

[1] Department of Computer Science
University of Pisa
[2] Istituto di Informatica e Telematica
Consiglio Nazionale delle Ricerche

Abstract. In this paper we describe our general framework for usage control (UCON) enforcement on GRID systems. It allows both GRID services level enforcement of UCON as well as fine-grained one at the level of local GRID node resources. In addition, next to the classical checks for usage control: checks of conditions, authorizations, and obligations, the framework also includes trust and risk management functionalities. Indeed, we show how trust and risk issues naturally arise when considering usage control in GRID systems and services and how our architecture is flexible enough to accommodate both notions in a pretty uniform way.

1 Introduction

Usage control (UCON) is a conceptual model, developed by Park and Sandhu (e.g. see [25]), which is able to embody and encompass most of existing access control models. The main features are attribute mutability that allows a flexible management of policies and continuity of the usage decision process, i.e. the resource access has a duration and the specific authorization factors must continuously hold. This enhanced flexibility w.r.t. the usual access control frameworks, where, for instance, authorizations are checked once before the access, induces several opportunities as well as new challenges.

Usage control seems a particularly suitable model for managing resources in GRID systems. Those systems often consist of federations of resource providers and users, with many long-lived executions and several conditions and factors to be considered during the usage decision process. For instance, it is common to have GRID computations lasting for hours/days. During the access it is possible that conditions that were satisfied when the access to the computational resources was requested, change by demanding a revocation of access to the resource itself.

GRID systems allow for remote execution of code, where the user that submitted the code is often a priori unknown. This feature demands for both coarse grained usage control, managing the access to the GRID services (also taking

* This work has been partially supported by the EU FP7 project *Context-aware Data-centric Information Sharing* (CONSEQUENCE).

S. Katsikas, J. Lopez, and M. Soriano (Eds.): TrustBus 2010, LNCS 6264, pp. 1–12, 2010.

into account the service workflow), and fine grained control on the interactions of the code with the resources of the GRID node.

Managing resources offered by several providers to several users demands also for mechanisms to represent trust relationships, and monitor their evolutions during the system computations. Indeed, it is possible to apply stricter policies depending on the trust level of certain users as well as to reduce the trust level of users that do not respect security policies as initially declared.

Also risk naturally arises when dealing with access and usage control in GRID services in several dimensions. For instance, a wrong access decision may have a negative impact on the resources of GRID providers as well as a wrong access revocation on GRID users. While the model defines a continuous monitoring of decision factors (authorizations, conditions and obligations), in order to ensure that these factors are satisfied even when the access is in progress, from a practical perspective this control often must be implemented by a sequence of discrete events, by introducing the possibility that the decisions are not precise enough.

The paper is organized as follows. Section 2 shows a possible scenario where the notions of usage control, trust and risk naturally arise in a GRID framework. Section 3 illustrates the application of usage control to GRID systems. Section 4 shows how our architecture considers also trust management languages. Section 5 shows potential areas where risk plays a central role and how we embedded it in our framework. Finally, Section 6 concludes the paper and recalls some related work.

2 A Possible Scenario

Consider an Italian university (ItUni) which is involved in a huge GRID project. There are several servers which apart of some internal jobs provide their computational resources (i.e., CPU cycles, memory, disk space) for heavy computations of other Italian universities belonging to the same GRID. Sometimes other research organisations, which are also the subjects of the Italian Ministry of Education, ask for access to the resources. Thus, the first challenge for ItUni is to map unknown subjects to the roles defined in its domain using trust chains defining indirect relationships between those subjects and ItUni.

ItUni allows usage of its resources only to the users and computational jobs which can be considered not very risky. This risk can be connected with a number of possible threats, e.g., normal operation of the servers can be stopped; more resources than asked can be consumed; important data stored or processed by internal jobs can be stolen or destroyed, etc. On the other hand, ItUni is paid for allowing other jobs to run on its servers. Moreover, in order to use the GRID resources for its own purpose the organisation has to allow certain amount of jobs to be processed by its servers (and get some amount of points).

There are two ways of checking the riskiness of granting access to a specific subject: use trust/reputation and/or risk analysis. Trust and Reputation management helps ItUni to allow access only to the subjects which do not usually abuse granted privileges according to the past experience of GRID nodes. Risk

analysis, next to taking into account past behaviour of subjects, also considers possible losses and benefits of every access. Since sometimes GRID jobs last for several days it is not enough to check reputation or risk ones before granting the access. This decision must be constantly re-evaluated during the whole session and the access should be revoked if further operation becomes too risky.

3 Usage Control in GRID

GRID technology enables resources sharing within a heterogeneous, dynamic, and distributed computational environment through Virtual Organization (VO) [9,11]. The Open GRID Service Architecture (OGSA) [13], defined a standard for sharing resources on the GRID that is based on the concept of GRID services. GRID services are designed to broadcast, provide and compose the available resources to the VO's members. The Globus Toolkit 4 (GT4) [10] is the reference implementation of GRID middleware according to the OGSA standard. The GRID Security Infrastructure (GSI) [23] implemented in GT4 provides a set of mechanisms and tools to manage accesses to GRID services. However, this support is not adequate for the kind of resource sharing defined by the GRID. For example, the default authorization in GT4 implies that an access decision is evaluated only once before starting the service, and no further controls are executed while the access is in progress. However, the usage of a services in GRID could be long-lived, lasting hours or even days. Let us suppose that the access to a service is granted if the requestor's reputation is above a given threshold. During the service execution, the reputation can be lowered as the result of other activities of the requestor. Meanwhile, one-time authorization does not affect granted rights and can not revoke the service execution because of the lowered reputation of the requestor. Hence, the security framework should allow the resource provider to specify that some factors should be reevaluated continuously during the service execution.

In [21,20,4], we advocated the adoption of the UCON model for continuous control of services usage in GRID. The Usage Control (UCON) model is a new model that encompasses and extends the existing access control models [25,30]. Its main novelty is the continuous enforcement of the security policy during the access time, because besides classical attributes, UCON also defines *mutable* attributes, whose value is updated as a consequence of accesses performed by subjects on objects. Moreover, besides the classical authorizations, UCON also introduces two new decision factors that are evaluated in the decision process: conditions and obligations. A more detailed description of usage control can be found in [17].

In our framework [21,20,4], we applied the usage control model at two levels: over GRID applications formed by a workflow of GRID services invocations, and over computational GRID services. The usage control over GRID applications monitors the workflow of services invocations, i.e., the invocation to GRID services performed by the GRID users, and the security policy defines the allowed pattern of invocations along with conditions and obligation that should

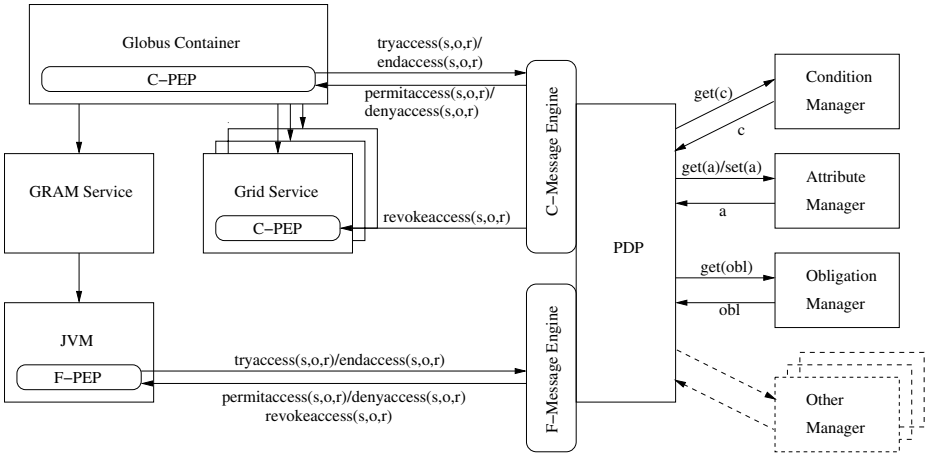

Fig. 1. Architecture of authorization framework

be satisfied before and/or during the service execution. The monitoring of GRID computational service, instead, concerns the applications that are run by remote GRID users on the local GRID node, and checks the usage of the underlying resources allocated for the service instance by the resource provider. Hence, at first the workflow level determines whether the GRID user has the right to access the service and monitors that this right is still valid while the service is running. The computational service, that in Globus is implemented by the GRAM service, is protected by a further level of control, the fine-grained one, that checks that the job submitted for execution satisfies the usage control policy while running.

The architecture we defined for the GRID service usage control is integrated in the Globus one, as shown in Figure 1, and consists of two main components: a Policy Decision Point (PDP), and a set of Policy Enforcement Point (PEP).

The PDP is the component that performs the decision process, by evaluating the security policy for each access request coming from a PEP. The PDP loads the security policy from a repository and every time an access decision is taken, updates the current state of the policy. The PDP is invoked by PEPs through the $tryaccess(s, o, r)$ control action every time a subject s requests to perform an operation r on an object o, and every time an access terminates, sending the $endaccess(s, o, r)$ control action. Moreover, the PDP continuously evaluates ongoing authorizations, conditions, and obligations according to the security policy. Indeed, the PDP invokes the proper PEP to terminate the service through the $revokeaccess(s, o, r)$ control action when an ongoing factor of the policy does not hold any more,

To process an access request, the PDP needs to evaluate various factors, and some of these factors are not directly managed by the PDP, but by some other components of the architecture. Hence, the PDP can be viewed as an orchestrator of various policy managers. Every time when a policy requires the

evaluation of a factor that is not managed by the PDP, the PDP issues a request to the proper manager. Moreover, the PDP could also send some feedbacks to the managers, to affect their internal status. In the architecture shown in Figure 1, the PDP interacts with the Condition Manager, the Attribute Manager, and the Obligation Manager. However, other managers could be integrated in the architecture. For instance, Section 4 describes the integration of the RTML manager, that deals with user Trust and Reputation, and Section 5 presents the integration of a Risk manager.

The Condition Manager is invoked by the PDP every time a security policy requires the evaluation of environmental conditions. The Condition Manager interacts with the underlying system to get environmental information. As an example, the Condition Manager could retrieve the current time, workload, free memory, and so on. The Attribute Manager is in charge of retrieving fresh values of attributes. When PDP needs a value of subject's or object's attribute, it invokes the Attribute Manager. The PDP also exploits the Attribute Manager to update values of attributes. The Obligation Manager monitors the execution of obligations. If an obligation is controllable, the Obligation Manager can ask the subject to perform it. Instead, if an obligation is observable only, the Obligation Manager simply checks it. The Obligation Manager returns true to the PDP if the obligation is satisfied, or false otherwise.

The PEP components are integrated in the Globus architecture components to intercept invocations of security-relevant operations, send the $tryaccess(s, o, r)$ control action to the PDP, and suspend the operation waiting for the decision of the PDP. Each PEP enforces the decisions received by the PDP by executing or skipping the required security-relevant operation. Moreover, each PEP is also able to detect when a security-relevant operation terminates, to issue the $endaccess(s, o, r)$ control action to the PDP, and to force the termination of the operation while it is still in progress to enforce the $revokeaccess(s, o, r)$ control action. The PEPs designed to intercept service invocations are different from the ones that controls applications executed by GRID computational services. These PEPs manage distinct sets of actions on distinct objects, and they are integrated in different components of the architecture. The PEP that intercepts all requests for GRID services done by remote GRID users (C-PEP), consists of two parts. The first part is embedded in the Globus container and enforces PDP decisions done before starting the service. The second part is active during the usage and is responsible for continuous policy enforcement, and revokes access if usage conditions are not satisfied. In our architecture, a fine-grained PEP (F-PEP) is designed to monitor the execution of Java computational jobs. The F-PEP is integrated into Java Virtual Machine (JVM) that executes Java jobs submitted by GRID users, and intercept the operations that applications try to perform on the resources of the underlying machine. The interactions among PEPs and PDP are executed according to a message exchange protocol, and are implemented through two messages dispatchers, C-Message and F-Message engine, for the two levels.

4 Trust and Reputation Management with Usage Control for GRID

Besides the need of monitoring the usage of the shared resources after the access right has been granted to GRID users, security management in GRID environment also requires to establish secure relationships between GRID resource providers and users. As a matter of fact, direct trust relationships among GRID users may not exist a priori, because during the VO formation phase, no trust relationships are required to be in place among the parties that are joining the organization. Hence, the security support should be able to evaluate whether an unknown VO member that requires to access the local resources is reliable enough to grant him the access. Moreover, the access should be granted even if the user didn't subscribe to the GRID node before. The standard authorization system provided by the Globus Toolkit is static, because the identity of each authorized user is statically mapped on a local account that is exploited to execute remote jobs on behalf of the user. Hence, given a GRID resource R, only the users that have been previously registered on R can access the services provided by R. This feature is a limitation in an open and distributed environment such as GRID.

In [6,5], we proposed to exploit Trust and Reputation management to enhance the GRID security support. In particular, in [6] we proposed the integration of an extended version of the RTML framework in the Globus authorization infrastructure to grant access rights taking into account the trust relationships that each VO member collects as a consequence of his interactions with other GRID resources. The Role-based Trust Management framework (RTML) [18], [27] combines the strength of Trust-Management (TM) and Role-Based Access Control (RBAC), providing policy language, semantics, deduction engine, and concrete tools to manage access control and authorization in large-scale and decentralized systems. The trust relationships are expressed in form of credentials, issued by the GRID sites, that define the roles that a given user holds in those domains. Each role is paired with a set of privileges on the resources of the domain, and it is enforced by creating new local accounts with the corresponding set of rights. The set of credentials is dynamic, because new credentials could be added by other GRID sites and some of the existing ones could be expired. When an unknown user asks to access the GRID services provided by a given domain, he submits the credentials that grant him some roles in some (other) domains. The RTML authorization system computes the roles that the user holds in the local domain by combining the credential submitted by the user with the local access rules, that represent the trust relationships that the local domain has with other organizations. If at least one of the roles found by the RTML system grants the right to access the requested service, than the access is allowed, otherwise is denied.

Example 1. Figure 4 shows an example of RTML user's credentials and provider's access rules. In this case, the user Alice has the role *GRIDAdmin* in the domain *CNR*, and the role *user* in the domain *ERCIM*. Alice exploits these credentials

1. *CNR.GRIDAdmin(mat='12345', firstname='alice', lastname='rossi')* ← *Alice*
2. *ERCIM.user(number='IS137', firstname='alice', lastname='rossi')* ← *Alice*

User's credentials

1. *ItUni.GRIDGuest* ← *CNR.GRIDAdmin(mat=?,firstname=*ref_{first}*,lastname=*ref_{last}*)* ∩ *ERCIM.user(number=?, firstname=*ref_{first}*, lastname=*ref_{last}*)*

Provider's access rules

Fig. 2. Example of user's credentials and provider's access rules in RTML

to access the services of domain *ItUni*, because the access rules of this domain grants the role *GRIDGuest* to those users that have the roles *GRIDAdmin* in *CNR*, and *user* in *ERCIM*.

In [5], we enhanced the previous approach by extending the RTML framework to deal with a quantitative notion of reputation, and by integrating the resulting framework within a preliminary prototype of fine-grained usage control system for GRID. Hence, the resulting architecture has the capabilities of: *i)* allowing the access to GRID computational resources to unknown users on the basis of the reputation paired to them as a result of their previous interactions with other (trusted) GRID nodes, *ii)* monitoring the operations executed by the user's applications on the local resources once the access right has been granted according to the Usage Control model and also exploiting Trust and Reputation as decision factors collected by the PDP to perform the decision process, and *iii)* providing feedbacks on the behaviour of the user that define the new reputation of the user according to this GRID provider. These feedbacks (represented as credentials) will be exploited by other GRID nodes to decide whether to grant or not some access rights to the user in their domains.

From the architectural point of view, the extended version of the RTML framework has been interfaced with the PDP like the managers of other decision factors, and it is invoked by the PDP, when the security policy requires the evaluation of the roles held by a specific user or of his Reputation. The RTML framework exploits both the credentials submitted by the user and the ones stored in the local repository of the GRID node to infer the roles held by the GRID user and his Reputation. The result is returned to the PDP that combines it with the other factors expressed in the policy to determine the required access right.

5 Approach for Risk-Based Usage Control

Allowing access to a resource is always connected with a risk that the resource will be abused. It is not always possible to allow only trusted subjects to access a resource. This problem becomes even more important when we allow access for unknown users (using Trust Management, see Section 4). Such assignment

could be too coarse. Therefore, we would like to allow access only to the subjects which, we believe, will not abuse its privileges. One way of doing this is to use reputation of the subject, as shown in Section 4. A more explicit way of taking this risk into account is to compute possible losses, using a well-known formula: $loss = probability \times impact$ and compare these losses with potential benefits.

The behaviour of the system we consider is the following. A resource provider and a resource consumer (subject) make an agreement about the usage of a resource (object). The resource consumer pays the agreed amount in advance the the resource provider shares the resources. In case the resource provider decides that the access should be revoked it pays money back (full amount). Naturally, the resource provider suffers from some losses if the resource consumer abuses its privileges as well as when the access is revoked while the resource consumer behaves well (loss of reputation). Revoking access of a resource consumer which misbehaves does not result in any loss (though, some benefit can be assigned to this case if needed).

Let C_B be the benefit a resource provider gets if it allows access to its resources; C_L be the losses the resource provider suffers if the resource consumer abuses its privileges in some way; C'_L be the losses the resource provider suffers if it revokes the access when the resource consumer behaved well; and p be the probability that the privileges will be abused by the consumer. In order to make a rational decision we should compare the resulting cost in case access is allowed with the cost when the access is revoked. Algebraically, we can show this relation for access decision making as follows:

$$C_B - C_L * p \lesseqgtr -C'_L(1-p) \tag{1}$$

The decision to allow access should be made if the left part of the Equation 1 is greater than the right one. Such situation means that it is more profitable to allow access rather than to deny access. Naturally, the access should be denied/revoked if we have the reverse condition.

A subject can abuse its privileges in different ways (make a server unavailable, install a back door, damage sensitive data, etc.) and every specific abuse cause different amount of losses. Thus, we need to consider every loss separately (C^i_L). In order to collect and process statistics required for determining the probabilities of some violation (p^i) we can use a limited set of such violations. Now, every entity in the GRID can collect such statistics about violations done by its users and share this statistics with other providers. E.g., in the simplest case we can use three well-known aspects of security for identifying these types of violation: i) loss of Confidentiality; ii) loss of Integrity iii) loss of Availability similar to [7]. For a more fine-grained analysis more elaborated list is required.

Equation 1 can be rewritten as follows:

$$C_B - \sum_{\forall i} C^i_L * p^i \lesseqgtr -C'_L \prod_{\forall i}(1-p^i) \tag{2}$$

Access control models empowered with risk analysis [28,7,22,2] use a similar strategy to decide if access should be allowed or denied. Such model is based

on the idea that the required values (benefits, probabilities, and costs) do not change during the whole session. But GRID projects may last for hours and days. Some parameters may change during a usage session and the validity of access decision made long time ago must be reconsidered. Moreover, existing access control models consider that access decision is based on attributes of the resource requestor only. In fact, in some situations environment and attributes of other resource requestors may affect the decision making process.

There are two parameters which can change during the access session: benefit and amount of losses. These values often depend on the attributes of other partners and environmental conditions. Naturally, probability of some misbehaviour for a concrete subject also may change, though this change happens not so sharply.

Example 2. Benefit, which ItUni gains, can be increased if there is a need for the ItUni to receive points to use the GRID for its purposes (points are given for allowing other partners to use the resources and consumed when resources of others are used). Benefits can be decreased if one of primary partners asks for resources while a third party uses most of them. On the other hand, losses can be increased when some sensitive data are processed by the servers.

In other words, losses and benefits are not constant during the access session, but a result of some function which depends on some attributes. Both these function require knowledge of the context and can be defined precisely only by experts. Sometimes it is a manager which simply should change the values. Note, that this change does not required rewriting of policies and evolvement of security staff. The probabilities of violation are simply attributes of subjects.

Example 3. The computation of losses in ItUni can be a simple summing function, which sums up losses of all sensitive data processed in by the servers. The need of points can be expressed as the losses caused by idle of the projects ItUni wants to execute with GRID. This value is added to the overall payment (benefit) for usage of the resources.

A risk manager can be easily added to our architecture shown in Figure 1. This manager is just another manager block. In order to make a decision about the access the risk manager needs to know: 1) what kind of threats must be taken into account; 2) functions or values for losses and benefits and required attributes; 3) probabilities of violation, which are just attributes of the subject. Possible threats and functions for benefits and losses are described in access policies. Current values required for computation of risk are taken from the PDP which, in its turn, gets attributes from the Attribute and Condition Managers. As a result, the risk manager returns a boolean result to the PDP: allow or deny access.

6 Conclusion and Related Work

In the paper we described our architecture for usage control on GRID. This architecture allows to check conditions, authorisations, and obligations during

various time of a session. Moreover, the architecture also supports the trust management functionality. This functionality allows to grant access to unknown subjects on the basis of the trust relationships that he collected with other members of the VO. Reputation is added to the functionality of the architecture in order to deny access for those subjects which are considered not trustworthy enough. Finally, risk assessment helps to make a rational decision about access based on the analysis of losses and benefits. The architecture also considers that some parameters may change in time (e.g., reputation and risk) and, thus, provides more flexibility in access management.

There is some related work on the topics of this paper. As a matter of fact, some attempts have been done to enhance GRID security adopting standard authorization engines, such as VOMS, CAS, PERMIS and Akenti [1,3,12,26], but none of them provide continuous monitoring of service usage. Instead, in [29] the inventors of UCON also propose the adoption of their model in collaborative computing systems, such as the GRID. Their approach is based on a centralized repository for attribute management, and the authors adopt the push mode for immutable attributes, i.e. these attributes are submitted to GRID services by the user itself, and the pull mode for mutable attributes, i.e. the values of these attributes are collected when the policy evaluation is executed. Security policies are expressed using XACML.

Although risk assessment has been applied to access control by several authors, we are only aware of two papers on UCON and risk, namely [16,15]. In these papers we considered different aspects of applying risk in the UCON model: in [15] we assessed the risk caused by unfresh values of attributes, while in [16] the problem was to select the less risky service provider and control usage of client's data after granting the access. On the other hand, the notion of risk has been applied to several access control models. Indeed, risk for access control can be used as a static parameter to help a policy designer to assign privileges having possible losses in mind [19,14]. But a more promising scenario is when risk is considered to be dynamic [28,7,22]. N. N. Diep et. al., [7] explicitly show that risk should be competed and the decision is made comparing the risk value with a threshold. On the contrary, L. Zhang et. al., [28] R. W. McGraw [22] do not pay attention to how risk is computed, but state that the decision should be made comparing possible losses and benefits. Q. Ni et. al., [24] considered most of the parameters required for computation of risk as static, but used access quota which reduces with increasing of the amount of performed actions (aggregating risk of these actions). This change of quota can be seen as the change of the benefit value in terms of our model. There are also several papers which pay more attention to how to integrate risk in access control policies rather than how to use risk for making an access decision [2,8].

References

1. Alfieri, R., Cecchini, R., Ciaschini, V., dell Agnello, L., Frohner, A., Gianoli, A., Lorentey, K., Spataro, F.: VOMS: An authorisation system for virtual organizations. In: Proceedings of 1st European Across Grid Conference (2003)

2. Aziz, A.B., Foley, A.S., Herbert, A.J., Swart, A.G.: Reconfiguring role based access control policies using risk semantics. Journal of High Speed Networks 15(3), 261–273 (2006)
3. Chadwick, D., Otenko, A.: The PERMIS X.509 role-based privilege management infrastructure. In: Seventh ACM Symposium on Access Control Models and Technologies, pp. 135–140. ACM Press, New York (2002)
4. Colombo, M., Lazouski, A., Martinelli, F., Mori, P.: Controlling the usage of grid services. International Journal of Computational Science (2010)
5. Colombo, M., Martinelli, F., Mori, P., Petrocchi, M., Vaccarelli, A.: Fine grained access control with trust and reputation management for globus. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part II. LNCS, vol. 4804, pp. 1505–1515. Springer, Heidelberg (2007)
6. Colombo, M., Martinelli, F., Mori, P., Vaccarelli, A.: Extending the globus architecture with role-based trust management. In: Moreno Díaz, R., Pichler, F., Quesada Arencibia, A. (eds.) EUROCAST 2007. LNCS, vol. 4739, pp. 448–456. Springer, Heidelberg (2007)
7. Diep, N.N., Hung, L.X., Zhung, Y., Lee, S., Lee, Y.-K., Lee, H.: Enforcing access control using risk assessment. In: ECUMN '07: Proceedings of the Fourth European Conference on Universal Multiservice Networks, Washington, DC, USA, pp. 419–424. IEEE Computer Society, Los Alamitos (2007)
8. Dimmock, N., Belokosztolszki, A., Eyers, D., Bacon, J., Moody, K.: Using trust and risk in role-based access control policies. In: Proceedings of the 9th ACM Symposium on Access Control Models and Technologies, pp. 156–162. ACM, New York (2004)
9. Foster, I.: The anatomy of the grid: Enabling scalable virtual organizations. In: Sakellariou, R., Keane, J.A., Gurd, J.R., Freeman, L. (eds.) Euro-Par 2001. LNCS, vol. 2150, p. 1. Springer, Heidelberg (2001)
10. Foster, I.: Globus toolkit version 4: Software for service-oriented systems. In: Jin, H., Reed, D., Jiang, W. (eds.) NPC 2005. LNCS, vol. 3779, pp. 2–13. Springer, Heidelberg (2005)
11. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The physiology of the grid: An open grid service architecture for distributed system integration. Globus Project (2002), http://www.globus.org/research/papers/ogsa.pdf
12. Foster, I., Kesselman, C., Pearlman, L., Tuecke, S., Welch, V.: A community authorization service for group collaboration. In: Proceedings of the 3rd IEEE Int. Workshop on Policies for Distributed Systems and Networks (POLICY 2002), pp. 50–59 (2002)
13. Foster, I., Kishimoto, H., Savva, A., Berry, D., Djaoui, A., Grimshaw, A., Horn, B., Maciel, F., Siebenlist, F., Subramaniam, R., Treadwell, J., Reich, J.V.: The open grid service architecture (ogsa), version 1.5. Open Grid Forum Document Series: GFD-I.080 (2006), http://www.ogf.org/documents/GFD.80.pdf
14. Han, Y., Hori, Y., Sakurai, K.: Security policy pre-evaluation towards risk analysis. In: Proceedings of the 2008 International Conference on Information Security and Assurance (ISA 2008), Washington, DC, USA, pp. 415–420. IEEE Computer Society, Los Alamitos (2008)
15. Krautsevich, L., Lazouski, A., Martinelli, F., Yautsiukhin, A.: Risk-aware usage decision making in highly dynamic systems. In: Proceedings of the Fifth International Conference on Internet Monitoring and Protection, Barcelona, Spain (May 2010)

16. Krautsevich, L., Lazouski, A., Martinelli, F., Yautsiukhin, A.: Risk-based usage control for service oriented architecture. In: Proceedings of the 18th Euromicro Conference on Parallel, Distributed and Network-Based Processing. IEEE Computer Society Press, Los Alamitos (2010)
17. Lazouski, A., Martinelli, F., Mori, P.: A survey of usage control in computer security. Computer Science Review (4), 81–99 (2010)
18. Li, N., Mitchell, J., Winsborough, W.: Design of a role-based trust management framework. In: Symposium on Security and Privacy, pp. 114–130. IEEE Computer Society, Los Alamitos (2002)
19. Li, Y., Sun, H., Chen, Z., Ren, J., Luo, H.: Using trust and risk in access control for grid environment. In: Proceedings of the 2008 International Conference on Security Technology, Washington, DC, USA, pp. 13–16. IEEE Computer Society, Los Alamitos (2008)
20. Martinelli, F., Mori, P.: On usage control for grid systems. Future Generation Computer Systems 26(7), 1032–1042 (2010)
21. Martinelli, F., Mori, P., Vaccarelli, A.: Towards continuous usage control on grid computational services. In: ICAS-ICNS '05: Proceedings of the Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services, p. 82. IEEE Computer Society, Los Alamitos (2005)
22. McGraw, R.W.: Risk-adaptable access control (radac), `http://csrc.nist.gov/news_events/privilege-management-workshop/radac-Paper0001.pdf` (September 16, 2009)
23. Nagaratnam, N., Janson, P., Dayka, J., Nadalin, A., Siebenlist, F., Welch, V., Foster, I., Tuecke, S.: Security architecture for open grid services. Global Grid Forum Recommendation (2003)
24. Ni, Q., Bertino, E., Lobo, J.: Risk-based access control systems built on fuzzy inferences. In: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, pp. 250–260. ACM Press, New York (2010)
25. Park, J., Sandhu, R.: The UCON ABC usage control model. ACM Transactions on Information and System Security (TISSEC) 7(1), 128–174 (2004)
26. Thompson, M., Essiari, A., Mudumbai, S.: Certificate-based authorization policy in a pki environment. ACM Transactions on Information and System Security (TISSEC) 6(4), 566–588 (2003)
27. Winsborough, W., Mitchell, J.: Distributed credential chain discovery in trust management. Journal of Computer Security 11(1), 36–86 (2003)
28. Zhang, L., Brodsky, A., Jajodia, S.: Toward information sharing: Benefit and risk access control (barac). In: Proceedings of the 7th International Workshop on Policies for Distributed Systems and Networks, Washington, DC, USA, pp. 45–53. IEEE Computer Society, Los Alamitos (2006)
29. Zhang, X., Nakae, M., Covington, M.J., Sandhu, R.: Toward a usage-based security framework for collaborative computing systems. ACM Transactions on Information and System Security (TISSEC) (2008)
30. Zhang, X., Parisi-Presicce, F., Sandhu, R., Park, J.: Formal model and policy specification of usage control. ACM Transactions on Information and System Security (TISSEC) 8(4), 351–387 (2005)