Security for Grids

Marty Humphrey, Mary R. Thompson and Keith R. Jackson

***Abstract* — Securing a Grid environment presents a distinctive set of challenges. This paper groups the activities that need to be secured into four categories: naming and authentication; secure communication; trust, policy, and authorization; and enforcement of access control. It examines the current state of the art in securing these processes and introduces new technologies that promise to meet the security requirements of Grids more completely.**

***Index Terms* — Grid security, authentication, authorization, trust management, secure communication, security policy**

I. Introduction

The goal of Grid Computing is to create a "virtual organization" across one or more physical organizations (or "administrative domains"). The resources of these virtual organizations range in scale from a small number of large clusters (such as the TeraGrid [1]) to millions of PC-class machines (such as SETI@Home [2]). The resulting value of the virtual organization to users in each of the physical organizations is that the users can be more productive, either in their own activities or in their collaborations with other people across the virtual organization. This enhanced productivity is achieved by having access to a greater number or variety of resources (such as computers, databases, or specialized equipment) or by having easy-to-use mechanisms by which to collaborate with others outside of the user's home enterprise.

These virtual organizations require common solutions for resource management, data management and access, application development environments, and information services. In many ways, arguably, the most significant challenge for Grid computing is to develop a comprehensive set of mechanisms and policies for *securing* the Grid. Users need to know if they are interacting with the "right" piece of software or human, and that their messages will not be modified or stolen as they traverse the virtual organization (if the users have such a requirement). Users will often require the ability to prevent others from reading data that they have stored in the virtual organization. In short, users must *trust* the software infrastructure of the Grid to sufficiently prevent malicious activities (of course, it is reasonable to expect that the user must be proactive in this regard and that this does not come without some burden on part of the individual user).

One of the critical differences between Grid security and host or site security regard *site autonomy*. If a system administrator is faced with the challenge of securing a host or site, the system administrator generally establishes a logical barrier around the site or host and analyzes the ways in which access to that host or site can be gained through the barrier. For each mechanism, simplistically, if the reward is greater than the risk, then the mechanism is allowed to exist. For example, many default services on a Linux machine will be disabled as a result of this analysis to prevent attackers from potentially gaining access to the machine. In contrast to this situation in which the system administrator has complete control to modify both the security mechanisms and security policies, resource providers for the virtual organization of the Grid must understand and accommodate mechanisms and policies that are not strictly under their control. For example, one site might allow password-based logins while another might require Kerberos-based logins [3]. While the virtual organization certainly could define some common security mechanisms and policies across the entire virtual organization, it is more often the case that the Grid spanning the two sites must accommodate both approaches (in other words, the physical organization takes precedence over the virtual organization). In addition, a site that provides resources to a Grid might now have to consider opening some access points that were closed in the process of securing the host or site so that the resource can be utilized by non-local members of the virtual organization.

While acknowledging and respecting this fundamental need for site autonomy, there are a number of requirements for Grid security in order to achieve the goals of the virtual organization. Users need globally defined names that will be recognized at all sites to which they have access. A user's identity needs to be passed securely and transparently between sites as a job progresses. Grid sites need to enter into trust relations with Grid users and with the other Grid resource sites. Users need a simple, consistent way to get authorization to use the Grid resources, e.g., same mechanisms across multiple sites.

This paper presents the state of the art with regard to Grid

Manuscript received April 5, 2004. This work was supported in part by the National Science Foundation under grants ACI-0203960, ANI-0222571, and the National Partnership for Advanced Computational Infrastructure (NPACI); and U.S. Department of Energy, Office of Science, Office of Advanced Science, Mathematical, Information and Computation Sciences under contract DE-AC03-76SF00098. LBNL Report Number ????

M. Humphrey is with the Department of Computer Science of the University of Virginia, Charlottesville, VA (e-mail: humphrey@cs.virginia.edu).

M. R. Thompson is with Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA (e-mail: mrthompson@lbl.gov)

K. R. Jackson is with Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA (e-mail: krjackson@lbl.gov)

security and identifies open issues. After introducing terminology (Section II), this paper groups the techniques for securing the Grid into five categories: naming and authentication, communication, trust management, authorization and enforcement. The goal of this paper (and our earlier work on the security implications of basic Grid usage patterns [4]) is to enable Grid users and Grid resource providers to understand the best practices with regard to Grid security and to understand the difficult challenges that to date remain unsolved.

II. Terminology and Background

There are a number of basic definitions regarding computer security (for an excellent treatment of computer security in general see [5], the RSA FAQ [6], and Peter Gutmann's on-line tutorial [7]). *Authentication* is the act of ensuring that someone or something is whom they claim to be. *Authorization* is the right to perform some action (such as reading a particular database). *Integrity* refers to the ability of the computer system to ensure that the data is protected from unauthorized modifications. *Confidentiality* is the ability to keep information from being disclosed to unauthorized users. *Privacy* refers to the ability to keep some information solely to oneself. *Availability* refers to the ability of authorized parties to obtain access to the information when it is needed. *Non-repudiation* refers to the inability of something that performed a particular action such as a financial transaction to later deny that they were indeed responsible for the event. *Trust* can be defined as the assured reliance on the character, ability, strength, or truth of someone or something.

There are four general categories of attacks on security services. *Interruption* occurs when a message is blocked to/from a particular service. *Interception* refers to an intruder catching but not necessarily blocking a message intended for a recipient. *Modification* refers to the action of intercepting, modifying, and then re-transmitting a message to a security service. *Fabrication* refers to generating a new message from scratch and attempting to insert it into the normal message flow.

There are a number of cryptographic mechanisms to prevent the four categories of attacks on security services. These mechanisms underlie the Grid-specific technologies that are the subject of the remainder of this paper

Integrity. Integrity checks are provided primarily via hash functions (or "message digests"). For example, software developers often make their software available on their web sites and also state the MD5 [8] value of the distribution file. To verify integrity, once the distribution file is downloaded onto the local machine, the MD5 algorithm is performed on the local file. If the MD5 value computed locally is the same that is published on the web site, then the distribution file on the local machine is the same as the one in which the author performed the MD5. Note that the correctness of this approach is predicated on the security of the site from which the distribution file was downloaded (an attacker could have maliciously replaced both the distribution file and the MD5 value), and that it is computationally infeasible to produce two distribution files with the same MD5 values. If the integrity of the download site is questionable the MD5 hash can be signed, and the public key needed to verify the signature provided from another more secure site.

Confidentiality. The standard approach to ensure confidentiality is through *encryption*, which is the application of an algorithm that transforms "plaintext" to "ciphertext" whose meaning is hidden but can be restored to the original plaintext by another algorithm (the invocation of which is called *decryption*). Secret algorithms, which by definition are intended to be known only by the parties involved, are not generally used in the commercial or scientific sectors because they are not subject to public scrutiny and thus believed to be inherently weaker. Public algorithms can be symmetric or asymmetric; "symmetric" means that the same key is used to both encrypt and decrypt the message (both the sender and the receiver have a copy of this key). Symmetric cryptography is also known as secret-key cryptography. In contrast, asymmetric cryptography is often referred to as public-key cryptography because each participant has two keys, one called the public key and one called the private key (which the owner keeps secret). The two keys are mathematically related. In asymmetric cryptosystems that support encryption/decryption, the sender encrypts the message with the receiver's public key and sends the message; the receiver decrypts the message with the receiver's private key. Because only the receiver has the private key, the message is confidential. The most popular symmetric encryption algorithm is the Data Encryption Standard (DES) [10], although the strength of DES has been called into question, and NIST currently recommends using the new Advanced Encryption Standard (AES) algorithm [11]. The most popular public-key cryptosystem is RSA [12] (which supports both encryption/decryption and digital signatures, discussed shortly).

Authentication. In general, authentication is through the presentation of some token that cannot be forged. This can either be in a peer-to-peer relationship—such as a password that only the client and the server know—or through a trusted third party such as a Kerberos server or a Certification Authority (CA). Biometrics can also be used, especially as a mechanism by which a human can acquire a token that is later presented to a service for authentication purposes. For example, a fingerprint scanner can be used to login to a local machine.

Non-repudiation. The digital signature can be used to claim that an entity was uniquely responsible for a message or action. For example, a document is digitally signed usually by taking the hash of the document (such as described in the "Integrity" section, previously), encrypting the hash with the sender's private key, and attaching the signature to the cleartext message. The receiver of the message checks the signature by calculating the hash of the cleartext document and comparing

the hash value with the value purportedly attached to the message itself (note that the receiver first decrypts the value attached to the message with the purported sender's public key). If the two has values match, the receiver has strong evidence to believe that only the purported sender could have sent the message. (Note that the use of a digital signature is for authentication and is orthogonal to the use of public-key cryptography for confidentiality.) The Digital Signature Algorithm (DSA) is a popular public-key algorithm that can only be used for signatures and not encryption. Although digital signatures provide an important building block for non-repudiation, they are not sufficient. A malicious user could conduct a financial transaction with someone; digitally sign the transaction for payment, and then later claim that his private key had been stolen before the transaction occurred. One proposal to mitigate this risk is the use of a trusted Time Stamp Authority [9]. The TSA would attest to the fact that a transaction took place at certain time by attaching a digitally signed time record to the original signature.

Extensive logs can also be used as the basis for claiming that a particular user performed a particular action. As compared to unsigned logs, digital signatures are largely regarded as stronger evidence for non-repudiation because of the implicit belief that access to a user's private key (to impersonate the user) is more difficult than, say, obtaining a password that the user might use to gain access to some service or gaining access to tamper with the logs

Authorization. There are two general approaches for authorization: identity-based or token-based. Identity-based approaches are typically associated with access control lists, while token-based approaches are also referred to as capability-based authorization. In identity-based approaches, only the authenticated user identity is presented to the resource, which then checks an internal list of allowed identity/action pairs. In token-based approaches, an unforgeable token is granted to the user, who then presents it to the service as proof of her rights. In some sense, the service does not care *who* the presenter is, rather just that the request came with the appropriate token. A drawback of identity-based approaches is that identity-based approaches cannot easily support *delegation*, in which one person allows/requests another user or software agent to act on the user's behalf. On the other hand, a drawback of a token-based approach is that it may be very difficult to dynamically revoke access rights.

III. Naming and Authentication

Authentication is proving your identity to some person or some agent running on a computer. Naming is the assignment of some identifier to a unique person or other long-lived entity so that it can be used for authorization and auditing. Within a single Grid, it is most convenient to have a globally unique name for each user or entity. Since human names are not unique, and machine or resource names are only unique within the domain in which they are defined, defining a global Grid identifier takes some thought.

A. Naming Approaches

There are several approaches to the global naming problem. One is to generate a random number from a large sparse domain space and count on the generation algorithm to ensure its probable uniqueness. The advantage of this approach is that names can be assigned by different agents with no co-ordination and still be globally unique. The disadvantage is that the identifiers tend to be long and provide no clue as to the real world entity to which they belong. PGP public keys are an example of such identifiers [13].

Another approach is to start with a meaningful name, such as a person's legal name and then add more identifying components until it is "globally" unique. For example, John son of Thomas of the village of Surrey, or to be more up-to-date a DNS name of the form host.site.domain The X.500 [15] naming structure is an attempt to define enough components that people can be named uniquely and meaningfully. The various sites doing the naming must co-operate to the extent that each one has a uniquely assigned component name or names that it uses as part of all the names it creates. This in turn requires some sort of global registry of such names. X.509 names [16][17], derived from the X.500 standard, are commonly used by Grid software to provide global names for users and hosts.

A third approach is to maintain that all names have meaning only in a local context. One manifestation of this model is SDSI (Simple Distributed Security Interface) [18], which identifies users by a public key and allows the users (relying parties) of these keys to assign a local name that has meaning in their domain. These names are assigned manually by a site administrator after reviewing the evidence presented by the owner of the public/private key pair as to who he is. In an attempt to allow scaling, name spaces can be linked, so that one site can use another site's defined names.

Identity servers such as used in Shibboleth [19], Microsoft's Passport server [20] and the Liberty Alliance [21][22] map a user name and attributes onto a unique identifying handle. These servers are designed to provide a single signon to a set of loosely connected resource providers such as in the Web Services model [23]. A major challenge of an identity server is to be able to store all the attributes about an individual that may be needed by any of the service providers that the users wishes to contact, but at the same time to protect the privacy of the user. To respond to users' privacy concerns, these servers may restrict the information that can be discovered about the holder of the handle depending on who is asking for information. For example, Shibboleth does not release the user's name except to authorized requestors. These servers support an authorization model that allows access based on attributes such as group membership or role in an organization, rather than identity.

B. Techniques for Authentication

Authentication in a computer environment is the process of associating a real-world identity with a request to a machine. An underlying assumption of authentication is that a unique

machine-readable unique-id or global name will always be assigned to the same unique real world person or computer. Authentication takes place when the connecting entity provides such a unique-id and the authenticating agent can verify that the id legitimately represents the connecting entity.

Authentication to a single host is most commonly done by a login process that uses a password shared by the user and the host. In many systems the host stores only a one-way hash of the password to increase the difficulty of a third party stealing it. When the login in performed over the WAN, the clear text password is very vulnerable to theft in transit from the user's keyboard to the target machine. Also if a user needs to login on many hosts, either many passwords are required or a single password in stored in many places, increasing the vulnerability of one system to a security breach in another system.

Kerberos [3] is an authentication system designed to allow a single sign-on to many machines within an administrative domain. Only the Kerberos Key Distribution Center (*KDC*) needs to know everyone's password. It acts as a trusted third party between the user and a target host. Both the user and the target host establish an authenticated secure connection with the KDC. The KDC gives the user an encrypted token that he can present to a target server, who can then present it to the KDC to establish the identity of the requestor. In a Kerberos login the user does not send the password to the KDC, but uses it to encrypt a challenge phrase that Kerberos can decrypt using its copy of the password. Since the KDC stores all passwords, it becomes a single point of failure for the whole domain. Even though the KDC is expected to be operated in a very secure manner, the hosts that rely on a KDC tend to be limited to a single administrative domain.

A more loosely coupled approach to authentication across many domains is to use public key infrastructure (PKI) technology. The target host that is going to authenticate a user has a verified copy of the user's public key. Authentication is performed by the user encrypting a challenge phrase with her private key, which the target host can decrypt with the user's public key. Thus proving her possession of the private key associated with the public key. (See Section IV on Secure Communication for more details on this protocol.)

The most common public key authentication protocol in use in Grids today is the Transport Layer Security (TLS) [24] protocol, that was derived from the Secure Sockets Layer (SSL) v3 [25] protocol. TLS uses an X.509 public key certificate [16][17], which binds a multi-component meaningful name, called a *Distinguished Name (DN),* to a public key. The binding is attested to by a *Certification Authority (CA),* who performs specified actions to ensure that the name represents the individual who knows the private key associated with the public key. When a X.509 certificate is presented as an authentication token, the server challenges the connecting entity using the *TLS* handshake protocol to prove its knowledge of the private key associated with the public key in the certificate. Once the remote entity has been verified, the authentication agent needs to secure the communication channel so that no one but the authenticated entity can use it. At this point there may be a local user id and/or running job associated with the channel. From then on, all actions performed as a result of requests coming from that channel are assumed to be done by the entity named by the DN in the certificate.

There are other standard protocols such as PGP [13] and SPKI [14] that can be used to create a secure and authenticated connection using only a public key to identify the requestor. The target host may have a mapping of the public key to a local user name for the connecting entity. The major difference between these methods and TLS is that in the former the public key of the user must be known by the target host by some secure method, and if a local name is needed it must be assigned to the private key by some out-of-band procedure. In the case of X.509 certificates, both the public key and the distinguished name are included in the certificate. The only information that the target host needs to authenticate the user is the public key of the CA that issued the X.509 certificate. Therefore, the key distribution problem is reduced to distributing the small set of CA public keys rather than user keys and names.

In any public key scheme there must be a way to revoke a public key, if the private key is known to be compromised. In the X.509 case it is the responsibility of the CA that issued the certificate to revoke the certificate and provide some way for the relying parties to get this information. One method is to publish a certificate revocation list (CRL). Another more recent method is to use the online certificate status protocol (OCSP) [26] to query the validity of the certificate. In the PGP [13] and SPKI [14] schemes the holder of the private has the responsibility of providing revocation information to any relying party that may have its public key stored.

There have been a number of projects that allow one credential to be used as the basis for obtaining a different credential. KX.509 [27] is a Kerberized client-side program that acquires an X.509 certificate using a client's existing Kerberos ticket. PKINIT [28] describes how to use public key cryptography to obtain a Kerberos ticket. MyProxy [29][30] is a widely deployed online credential repository for Grids. It is most often used to exchange a password-based credential to obtain a GSI credential, but in the general case, it can be used to exchange one arbitrary credential for another. MyProxy is particularly valuable in that it ameliorates some of the problems associated with a PKI, essentially making a PKI easier to use and manage.

IV. Secure Communication

Secure communication provides the ability for two or more entities to conduct a conversation with *integrity,* and *confidentiality* if required. Doing this requires that the parties to the communication authenticate to each other, and that the corresponding communication channel support integrity checking and possibly confidentiality. As mentioned earlier in the paper, *integrity* is typically provided using standard hash

algorithms, and *confidentiality* is provided using encryption.

A number of different mechanisms exist for securing communication channels. Some go back thousands of years, and others have only been invented in the past 25 years.

The simplest mechanism for securing a communication channel is through the usage of symmetric cryptography. If both parties share a secret, such as a series of random numbers, and a common encryption algorithm to transform the plaintext to ciphertext the secret can then be used to exchange confidential messages. A similar process using a hash algorithm can be used to provide message integrity.

One of the most significant drawbacks to this approach is how to distribute the secret to the appropriate parties without anyone else knowing it. Post 1950, the banking industry developed highly complicated schemes for distributing secrets to banks to allow for secure communication between the financial institutions. Typically, these keys had to be transported in armored cars at great cost to the banks.

The most common way to address this problem today is through the usage of asymmetric cryptography. Instead of distributing a secret, parties who wish to communicate can publish their "public keys" in an accessible place and others can retrieve them for usage in asymmetric cryptography. In the Grid, typically individual keys are not published, but instead the public key of the CA that signed the public keys is published. This becomes a root of trust for identity in the PKI. Any public key signed by the trusted CA is trusted to represent a unique individual or entity.

These keys can be used by protocols such as TLS or IPSec [31] to establish a symmetric session key that is known by both ends of the authenticated connection and used in all subsequent communications. This is mainly done for performance reasons, since symmetric cryptography is typically significantly faster then asymmetric cryptography.

The Grid Security Infrastructure (GSI) is built on top of the TLS protocol. The principles of GSI are described by the Globus designers in [32][33]; similar principles are described in [34] by the designers of Legion [35]. In addition to the standard TLS protocol, GSI provides the ability to delegate an X.509 proxy certificate to the remote entity thereby allowing the entity to perform actions on your behalf. This is required in a Grid where a job may be submitted to a scheduler that needs to access files on your behalf, submit computations, and return results. The other main difference between TLS and GSI is that TLS, while a standard protocol, does not have a standard programming API. GSI uses the IETF GSS API [36] to provide a standard API for GSI programming. While in theory the GSSAPI can be bound to multiple implementations (such as Kerberos), in GSI it is almost always implemented via OpenSSL [37].

So far in discussing secure communication we have assumed nothing about the communicating entities. They could be humans or computers. They could have a direct link between them, or their messages could be routed through other intermediaries.

IPSec [31] is often used for computer-to-computer security. It provides authentication and encryption at the IP layer. Key exchange can either be done manually, or through the Internet Key Exchange (IKE) [38] protocol. Authentication is done through the usage of Authentication Headers (AH) that contain cryptographic checksums on the IP packets. After authenticating a Session Security Association (SA) is created. This SA is referenced in each IP packet header to track the security parameters, i.e., what encryption algorithm is being used. The Encapsulating Security Protocol (ESP) encrypts each IP packet.

Both TLS and GSI operate at the transport layer. They require an ordered reliable transport connection, so typically they are implemented over TCP. While this has been sufficient for most current Grid usages, recent moves towards using Web Service [23] based technologies on the Grid make this more problematic. In particular, the SOAP Protocol [39] is being used by the emerging Open Grid Services Architecture (OGSA) [40][41], which is being developed in the Global Grid Forum [42]. Within OGSA, recently, the rendering of Grid Services has moved from the Open Grid Services Infrastructure (OGSI) [43] to WS-Resource Framework [44]; however the reliance on SOAP remains. SOAP messages can be routed independently of the underlying transport connection. This is very useful to allow intermediaries to act on the messages before passing them onto the recipient. It can be used for load balancing, firewall examination, etc. Although there may not be a direct TCP connection between the two end points that are exchanging messages they may still wish to authenticate each other and communicate securely.

This requirement has lead to the development of message-based security. By applying the cryptographic techniques described earlier at the message layer, end-to-end message security can be achieved. Hence, the last few years have seen the development of several key standards to support message layer security. The XML Digital Signature standard [45] describes how to digitally sign arbitrary XML messages, and the XML Encryption standard [46] describes similar functionality for encrypting messages. Each of these standards has particular bindings for how it should be used with SOAP messages. Enhancements to SOAP messaging to provide message integrity and confidentiality are being standardized in OASIS [47] at the time of this writing (WS-Security 2004 [48]). In addition to these standards, recent work has begun on moving the authentication protocol to the message layer. The WS-Secure Conversation specification [49] describes how two entities can authenticate each other at the message layer. The most recent versions of Grid middleware such as Globus Toolkit™ version 3 [50], pyGridWare [51], and OGSI.NET/WSRF.NET [52][53] are using a version of WS-Secure Conversation that has basically moved the TLS authentication handshake up into the SOAP message layer.

V. Trust, Policy, and Authorization

Trust can be generally defined as having confidence that a

party will behave in an expected manner despite the lack of ability to monitor or control that other party. Normally trust is positive, predicting a good outcome in uncertain circumstances. Trust management is the process of deciding what entities are to be trusted to do what actions. In an environment where access policy is described in terms of the identity of users or required attributes, trust management consists of defining the sources of authorities for user identification, attribute assignment and possibly policy creation. In a system where users are granted authorization tokens, the entire authorization system has been called trust management [54]. In a system where users can delegate some or all of their rights to others users the control of such delegation is part of trust management.

In the context of this section, we define policy from the second perspective stated by the GGF Grid Policy research group [55] as fundamentally a set of principals or rules that regulate the behavior of a system. External representations of policy are highly desirable, in order to achieve flexibility, transparency and scalability of a system. Policy related to security may regulate trust, including delegation of trust, authentication, authorization, and levels of message or data integrity and confidentiality.

Authorization in the context of this section, is the determination of who perform what actions. This is determined as a result of evaluating the request of an authenticated user against the trust and authorization policy association with a resource domain.

In traditional single site environments, trust management is usually handled outside of the authentication and authorization systems. Authentication and access information is trusted because it is found in a trusted site, e.g. /etc/password or UNIX ACLs, or comes from a trusted server, e.g. Kerberos or NIS [56]. The control on who can modify this information is part of the system configuration and usually is restricted to a set of privileged users. When permitted users and access to resources need to be specified by off-site administrators this approach does not work well. Usually system privileged users have far greater privileges than would be needed just to authorize access to a single set of resources making site administrators reluctant to grant such privileges to users outside their local domain. Resource systems such as AFS [57] and LDAP [58] that are intended to be independent of specific hosts support a privileged user whose privileges can be limited to a subset of the resources. The granting of access to such users is handled separately from the granting of access to the protected resources. Thus, we can think of trust management as a separate process to authorization.

The TLS protocol enforces a simple trust management scheme in that only users whose X.509 certificates are signed by the known CAs are allowed to connect via the TLS protocol. GSI expands on this idea a bit, by adding a CA signing policy file for each CA that specifies the namespace in which it may issue names. In the GSI the only Grid level authorization rules are in the Grid map file, which maps a Grid DN to a local user account. The authority for this information is the file in which it is found, and the authorization to write in that file depends on a local privileged account. This scheme turned out not to scale to the case where remote administrators need to control access to local resources. CAS [59] and VOMS [60] are solutions to this problem. In those cases the trust in the authorization information comes from a secure server that the local site has chosen to trust. In both CAS and VOMS there is an independent trust policy controlling what parties can change the information in the server.

Authorization systems such as Akenti [61] and Permis [62] explicitly state what authorities are allowed to grant access privileges or user attributes. There is ultimately some root source of authority who writes the first policy entry. However, that entry can specify additional authorities and in what parts of the resource domain they can issue assertions. In the case of the current implementation of Permis, both trust and authorization are found in a secure place by the policy decision point. Since Akenti supports distributed policy statements, all policy, authorizations and attributes are signed, and the signer must be a recognized authority for whatever it is signing. The trust management is integrated with the resource access policy, as the authority to assert a property is explicitly stated as part of the resource policy. Akenti specifies the CAs that may issue X.509 certificates, the stakeholders that may write policy for specified resources and the authorities that may issues specified attributes for users.

SAML [63][64] and XACML [65][66] are two emerging XML languages for use in authorization queries (SAML) and authorization policy statements (XACML). SAML allows for the explicit signing of assertions so that when attributes are used in requesting access they can be strongly bound to an authority. XACML supports explicit trust management in a limited way. It allows a policy to specify a single issuer for an attribute; however, the signing of policy is outside the scope of the XACML schema. It just assumes that the policy will be found in a secure manner.

The proposed Grid authorization standard [67] advocates SAML authorization queries and responses as a standard interface to an OGSA authorization service, but does not specify a particular policy representation or a language for trust management. If all policy is kept in a central place and/or provided by a trusted server, then a simple trust management policy that assumes a secure URL for the server will suffice. However, a more distributed scheme requires explicit trust management to allow it to scale for multiple resource sites, multiple stakeholders, attribute authorities and CAs.

More broadly, OGSA plans to exploit both academic research as well as commercial standards for policies (not necessarily specifically related to security) on Grid services. The WS-Agreement specification proposes support for service management, which is “the ability to create Grid services and adjust their policies and behaviors based on organizational goals and application requirements” [68]. This is a form of Service-Level Agreement (SLA) for OGSA. WS-Security

Policy [69] fits into the WS-Security framework [70] and provides the ability of Web Services clients and services to express preferences and requirements on such actions that can utilize the WS-Security specification (such as "The service recognizes password authentication but would prefer a Kerberos token.") Preconditions and obligation in policy provide a compact means of representing what a user must do *before* performing a specific action as well as what they must do *after* an action (such as Ponder [71] and [72]). Research in explicit policy management in Grids is starting to appear in Grid workshops (e.g., [73][74][75]).

VI. Enforcement of Access Control

In many ways, the enforcement of access control is the most important and the most challenging part. The local resource is ultimately responsible for enforcement of any authorization decisions made by the authorization system. As we have seen above there are many different approaches to Trust Management, but typically these systems stop at the point of defining their interactions with the enforcement point. This point is sometimes referred to as a "Access Control Enforcement Point" (AEF) [76] or a "Policy Enforcement Point" (PEP) [77]. Both the IETF/DMTF and the OASIS communities have proposed standards for exchanging messages between the authorization system and the decision point, but this still leaves the ultimate enforcement up to the local resource. If mechanisms are not in place on the local resource to enforce compliance with policy, this can be the weakest link in the authorization chain. Modern operating systems typically do not have mechanisms in place for enforcing policy at the kernel level. A machine that has been "hacked" typically has code installed on it to circumvent all local enforcement functionality for the hacker. It also circumvents any local auditing or logging enabled on the local resource. Thus, it can appear that policy is being enforced properly while in fact this is not true.

The first line of defense against misuse of a site's resources is often a site firewall. Such firewalls are implemented on secure hosts, solely controlled by the site administrators and implement simple site-wide access rules, for example, allowing access only to specified ports and specified protocols. Such a firewall is a serious obstacle to Grid computing, which uses a wide and sometimes unpredictable range of ports and new protocols. Additionally, firewalls are often administered by site security personnel who may not be familiar with the specific resources that are to be shared or the secure Grid protocols that are being used. However, firewalls are beginning to be seen as an opportunity to enforce Grid security policy by creating Grid or application aware firewalls. [78][79]. Work has also been started on adapting the BRO adaptive intrusion detection system [80] to be aware of Grid protocols.

Ultimately better mechanisms are required at the local resource level to enforce authorization decisions. This includes support for policy enforcement at the lowest level of the system. It also includes mechanisms for creating secure audit trails that can be used to prove that the authorization policy was enforced correctly by the local resource.

VII. Conclusion

Security is one of the most challenging aspects of Grid Computing. To date, the Grid security community has been very successful by methodically evaluating approaches and mechanisms that have been previously developed for broader computing (such as underlying cryptography) and adapting or further developing these approaches to satisfy the unique, cross-domain requirements of the Grid. Today, the Grid security community faces a critical opportunity in that the Grid, which has traditionally been the concern only of academic and national lab communities around the world, is converging with the commercial sector through the Open Grid Services Architecture and Web Services. Well-founded approaches for authentication and message security will be synergized with new, emerging approaches for explicit trust and policy management. For example, the "Web Services Roadmap" of IBM and Microsoft [81] is in part strongly influencing the efforts of the OGSA Sec working group of the Global Grid Forum [82][83]. In addition, significant challenges remain for such topics as privacy management, denial of service, and integrated, cross-domain auditing. Grid security and computer security in general will never be completely "solved", but undoubtedly these new capabilities will ensure a more trustworthy Grid infrastructure.

References

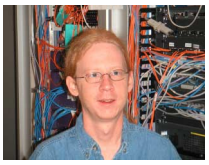
[1] National Science Foundation TeraGrid. Available: http://www.teragrid.org
[2] SETI@Home: The Search for Extraterrestrial Intelligence. Available: http://setiathome.ssl.berkeley.edu/
[3] B. Clifford Neuman and Theodore Ts'o. Kerberos: An Authentication Service for Computer Networks, *IEEE Communications*, 32(9):33-38. September 1994.
[4] M. Humphrey and M. Thompson. Security Implications of Typical Grid Computing Usage Scenarios. In *Proceedings of the 10th International Symposium on High Performance Distributed Computing (HPDC)*, San Francisco, California, August 7-9, 2001.
[5] C. Landwehr. Computer Security. International Journal of Information Security. Spring-Verlag. Vo1, Number 1. Aug 2001. pp. 3-13.
[6] RSA Laboratories' Frequently Asked Questions about Today's Cryptography, Version 4.1. Available: http://www.rsasecurity.com/rsalabs/faq/
[7] P. Gutmann. Computer Security tutorial. http://www.cs.auckland.ac.nz/~pgut001/tutorial/
[8] R. Rivest. The MD5 Message-Digest Algorithm. RFC 1321. Available: http://www.faqs.org/rfcs/rfc1321.html
[9] C. Adams, P. Cain, D. Pinkas, and R. Zuccherato. Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP). RFC 3161. August 2001. Available : http://www.ietf.org/rfc/rfc3161.txt
[10] ANSI X3.106, "American National Standard for Information Systems-Data Link Encryption," American National Standards Institute, 1983.
[11] Federal Information Processing Standards Publication 197. Novermber 2001. Available: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
[12] R. Rivest, A. Shamir, and L. M. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM, v. 21, n. 2, Feb 1978, pp. 120-126.
[13] MIT Distribution Center for PGP (Pretty Good Privacy). http://web.mit.edu/network/pgp.html

[14] C. Ellison, et.al SPKI Certificate Theory, IETF RFC 2693, available from http://www.ietf.org/rfc/rfc2693.txt
[15] CCITT X.500 Series (1994) | ISO/IEC 9594,1-9:1994, *Information Technology - Open Systems Interconnection - The Directory*
[16] ITU-T Recommendation X.509 (1997 E): Information Technology - Open Systems Interconnection - The Directory: Authentication Framework, June 1997.
[17] R. Housley, W. Polk, W. Ford, and D. Solo., "Internet X.509 Public Key Infrastructure: Certificate and CRL Profile", RFC 3280, April 2002.
[18] R. Rivest and B. Lampson. "SDSI – A Simple Distributed Security Infrastructure". Available: http://theory.lcs.mit.edu/~rivest/sdsi10.html
[19] Shibboleth. http://shibboleth.internet2.edu/
[20] Passport, http://www.passport.net/Consumer/Default.asp?lc=1033
[21] Liberty Alliance Project. Introduction to the Liberty Alliance Identity Architecture. Revision 1.0. March, 2003. http://www.projectliberty.org
[22] Linn, John, editor. Liberty Trust Models Guidelines. Version 1.0. Liberty Alliance Project. http://www.projectliberty.org/specs/liberty-trust-models-guidelines-v1.0.pdf
[23] Web Services, http://www.w3.org/2002/ws/ or http://www.webservices.org/index.php/article/archive/61
[24] T. Dierks and E. Rescorla. The TLS Protocol Version 1.1. RFC 2246. March 2004. Available: http://www.ietf.org/internet-drafts/draft-ietf-tls-rfc2246-bis-06.txt
[25] A. Frier, P. Karlton, and P. Kocher, "The SSL 3.0 Protocol", Netscape Communications Corp., Nov 18, 1996.
[26] M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol (OCSP). RFC 2560. June 1999. Available: http://www.ietf.org/rfc/rfc2560.txt
[27] Olga Kornievskaia, Peter Honeyman, Bill Doster, and Kevin Coffman, "Kerberized Credential Translation: A Solution to Web Access Control," USENIX Security Symposium, Washington, D.C. (August 2001).
[28] B. Tung, C. Neuman, M. Hur, A. Medvinsky, S. Medvinski, J. Wray, and J. Trostle. Public Key Cryptography for Initial Authentication in Kerberos. RFC 1510bis. Aug 20, 2004. Available: http://www.ietf.org/internet-drafts/draft-ietf-cat-kerberos-pk-init-18.txt
[29] J. Novotny, S. Tuecke, and V. Welch. An Online Credential Repository for the Grid: MyProxy. Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10), IEEE Press, August 2001.
[30] MyProxy Online Credential Repository. http://grid.ncsa.uiuc.edu/myproxy/
[31] S. Kent, R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401. November 1998. Available: http://www.ietf.org/rfc/rfc2401.txt
[32] I. Foster, C. Kesselman, G. Tsudik, S. Tuecke. A Security Architecture for Computational Grids. *Proc. 5th ACM Conference on Computer and Communications Security Conference*, pp. 83-92, 1998.
[33] R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, V. Welch. A National-Scale Authentication Infrastructure. *IEEE Computer*, 33(12):60-66, 2000.
[34] A. Ferrari, F. Knabe, M. Humphrey, S. Chapin, and A. Grimshaw. A Flexible Security System for Metacomputing Environments. *Proc. High Performance Computing and Networking Europe 1999*, Amsterdam, April 1999.
[35] A.S. Grimshaw, A.J. Ferrari, F.C. Knabe and M.A. Humphrey, "Wide-Area Computing: Resource Sharing on a Large Scale," IEEE Computer, 32(5): 29-37, May 1999.
[36] J. Linn. Generic Security Service Application Program Interface Version 2, Update 1. RFC 2743. January 2000. Available: http://www.ietf.org/rfc/rfc2743.txt
[37] OpenSSL. http://www.openssl.org
[38] D. Harkins and D. Carrel. The Internet Key Exchange (IKE). RFC 2409. November 1998. Available: http://www.ietf.org/rfc/rfc2409.txt
[39] M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau, and H.F. Nielsen. SOAP Version 1.2 Part 1: Messaging Framework. W3C Recommendation 24 June 2003. Available: http://www.w3.org/TR/soap12-part1/
[40] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Draft of 6/22/02. http://www.gridforum.org/ogsi-wg/drafts/ogsa_draft2.9_2002-06-22.pdf
[41] Open Grid Services Architecture (OGSA) Working Group. https://forge.gridforum.org/projects/ogsa-wg
[42] Global Grid Forum. http://www.ggf.org
[43] S. Tuecke et. al. Open Grid Services Infrastructure (OGSI) Version 1.0. Global Grid Forum. GFD-R-P.15. Version as of June 27, 2003.
[44] I. Foster, e*t. al.* Modeling Stateful Resources with Web Services. Available: http://www.globus.org/wsrf/ModelingState.pdf
[45] E. Eastlake, J. Reagle, and D. Solo, eds. XML-Signature Syntax and Processing. W3C Recommendation 12 February 2002. Available: http://www.w3.org/TR/xmldsig-core/
[46] D. Eastlake, J. Reagle, eds. XML Encryption Syntax and Processing. W3C Recommendation 10 December 2002. Available: http://www.w3.org/TR/xmlenc-core/
[47] Organization for the Advancement of Structured Information Standards (OASIS). http://www.oasis-open.org
[48] A. Nadalin, C. Kaler, P. Hallam-Baker, and R. Monzillo, eds. Web Services Security: SOAP Message Security 1.0 (WS-Security 2004). Monday, 15 March 2004. Available: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
[49] G. Della-Libera et. Al. Web Services Secure Conversation Language (WS-SecureConversation). Version 1.0. December 18, 2002. Available: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-secureconversation.asp
[50] Globus Toolkit version 3.x . http://www.globus.org/gt3/
[51] Python OGSI Client and Implementation (pyGridWare). http://www-itg.lbl.gov/gtg/projects/pyGridWare/index.html
[52] G. Wasson, N. Beekwilder, M. Morgan, and M. Humphrey. OGSI.NET: OGSI-compliance on the .NET Framework. In *Proceedings of the 2004 IEEE International Symposium on Cluster Computing and the Grid.* April 19-22, 2004. Chicago, Illinois.
[53] OGSI.NET and WSRF.NET. University of Virginia http://www.cs.virginia.edu/~humphrey/GCG/ogsi.net.html
[54] Matt Blaze, Joan Feigenbaum, and Angelos D. Keromytis. KeyNote: Trust Management for Public-Key Infrastructures. In Proceedings of the 1998 Security Protocols International Workshop, Springer LNCS vol. 1550, pp. 59 - 63. April 1998, Cambridge, England. Also AT&T Technical Report 98.11.1.
[55] Grid Policy Research Group, GGF https://forge.gridforum.org/projects/policy-rg/
[56] H. Stem, M. Eisler, R. Labiaga. *Managing NFS and NIS*, 2nd Edition O'Reilly, July 2001
[57] M. Satyanarayanan. Scalable, Secure, and Highly Available Distributed File Access. IEEE Computer. May 1990, Vol 23, No. 5.
[58] M. Wahl, T. Howes, and S. Kille. Lightweight Directory Access Protocol (v3). RFC 2251. December 1997. Available: http://www.ietf.org/rfc/rfc2251.txt
[59] L. Pearlman, V. Welch, I. Foster, C. Kesselman, S. Tuecke. A Community Authorization Service for Group Collaboration. *Proceedings of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks*, 2002.
[60] Virtual Organization Membership Service (VOMS). http://hep-project-grid-scg.web.cern.ch/hep-project-grid-scg/voms.html
[61] Mary Thompson, William Johnston, Srilekha Mudumbai, Gary Hoo, Keith Jackson. Certificate-based Access Control for Widely Distributed Resources . *Proceedings of the Eighth Usenix Security Symposium*, Aug. `99
[62] David Chadwick and Alexander Otenko. The PERMIS X.509 role based privilege management infrastructure. Future Generation Computer Systems. Volume 19, Issue 2 (February 2003). Special section: Selected papers from the TERENA networking conference 2002 .pp. 277 - 289
[63] Organization for the Advancement of Structured Information Standards (OASIS). Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1. OASIS Standard, 2 September 2003.
[64] OpenSAML – an Open Source Security Assertion Markup Language implementation. Internet2. http://www.opensaml.org/
[65] Organization for the Advancement of Structured Information Standards (OASIS). Extensible Access Control Markup Language (XACML) Version 1.0. OASIS Standard, 18 February 2003. http://www.oasis-open.org/committees/xacml/
[66] Sun's XACML Implementation. http://sunxacml.sourceforge.net/
[67] Global Grid Forum OGSA Authorization Working Group. https://forge.gridforum.org/projects/ogsa-authz

[68] K. Czajkowski, A. Dan, J. Rofrano, S. Tuecke, and M. Xu. Agreement-based Service Management (WS-Agreement). Global Grid Forum draft-ggf-graap-agreement-1. Version as of Feb 8, 2004.
[69] G. Della-Libera et. al. Web Services Security Policy (WS-SecurityPolicy). Version of 18 December 2002. http://www-106.ibm.com/developerworks/library/ws-secpol/
[70] D. Box, et. al. Web Services Policy Framework (WS-Policy). Version of 28 May 2003. http://www-106.ibm.com/developerworks/library/ws-polfram/
[71] N. Dulay, Lupu, E., Sloman, M. and Damianou, N. 2001. A Policy Deployment Model for the Ponder Language. *Proc. IEEE/IFIP International Symposium on Integrated Network Management (IM'2001)*
[72] C. Bettini, Jajodia, S., Wang, X.S., Wijesekera, D. 2002. Obligation Monitoring in Policy Management. *Policy 2002 Workshop*.
[73] B. Sundaram, and B. Chapman. XML-Based Policy Engine Framework for Usage Policy Management in Grids. *Proceedings of the Third International Workshop on Grid Computing (Grid 2002).* Baltimore, MD, November 2002.
[74] D. Verma, S. Sahu, S. Calo, M. Beigi, and I. Chang**.** A Policy Service for GRID Computing. *Proceedings of the Third International Workshop on Grid Computing (Grid 2002).*
[75] G. Wasson and M. Humphrey. Policy and Enforcement in Virtual Organizations. In *4th International Workshop on Grid Computing (Grid2003)* (associated with Supercomputing 2003). Phoenix, AZ. Nov 17, 2003.
[76] E. F. Michiels (co-editor). Information Technology-OSI-Security Frameworks for Open Systems: Access Control Framework . 1995.
[77] D. Durham, ed. The COPS (Common Open Policy Service) Protocol. RFC 2748. January 2000. Available: http://www.ietf.org/rfc/rfc2748.txt
[78] I. Djordjevic, T. Dimitrakos, C. Philips An Architecture for Dynamic *Security Perimeters of Virtual Collaborative Networks* Proceedings of the 9th IEEE/IFIP Network Operations and Management Symposium (NOMS 2004) IEEE CS, (April 2004)
[79] GRASP project, Grid-like architecture for Application Service Providers, http://www.bitd.clrc.ac.uk/Activity/ACTIVITY=GRASP;
[80] V. Paxson, Bro: A System for Detecting Network Intruders in Real-Time, Computer Networks, 31(23-24), pp. 2435-2463, 14 Dec. 1999. (HTML)
[81] IBM and Microsoft. Security in a Web Services World: A Proposed Architecture and Roadmap. April 7, 2002,version 1.0. http://www-106.ibm.com/developerworks/library/ws-secmap/
[82] N. Nagaratnam *et. al.* Security Architecture for Open Grid Services. Global Grid Forum Working Draft. Revision as of 6/5/2003.
[83] F. Siebenlist *et. al.* OGSA Security Roadmap: Global Grid Forum Specification Roadmap towards a Secure OGSA. Global Grid Forum Working Draft. July 2002.

Marty A. Humphrey was awarded the Bachelor of Science degree in electrical and computer engineering in 1986 and Master of Science degree in electrical and computer engineering in 1988 from Clarkson University, Potsdam, NY, and a PhD in computer science from the University of Massachusetts in 1996. From 1998 to the present, he has been with the Department of Computer Science at the University of Virginia, Charlottesville, VA where he was first a Research Assistant Professor and is currently (2002-) an Assistant Professor. His areas of research include many aspects of Grid Computing, including security, programming models, performance, Grid testing, and Grid usability. He is active in the Global Grid Forum, where he serves as co-director of the Security Area and also co-chair of the OGSA Security working group.

Mary R. Thompson is currently a Staff Scientist and Group leader at Lawrence Berkeley National Laboratory. She received a BS in Physics from Stanford University in 1963 and an MS in Computer Science from the University of California at Santa Barbara in 1976. She joined the Computational Research Division at Berkeley National Laboratory in 1995. Prior to that she worked at Carnegie Mellon university on the Mach and Hydra operating systems, and at the Laboratory for Computer Science at MIT on the Multics operating system. Her research interests include distributed access control, secure communications, and PKI based applications. She is a member of the Global Grid Forum, active in the security area, and an IEEE member.



Keith R. Jackson is currently a Scientist at the Lawrence Berkeley National Laboratory, where he is a member of the Secure Grid Technology Group. He has been involved in developing a PKI based authorization system (Akenti), and a secure advanced reservation system (STARS). He is currently a Principal Investigator on three projects focused on developing component-based interfaces to "Grid" services, and prototyping large-scale computational and data "Grids". His interests include distributed access control, distributed system security, advanced reservations, network quality of service, component based middleware, and PKI based applications.