Marty Humphrey
University of Virginia
Mary Thompson
Lawrence Berkeley National Laboratory

October 2000

Security Implications of Typical Grid Computing Usage Scenarios

<draft-gridforum-security-implications-01.doc>

Status of this Memo

This memo provides information for the Grid Forum community. This memo does not specify a Grid Forum standard of any kind. Distribution of this memo is unlimited.

Abstract

There are many ways to access the immense computational power of a Computational Grid, each with unique security requirements and implications for both the resource user and the resource provider. This document presents a representative set of Grid usage scenarios and accompanying analysis with regard to security requirements such as authentication, authorization, integrity, and confidentiality. The main value of these scenarios and the associated security discussions are to provide a library of situations against which an application designer can match, thereby facilitating security-aware application use and development from the initial stages of the application design and invocation. A broader goal of these scenarios is to increase the awareness of security issues in Grid Computing.

Marty Humphrey
University of Virginia
Mary Thompson
Lawrence Berkeley National Laboratory
October 2000

1 Introduction

Computational Grid Infrastructure software such as Globus [1] or Legion [2] enable a user to identify and use the fastest available machine irrespective of location and ownership and can facilitate a user spreading a single large computation across multiple machines, again irrespective of physical location. However, without an adequate understanding of the security implications of a Grid, both the Grid user and the system administrator who contributes resources to a Grid can be subject to significant compromises in security.

A Grid is comprised of a collection of hardware and software resources whose origins may not be obvious to a Grid user. When a user wants to run on a particular machine, the user needs assurances that the machine has not been compromised, making his proprietary application or data subject to being stolen. When a user's job executes, it may require confidential message-passing services. A user or the Grid infrastructure software may set up a long-lived service such as a specialized scheduler and require that only certain users are allowed to access the service. In each of these cases, the application must anticipate and be designed to provide this required security-related functionality, and the invoker of these applications must understand how to check if these security services are available and how they can be invoked.

The purpose of this document is to present various Grid Computing usage scenarios and analyze their security approaches and implications. These scenarios are designed to provide guidance for the Grid user, the Grid application developer, and the Grid resource provider. For the Grid user, these scenarios describe the security implications related to the interaction with existing components. For the Grid application developer who wishes to design and deploy an application for use in a Grid and does not know "where to begin" with regard to computer security, these scenarios provide a library of cases to which to compare. For the resource provider, these scenarios describe what can be expected of applications that may run on their resources, specifically with regard to interaction with other parts of the Grid and the local machine itself. In general, the intent of these scenarios and their analyses is to foster the development and deployment of interoperable security-aware Grid applications from first designs, eliminating the need to redesign and "patch" applications to accommodate the security concerns arising from large-scale deployment and availability.

This document is an informational Grid Forum draft. It is intended to provide information; there are no requirements or specifications in this document. Section 2 of this document describes the necessary steps prior to a Grid session. Section 3 describes the scenarios. Each scenario is discussed in turn with respect to the security implications of the usage scenario. Section 4 contains a list of contributors we wish to thank. Section 5 provides a list references. Section 6 discusses security considerations. Section 7 provides contact information for the editors.

2 Necessary Steps Prior to Grid Session

This section makes explicit some of the underlying assumptions about the kind of infrastructure that is required to support secure interactions between Grid users and services. The most fundamental of these assumptions is that each user and principal will have a Grid-wide identity that all the other Grid principals can verify. Another assumption is that *some* local resource managers will require legacy local user ids for users of their resources, so there must be a way to map from Grid IDs to local userids. Access control will be enforced both by local resource managers often using legacy access control mechanisms and by Grid aware services that may want to use Grid-centered access policies. In either case there must be simple ways for users to request access rights and allocations and the stakeholders to grant them. The use of short-term proxy certificates in place of the long term Grid ID is a desirable feature of a distributed system, since it limits the exposure of long-term private keys. The delegation of rights and/or identity from a user to the servers operating on his behalf is also required. Finally, a user may want a way to specify security options for a session.

Most computing resources that are shared among a group of users require that a user have established a unique ID and allocation before he can use the resource. When using a Grid, a person wishes to have an ID that is unique within the Grid and recognized by all the local resource managers. Globus uses X.509 identity certificates that can either be issued by a Grid-centric Certificate Authority or by individual corporate CA's for Grid identities. Legion has a similar capability to issue an unforgeable "token" which a user can present to identify himself as a Legion user. A CA can require different levels of verification that the person requesting the certificate (usually via email) is in fact the person named in the certificate (such as showing up in person, presenting various documents such as a driver's license, passport, and/or background check). Currently neither Globus nor Legion require or support certification beyond a rudimentary level. Both Globus and Legion also support Grid IDs established through external means such as the enterprise-wide Kerberos [3] installations that exist in certain national laboratories.

A Grid is likely to have multiple CAs in place. If so, a crucial step is for the each CA to carefully establish a policy for its signed certificates, and for the CAs to cross-certify each other if possible. That is, a CA must clearly identify what, if anything, another CA's certificates mean to it. The means to do this are currently the subject of a Grid Forum Security Working Group sub-group.

Once the Grid ID is established, a person may request allocations on each machine on which she might potentially execute a task. Some sites (such as supercomputer centers) may require that each individual have a local user ID and allocation, other sites may allow group allocations or simply require that a user be permitted to use the resource possibly in a constrained manner (e.g. only on weekends or late nights). Establishing permissions and allocations on a resource depends on the resource owner's policy and may require sending email to the system administrator of the resource in question. For each resource that requires local user ids, the Grid system administrator on the resource must determine the mapping of Grid ID (such as smith@NAVO.HPC.MIL*)* to local user ID (such as *bsmith)* into a Grid-specific mechanism such as the **/etc/LegionUsers** file in

Legion and **/etc/gridmap** file in Globus. Note that Legion has the capability to map more than one Grid ID to a single local ID—while this increases flexibility in that a user does not have to have a specific account established on a machine in order to run on that machine, it makes accountability more difficult. The availability of allocation remaining must be known to the Grid services—the security implications of this are discussed in the scenarios of Section 3.

The need for short-term proxy certificates arises from the fact that the client tends to interact with Grid services through a sequence of requests over the network to various entities. During each of these transactions the client must present his Grid ID and use his private key to verify his identity. In order to minimize the use of long-term credentials, it is desirable to allow the user to set up a short-term (typically 12 hours) proxy credential to represent the user in interactions with the Grid. This short-lived keypair can be kept unencrypted by the local Grid software and can be used for signing, verifying, and (optionally) encryption. A second certificate is generated, signed by the long-lived keypair, stating that for the next 12 hours the public key of the user is the public key of the short-lived pair. The Globus security infrastructure software supports this through *grid-proxy-init*.

The last steps a person may take before interacting with services or resources on the Grid is to establish parameters that may be in place for the life of the session. For example, a person may specify her specific role that she wants to assume, such as system administrator for a particular resource or ordinary user. There are different rights and responsibilities for each role. In addition, a person may also set security-related parameters, such as the hosts that are trusted, the level of message integrity and confidentiality that is required and a limit to the amount of allocation that can be used (as discussed in Section 3.5).

3 Use Scenarios

These scenarios are roughly ordered by increasingly complexity, where complexity is loosely defined as the number of decisions that must be made by the Grid software or the scope of those decisions. Six categories are described: immediate job execution, job execution that requires advance scheduling, job control, accessing grid information services, setting or querying security parameters, and auditing use of Grid resources. The unique security implications of each scenario is discussed in turn.

In these scenarios the term **Grid user** or **user** refers to the person who is attempting to access a resource; **principal** is used to mean any entity, either human or process that has an identity associated with it and wants to make use of or to provide resources; **stakeholders** are people or organizations who set the use policy for a resource; a **Grid gateway** is a process which accepts remote requests to use resources; a **Grid resource gateway** is the process that actually controls the use of the resource (this may be legacy code); a **Grid administrator** is a Grid-aware person with responsibility for the overall functioning of the Grid (note that there will probably exist multiple Grid administrators with non-overlapping realms of responsibility in a single Grid); and **site administrators** are responsible for the functioning of a single site. The **user's home organization** is the administartive domain to which the user belongs which may have trust relationships or service agreements with some of the resource providers.

3.1 Immediate Job Execution

3.1.1 Run a job on a specified Grid computer; local I/O required

In a basic approach to remote job submission, the user specifies the execution host to be used and submits a job where either the code already exists on the target machine or else it is uploaded as part of the request. The job uses only remote computation cycles and possibly temporary file storage, the input data is uploaded as part of the job submission, and the output (stdio) is returned through the connection that was established at job submission. The security requirements and implications in this Grid usage scenario are:

a) Mutual authentication of user and Grid gateway on specified host

b) Grid gateway on specified host must map Grid ID to local ID

c) Grid gateway must submit the request to resource gateway in a manner so that the job will run as the authorized local user

In this scenario, authorization to use the target machine is performed by the Grid gateway. While in general the resource gateway may be a separate entity from the Grid gateway, in this case it is likely that the resource gateway is the queuing system on the target machine, and does not need to perform a specific authorization step. In this case, the target job does not have to given a copy of the credential of the user presented to the Grid gateway as part of the mutual authentication step.

3.1.2 Run a job on a specified Grid computer; non-local file I/O required

As an extension of scenario 3.1.1, the remote job must access non-local files, either by copying them from the remote site to the local site or by obtaining the non-local data as needed during the execution of the job. In addition to the security requirements and implications of Scenario 3.1.1:

a) If the model of execution is such that the Grid gateway (or some other Grid infrastructure process) performs the file transfer before execution, then this process must be given authorization to obtain these files on behalf of the user.

b) If (a) is not true, then the job directly obtains the data from the files and therefore must be given the appropriate credentials upon startup to obtain the data.

c) If the remote job writes output to the local file server, and the file server is AFS or DFS, then the remote job needs the user's Kerberos ticket (which may or may not be the same as the credentials used to authenticate to the Grid gateway)

d) If the output of the job is in the form of local files, and these files are required to be copied back to the machine from which the user submitted the job, then either the job itself or the Grid process that does the copying must be given the necessary credentials to be authorized with the Grid gateway on the local machine

In contrast to Scenario 3.1.1, in this example *delegation* is necessary to access services or information. In step (a), the Grid gateway does not automatically have permission to copy arbitrary files from the Grid environment, so it must be given the explicit capability. Similarly, in step (d), the Grid process needs the user’s credentials to be granted access to copy the local files back to the machine on which the job was submitted. In each of the delegation requirements, note that the specific right to be delegated is different. Delegation is an extremely difficult problem that is the subject of current research, so the current approach is to solve these requirements with unlimited delegation.

3.1.3 Run a job on “best” Grid computer

Instead of mandating the specific Grid computer in the request, a user may wish to run a job on the “best” computer, as defined to be the host that is “quickest’ or “cheapest” according to some metric. The choice is made by a third-party service, such as one of the emerging “Super Schedulers” as exemplified by the default scheduler in Legion. The user may specify a specific group from which to choose, or the user may leave it to the Super Scheduler to locate the set from which to choose. The additional security ramifications posed by this scenario are:

a) If the set of candidate hosts has not been identified by the user, the Super Scheduler will need to interact with the Information Services component(s) of the Grid to identify the set of possible hosts. Since access to such information is restricted, the Super Scheduler may need a delegated credential to act on behalf of the user.

b) The Super Scheduler must determine if the target user is allowed to execute on each of the target Grid machines, and, if so, the remaining allocations of the user in the particular role in which the user is acting. This information is determined by asking Information Services or querying each Grid machine directly. Again the Super Scheduler may need a delegated credential from the user.

c) The Super Scheduler needs to query the Grid hosts to determine available cycles so that it can make an informed scheduling decision.

Account information is largely regarded as being important to keep secret, so it will probably be the case that an arbitrary entity will not be allowed to ask the Information Services where a particular user can execute or how much allotment she has left. Therefore, either the Super Scheduler, as a principal, must be granted broad access to such information and trusted not to leak such information to any one except the affected user, or the Super Scheduler must be explicitly granted the right to ask on behalf of the user. See Section 4 for a discussion of the challenges of limited delegation, but in this scenario the Super Scheduler needs the rights to read information about a users access and allocations on specific machines form the Grid Information Service and possibly from the Grid hosts themselves.

3.1.4 Run a job that requires resources at multiple sites

In a scenario that many see as the defining contribution of Grid Computing technology, a user may wish to combine resources from multiple sites into a single, coordinated job. For example, a user could generate a large amount of data from a major shared instrument (e.g., accelerator or microscope study). This data needs to be uploaded to a large data store that in turn can be accessed by a powerful compute engine. Once preliminary data analysis has taken place, intermediate data may need to be saved and also passed on to a different compute engine for further analysis such as visualization procedures. This scenario requires a remote job to start other remote jobs or access files on behalf of the original user. The significant new security requirement of this scenario is:

a) A controlling agent or each remote job in a sequence has to be able to request resources on behalf of the user, perhaps through subsequent calls to a Super Scheduler.

The controlling agent or each remote job must be given a delegated credential from the user to further submit jobs on behalf of the user. If the agent has been granted a credential with unlimited delegation, it can impersonate the user for any purpose, such as a malicious activity. Thus it is more desirable for the agent to have a delegated credential that allows it to perform only limited tasks for a limited period of time. An issue that might arise here is the exact time during which the delegated credential is valid. If it expires too soon, the agent cannot perform its intended action, but if the timeout is too long, it could be a potential point of compromise. Another issue that can arise is how many levels of credential delegation should be allowed. If the user is application code that he did not write, he may not know how many levels of servers and objects are going to be instantiated. Allowing unlimited levels of delegation may be the only practical approach, but that exposes the user ‘s credential to more possible misuse.

3.2 Job Execution that Requires Advance Scheduling

Also see the “Security Requirements of the Scheduling Working Group” Draft document for scheduling scenarios

3.2.1 Make a reservation for simultaneous resources

If Scenario 3.1.4 involved the use of an instrument that produces a large data flow that must be processed in real time, it will require the advance reservation of data storage, network bandwidth and possibly compute cycles. Advance reservations require:

a) Delegation of the user’s rights to a Super Scheduler and Bandwidth Broker to make the reservations on behalf of the user.

b) Assurance that if a user has been granted a reservation for the future, she will have access at the time the reservation is claimed.

c) Bandwidth reservations usually require service agreements for priority bandwidth between ISP’s and compute sites. This implies that a bandwidth broker needs to know at reservation time that user’s connection will come from an authorized site.

If the model of execution is such that the Bandwidth Broker returns a *claim ticket* to the Super Scheduler, the transfer of the claim ticket from the Super Scheduler to the user must be protected, and the claim ticket itself must be non-forgeable.

3.2.2 Claim the reservations

The execution of a job on reserved resources can require multiple concurrent claiming procedures. In this model, a user directly interacts with the individual resource gateways to claim the reservation. In general, reservation claiming requires:

a) The user must be able to identify himself as the entity that made the reservation. The reservation may have been made on behalf of a group of users, in which case the user has to prove himself to be a member of the group. Another way of handling the situation where one person makes a reservation and a different person wants to claim it, is to allow the claim tickets to be transferred. In this case the resource gateway must be able to verify that the claim has been legitimately transferred by the person who made the original reservation to the current claimant.

b) The user should still have access to all the resources that he has reserved, except in extreme cases, such as when the user is no longer associated with the organization that is going to pay for the resource use or the organization has failed to pay its bills.

c) In the case of a user losing access to a resource, a check should be made of advance reservations in his name, and the appropriate parties should be notified of the change.

This scenario contains two important requirements in Grid Computing, *group membership* and *nonrepudiation*. Group membership is non-trivial because, while individual users should be able to define groups, it is not clear how exactly to do this. Nonrepudiation in this context refers to the requirement that the resource gateway should not be able to arbitrarily deny that it granted a reservation.

3.3 Job Control

3.3.1 Allow a user to monitor or attach to a running job

A standard requirement of people who start long-running remote jobs is to be able to disconnect from a job and then at a later time and possibly from a different location reattach to it. The user may just want to monitor the progress of a job, or may want to enter some steering information at specific points in the run. Monitoring a job's progress may be as simple as knowing where logging files are being written and having the access to read them. Steering implies that the user has defined "entry points" into the computation and has some way of controlling who may connect to them. In the collaborative environment facilitated by the Grid, a different user may want to use the monitoring or attachment points as well.

a) In this case the resource that is being protected is access to a running job created by a user, who will set the access policy and later be granted access by that policy. This can perhaps be most easily accomplished if the policy and code to enforce access is part of the job.
b) The point of entry is probably directly to the computation itself as opposed to through the Grid gateway or the resource gateway, so the potential collaborator must be able to authenticate to the computation itself.

The user in this scenario would probably rely on pre-defined libraries generated by security developers rather than creating an individual security solution. Utilizing well-accepted libraries facilitates interoperability.

3.3.2 System administrator(s) terminates out-of-control job

Certainly situations will occur in which a Grid user has submitted a job and does not realize that the job is behaving abnormally and perhaps consuming more resources that expected or allowed. Whereas the Grid Administrator(s) might be aware of an out-of-control job, only the system administrator of one or more physical resources will have the rights to terminate the job. This scenario requires:

a) A system administrator must detect the out-of-control process and trace its origin to a particular Grid user. Alternatively, Grid monitoring software might detect the out-of-control process and notify the system administrator.

b) The system administrator can optionally inform the Grid Administrators that the process is about to be terminated. The Grid Administrators need this information to coordinate the termination of this job across multiple Grid sites.

c) The Grid Administrators either atempt to terminate the individual components of the job by directly interacting with the job or by asking the system administrators to terminate those processes of the job that are on their respective machines.

d) The job owner must be notified by the Grid Administrator that his job has been terminated.

Although the steps described here are a function of the unique implementation of the Grid design, this scenario presents a number of interesting requirements. First, resources of a Grid are used both by “local” users and Grid users, so it is not necessarily obvious from where an out-of-control process originated. Therefore, Grid software must keep audit records (further discussed in Section 3.8), or at least provide a means by which this determination can be made. Second, there will generally not be a single person who has the power by which to kill a single “Grid Computation”, because it will span multiple resources of multiple administrative domains. As such, ideally, a coordinated effort must be made if a single job is to be prematurely terminated (note that this is unlikely at least in the near term). Third, the user must be told at the very least that her job has been prematurely terminated, as opposed to the computation just disappearing. The exact mechanism of doing this is not clear nor are the security implications.

3.4 Accessing Grid Information Services

3.4.1 Read/query information from one information server

The ability to locate services and to determine the status and otherwise availability of those services will be crucial in a well-functioning Computational Grid. In most Computational Grid architectures, there exist *Information Services* whose purpose is to be a centralized repository for information about the many services in the Grid. To avoid being a single point of failure, *Information Services* can be replicated, organized hierarchically, or organized geographically. However, high availability does not mean easy and uniformly accessible—many services require carefully controlled access to information regarding the services they provide, their current status, and who can use them. In general, when a Grid user queries a single information server:

a) Mutual authentication should take place between the user and the information services.

b) The information services should implement the access control policy as desired by the service.

While the information services require the user to authenticate, it is not strictly necessary for information services to authenticate to the user, for example, if the user subsequently authenticates to the service itself. The extra cost of mutual authentication in general can be weighed against the potential effects of malicious information. With regard to the information services providing the actual information requested, it could be the case that the individual services are allowing the information services to determine an "appropriate" access policy. However, a more general scenario is to allow each publisher to set the policy. In this case, the publisher and the information services must agree on a policy language. Subsequently, the publisher must trust that the information services accurately implements the policy.

3.4.2 Publish/upload information to an information service

Scenario 3.4.1 partially describes the implications of publishing information into a centralized repository from the perspective of the publisher. In order for a service to upload the information that the information services is providing to others:

a) Mutual authentication must take place between the publisher and the information services.

b) Confidentiality or message integrity on the communication from the publisher to the information services could be required by the publisher.

If there are no constraints on the information being provided by the publisher, then neither (a) nor (b) are necessary. However, in most cases, the publisher cares about who sees the information, making mutual authentication and confidentiality and/or message integrity important.

3.4.3 Query across multiple information services

The case where a user wishes to interact with multiple information services, receives non-contradictory information, and combines the information himself is a relatively straightforward extension to Scenario 3.4.1. However, when the information services return information that is not consistent with each other:

a) If there is no obvious reason to believe one piece of information over another, the determination of which information to believe must be based on a trust relationship established with one of the information services.

Anytime an information service is accessed, the user must trust the information being returned to a certain extent. It is important to note that authentication does not directly address trust, as authentication merely ascertains that a particular entity is who it claims to be, not that the entity is "doing the right thing". In the case where one information service contradicts another, the user needs to have his own policy for establishing trust which could be based on who wrote each service, who deployed each service, where each service is executing, how useful the information has been in the past from each of the information services, etc. This policy is set by the user and cannot be mandated by the Grid Administrator(s).

3.5 Setting or Querying Security Parameters

3.5.1 User sets message integrity and confidentiality parameters

An individual Grid user should have the capability to constrain the manner in which she interacts with the collective Grid services. One way in which to personalize a user's interaction with a Grid is for the user to define message integrity and confidentiality parameters. For example, a user can state that all communication between Grid services as a direct or indirect result of the user be encrypted stronger than a selected minimum amount (e.g., encryption algorithm and key size). The implications of this requirement are:

a) Message integrity implies supporting MAC algorithms.

b) Confidentiality requires a key agreement protocol to be supported.

c) Services must recognize the rationale for per-user security configuration and be designed accordingly.

d) There must exist a easy mechanism for users to specify such constraints.

e) There must be a secure and efficient mechanism to propagate or otherwise convey a particular user's integrity and confidentiality parameters from the user to the services.

This scenario exemplifies one of the key challenges in constructing a Grid—namely that there is a tension between support for heterogeneity and a requirement that services implement some subset of shared functionality. Many users will implement and deploy services for a Grid, each with a different API and different functionality. However, their utility will be significantly impeded if they mandate how users are to interact with them, as opposed to how the users would like to interact with them. Requirements for message integrity or confidentiality is an example of a requirement that may be imposed across a class of applications from their perspective users.

3.5.2 Resource provider sets message integrity and confidentiality parameters

Resource providers can also set security requirements on a per-resource or per-site basis. An example requirement is that all traffic into and out of a site be encrypted with a particular algorithm. The implications of this requirement are:

a) Services must be aware of and adhere to the message integrity and confidentiality rules of the resource provider on which they execute.

b) Users must be made aware of and adhere to the message integrity and confidentiality rules of the resource provider on which the services that they invoke execute.

Adhering to resource providers' requirements in general is difficult, if only because resource providers generally will not publish security requirements. Many resource providers believe in "security through obscurity" and thus will not publish such information. Other resource providers do not have a single clear document that details such security requirements, but rather the policy has been formed *ad hoc* in response to individual events.

3.5.3 Querying individual's access to a resource

There are several scenarios where one principal wants to know what his own or another's access rights are with respect to a resource.

a) A user may want to determine his access to a resource before attempting to use or schedule use of the resource.
b) A stakeholder may want to see what access a user has.
c) A Super Scheduler may need to see what machines and resources a user has access to.

In each case either the resource gateway or an independent policy analyzer must be able to determine a user's access given the Grid ID of the user and decide if the principal asking the question has the right to see the answer.

3.5.4 Stakeholders set authorization policy

A stakeholder for a resource on a remote machine may want to set or modify the policy for the use of a resource. He may want to see what the existing policy is before modifying it. In this scenario, it is assumed that there is an authentication policy interpreter separate from the resource itself. To determine if a request is authorized, the resource gives the identity of the authenticated user and the authorization policy to the separate policy interpreter and is returned a yes/no/maybe answer. The implications of this scenario are:

a) For policy information stored on the resource gateway, the stakeholder must be able to securely connect to the gateway machine (and subsequently hand-edit a policy file) or authenticate himself to a server on the gateway machine that can modify the policy information.

b) In the case that the server maintains the authorization policy, the server must be able to check that the stakeholder is authorized to change the information.

c) In the case that the server maintains the authorization policy, the server can require message integrity or confidentiality when it reads the policy

d) If the policy information can be stored locally to the stakeholder, the authorization policy must be kept securely.

e) Policy information may need a validity period or a priority assigned to it if the policy is intended to be temporary.

A challenge in supporting this scenario is that that may be multiple stakeholders that have jurisdiction over different usage rights of a single resource. Therefore, the server that maintains the policy must carefully enforce the policy regarding each stakeholder's ability to change the access policy.

3.5.5 Stakeholder wants to revoke access in a timely fashion

A stakeholder may want to deny access to a user or set of users and have it take effect promptly. While the semantics of “timely” or “promptly” vary from case to case, the general scenario is that an entity that was previously allowed access should not be denied access. The implications of this requirement are:

a) Any caching of access rights must be short-lived and/or provide a way of being flushed.

b) If policy information is stored in distributed places or multiple copies are kept, it must be linked together or indexed in some way so that all the copies can be deleted.

c) If capabilities are used they must be very short-lived or else kept in known places from which they can be removed.

There are similar issues that a Certificate Authority must address when a certificate must be revoked.

3.5.6 User requires confidentiality on stored data

In an extension to Scenario 3.5.1, a user may want to specify that certain files be encrypted or all the data at a given site be encrypted. The user may also wish to specifically mandate that a server that acts on her behalf store all data related to her encrypted. This scenario implies:

a) There may exist a need to share an encryption key with the program or server that is writing the file to storage.

b) The user and/or server will need long-term storage and escrow of encryption keys.

c) A secure system is required to associate keys with particular files.

In general, proper key management is a requirement for many of the scenarios. For example, certain administrative domains within a Grid may require *smart cards* for key management, as opposed to a password-based authentication scheme. The requirements for key management must be properly conveyed to the users by the Grid Administrators. Managing keys and understanding each of the individual key management requirements will be a challenge for the user, as a Grid may cross multiple administrative domains.

3.5.7 User, service provider, or administrator specifies trusted Grid hosts

As part of a session-specific configuration or in a directed scheduling request, a user may want to specify what hosts she is willing to use. If a job is going to use several hosts this information has to be passed along to the scheduler or the job controller. Similarly, a service provider may mandate that requests for service must arrive from a particular subset of hosts, perhaps because the other hosts are not trusted or because of billing purposes (if the service is not free). Lastly, a Grid administrator may specify that no user or service is allowed to interact with users or services from another administrative domain. For example, if NASA trusts DoD, but DoD does not trust NASA, then the DoD Grid Administrator(s) might require DoD users cannot use NASA machines in DoD-related computations. To support the specification of trusted Grid hosts or trusted Grid domains:

a) Grid hosts must be able to authenticate and possibly prove membership in a particular Grid domain. This can be done through host SSL credentials or secure DNS and IPSEC.

b) Servers in this category require a protocol in which both the identity and location/domain from which the request originated are authenticated. Clients must be ready to provide this information.

c) Grid administrators must be able to enforce these requirements.

Implementation of this requirement can be problematic with regard to all entities that could specify a set of trusted Grid hosts. For example, if the computation scenario is such that there is a *chaining* of services (e.g., user asks server-1, server-1 asks server-2, server-2 asks server-3, …, server-*n* returns information back to the user), then the entire chain might be required to be authenticated before server-*n* performs the requested action and subsequently returns the information to the user. Similarly, server-*n-1* must realize the user's restricted set of hosts before contacting server-*n* (if server-*n* is not contained in the list of trusted hosts, then server-*n-1* should not attempt to use server-*n*). This is also not easy for the Grid administrator to enforce.

3.6 Auditing Use of Grid Resources

3.6.1 System administrator wants to check a list of past requests

Ether a site system administrator or a Grid administrator may need to monitor all accesses to a resource. This information may be used for accounting purposes, for a routine security review, or for a real-time intrusion detection procedure. The system administrator may wish to check both the accesses allowed and the accesses rejected. This scenario implies:

a) The resource gateway server must keep an unforgeable log of all access, by unique user identification and time of access.

b) The format of the entries to this log must be negotiated between the system administrator and the resource gateway.

c) Access to this log should be carefully restricted.

d) The system administrator ask the resource gateway server to signal especially troublesome resource access requests via a mechanism separate from this log.

The system administrator and the resource gateway server must agree upon both the content and format of logged entries. Arbitrary servers running on a Grid host can also be required to do similar logging. How useful such a log file is will depend on how trusted the server is. Presumably it is trusted to some extent or else the local system administrator would not allow it to run at all, but by definition it is not a standard part of the Grid services.

3.6.2 Stakeholder wants to check who has been accessing resource

Whereas in Scenario 3.6.1 the system administrator can require access to all access logs for services on the system administrator's machines, a stakeholder should be able to only access the logs for which he has authorization. A stakeholder may want the same information as in Scenario 3.6.1 but only for resources for which he is the stakeholder. To meet this requirement:

a) Stakeholders should have limited access to the access logs.
b) There is a need to identify a stakeholder with a resource.

A stakeholder may wish to review the resource logs to determine the overall usefulness of the service or to determine whether the service is oversubscribed.

4 Acknowledgments

We are grateful to the many members of the Security working group who contributed to the discussions regarding this topic at Grid Forum 4 in Seattle in July 2000. In particular, Steve Tuecke greatly aided the organization of these topics. Many members of the Scheduling working group also contributed to the development of these ideas. In particular, Keith Jackson read versions of this draft and enhanced its development.

This work was supported in part by the National Science Foundation grant EIA-9974968, DoD/Logicon contract 979103 (DAHC94-96-C-0008), and by the NASA Information Power Grid program.

5 References

[1] Foster, Ian, and Carl Kesselman. "Globus: a metacomputing infrastructure toolkit". *International Journal of Supercomputer Applications*, 11(2):115-128, 1997.

[2] Grimshaw, Andrew S. and William A. Wulf. "The Legion Vision of a Worldwide Virtual Machine." *Communications of the ACM*, 40(1):39-45, January 1997.

[3] Neuman, B. Clifford and Theodore Ts'o. "Kerberos: An authentication service for computer networks." *IEEE Communications Magazine*, 32(9):33-38, September 1994.

6 Security Considerations

The entirety of this document contains security considerations.

7 Authors' Addresses

Marty Humphrey
Department of Computer Science
School of Engineering & Applied Science
University of Virginia

151 Engineer's Way, P.O. Box 400740
Charlottesville, VA 22904-4740
Phone: 804-982-2258
Email humphrey@cs.virginia.edu

Mary Thompson
Distributed Security Research Group
Lawrence Berkeley National Lab
1 Cyclotron Road; Mailstop: 50B-2239
Berkeley, CA 94720
Phone: (510) 486-7408
Email: mrthompson@lbl.gov