Sicurezza nelle Grid

Laura Pearlman
USC Information Sciences Institute
(and other members of the Globus Project team)

Sommario

- **Il Problema della Sicurezza nelle Grid**
- Background: Public Key Encryption and Certificate-Based Authentication
- Grid Security Infrastructure
- Autorizzazione

Il Problema della Sicurezza nelle Grid (1)

- Le risorse sono presenti domini amministrativi multipli e distinti ("siti").
- Una singola applicazione può usare risorse di più siti.
- Ogni sito ha i propri preesistenti requisiti, politiche, e meccanismi di sicurezza.

Il Problema della Sicurezza nelle Grid (2)

- Le Virtual Organization (VO) hanno le loro proprie politiche e specifiche di sicurezza.
- Un singolo utente o risorsa può far parte di più VO
- Gli insiemi di utenti, risorse, e siti in una VO può essere grande, dinamico, e variabile.
- E' spesso difficile stabilire relazioni fiduciarie tra siti.

Grid Security: Requisiti Utente

- Facilità d'uso (es., singolo controllo di accesso)
- Capacità di eseguire applicazioni che usano risorse distribuite in una VO
- Modello di affidabilità basato sugli utenti.
- Proxies / agenti

Requisiti dei Siti

- Aderenza alle politiche del sito (riguardanti autorizzazione, auditing, accounting)
- Interoperabilità dei meccanismi di sicurezza locali
- Politiche e meccanismi comprensibili e verificabili

Requisiti delle VO

- Aderenza alle politiche della VO (riguardanti autorizzazione, auditing, accounting)
- Politiche e meccanismi comprensibili e verificabili
- Meccanismi che gestiscano molti siti (scalabili).

Alcune Operazioni di Sicurezza

- Autenticazione
- Autorizzazione
- Revoca
- Protezione di Messaggi
  - Message integrity
  - Message confidentiality
- Delegation
- Auditing e Accounting
- Non-ripudio

Autenticazione

- Verifica dell'identità.
- Alcuni meccanismi esistono:
  - Username/password
  - Kerberos
  - Meccanismi a Chiave Pubblica
  - Biometrica

Autorizzazione

- Verifica dei diritti di uso
- Molti meccanismi esistono per la specifica e l'applicazione:
  - Offerti dai Sist. Operativi (es., permessi sui file Unix)
  - Offerti dalle applicazioni (e.g., permessi in un DBMS)
- Usualmente richiedono autenticazione, ma non sempre.

Revoca di Diritti

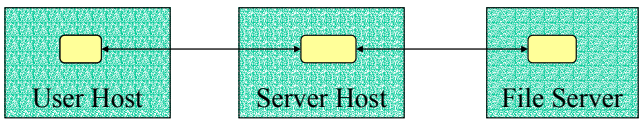
- Invalidare una credenziale di autenticazione o rimuovere una autorizzazione.
- Meccanismi di Revoca dipendono dai corrispondenti meccanismi di autenticazione/ autorizzazione.
- Problema principale: quanto tempo è necessario perchè una revoca abbia effetto?

Protezione di Messaggi

- Integrità
  - Autenticare il messaggio, e
  - Verificare che il messaggio ricevuto sia lo stesso messaggio inviato.
  - Una firma è un meccanismo di verifica dell'integrità anche se il mittente è offline.
- Riservatezza:
  - Assicurare che nessuno tranne il mittente e il destinatario possano leggere il messaggio.

Delegation

- Delegation è l'atto di fornire tutti o alcuni dei propri diritti ad un'altra entità.
- Ad esempio, un utente può eseguire un processo su un host remoto che richieda l'acceso a risorse su un terzo host.



Auditing e Accounting

- Auditing
  - Chi ha fatto che cosa su questa risorsa?
  - Richiede autenticazione
- Accounting
  - Chi ha usato e per quanto questa risorsa?
  - Politiche di costi

Non Ripudio

- Problema: come verificare un messaggio firmato quando la credenziale usata per firmarlo non è più valida?
  - La credenziale potrebbe essere stata revocata
  - La credenziale potrebbe essere scaduta
  - Il meccanismo di autenticazione usato nella firma potrebbe essere stato manomesso.
- Servizi di Non-ripudio sono servizi di terze parti che certificano e un marcano temporalmente un messaggio firmato.

Outline

- The Grid Security Problem
- Background: Public Key Encryption and Certificate-Based Authentication
- Grid Security Infrastructure
- Authorization

Public Key Encryption

- Encryption function E takes a key (k) and a message (m) and returns an encrypted message: $E(k, m_{plain}) = m_{cypher}$
- Public and private keys are inverses:

  $E(k_{public}, E(k_{private}, m)) = m$ and

  $E(k_{private}, E(k_{public}, m)) = m$.
- Knowing k_{public} does not make it easy to guess $k_{private}$.

Public Key Signatures

- The signer uses a well-known hash function (h) to compute a hash of the message to be sent and encrypts the hash with the private key:

  $H_{plain} = h(m)$

  $H_{cypher} = E(k_{private}, H_{plain})$

  thus:

  $h(m) = E(k_{public}, H_{cypher})$
- The signer sends the original message (m) plus the encrypted hash (H_{cypher}).
- The recipient calculates the message hash directly and decrypts the encrypted hash, and verifies that the two values are equal.

Public Key Authentication

- If Alice wants to verify Bob's identity, she first sends him a challenge message (m).
- Bob encrypts the challenge message with his private key, and sends Alice the result:

  $m' = E(k_{private}, m)$
- Alice decrypts Bob's message with his public key and verifies that it's the same as the original challenge:

  $M = E(k_{public}, m')$

Using Public Key Encryption for Confidentiality

- If Alice wants to send Bob a confidential message, she encrypts the message with his public key; he can then read it using his private key.
- Sometimes this is combined with authentication: Alice can encrypt a message with her private key and then encrypt the result with Bob's public key.

In Real Life…

- Because public-key encryption is computationally expensive, systems such as SSL/TLS use secret-key encryption (using a new, randomly-generated session key) for most operations. They use public-key encryption only for authentication and key exchange during session initiation.

Key Management

- PEM model: central registry of user/key mappings
- PGP model (also used by ssh): each person keeps track of the keys of each person they want to talk to.
- Certificate model: each person keeps track of the keys of a small number of trusted Certificate Authorities (CAs); these CAs sign certificates binding keys to names.

Advantages/Disadvantages of Certificate Approach

- Advantages
  - Third-party authentication, but the third party does not have to be online for authentication to work
  - Trust is decided by individuals.
  - Individuals retain complete control over their credentials.
- Disadvantages
  - Revocation can be a slow process
  - Site administrators often do not trust users to manage certificates wisely

X.509 Certificates

- An X.509 Certificate is a statement, signed by the issuer, binding a key to a name. It consists of:
  - A base structure containing:
    - A subject name
    - A public key
    - A validity time
    - The issuer's subject name
    - A "CA" flag
    - … and other information
  - Plus the issuer's signature of the base structure defined above
- The certificate can be made public; the associated private key must be kept private.

Certificate Extensions

- Critical extensions – may not be ignored
- Non-critical extensions – may be ignored

Getting an X.509 Certificate

- Typically, an individual (an "end entity") generates a public/private key pair, and puts the public key into a certificate request, which is sent to the CA.
- The CA decides whether or not to honor the request; if so, the CA creates a signed certificate and returns it to the end entity.
- No one but the end entity ever sees the private key.

CA Responsibilities

- Issuing certificates
- Revoking certificates
- Renewing certificates that are about to expire
- Replacing certificates for users who forgot their passwords
- Authenticating/authorizing requests for all of the above
- Maintaining certificate policy and certificate practice statements.

Deciding to Trust a CA

- Some factors to be considered:
  - Physical and operational security of CA site
  - CA policies about certificate issuance (who can get one, what kind of identification is required, etc.).
  - CA policies about certificate lifetime, revocation, renewal, etc.
- To help evaluate these factors, a CA typically publishes statements about its policies and practices.
- Mechanics of trusting a CA: relying party must do some (minor) configuration on each host.

Typical Workflow for a Small CA

- User sends certificate request
- CA authenticates request (e.g., talks to user in person to verify that the request came from that user, and checks id), and verifies that the user was authorized to make the request. This step may be delegated to a separate Registration Authority (RA)
- CA issues certificate and sends it to the user.

Other Ways to Generate and Manage Certificates

- Large CA with distributed RA
- Gateways to other authentication systems
  - Kx509 – generates certificates based on kerberos authentication
- Certificate repositories
  - Hold user credentials (certificates and private keys)
  - Typically use username/password authentication

Authenticating with an X.509 Certificate

- Validate the certificate (see next slide)
- Verify that the remote entity knows the private key associated with the public key in the certificate.
- Verify that the issuer is a certificate authority that you trust.

Authentication: Validating an X.509 Certificate

- Build a certificate chain
  - May be short:
    1. CA_1: a "trust anchor", an already-known and trusted CA cert
    2. an end entity certificate signed by CA_1.
  - May be longer:
    1. CA_1: a "trust anchor", an already-known and trusted CA cert
    2. CA_2: a CA cert signed by CA_1

       …

    N. CA_N: A CA cert signed by CA_{N-1}

    N+1. EEC: An end-entity cert signed by CA_N
- Validate each certificate in the chain
  - Check signatures, validity times, etc.

Authentication: "Proof of Possession" (signature)

- Several standard protocols exist to validate proof of possession of a private key
- Signature-based protocols, such as xml-signature, specify signature formats for certain types of messages using certificates.

Authentication: "Proof of Possession" (challenge/response)

- Protocols such as SSL/TLS use a challenge-response mechanism:
  - User sends certificate over the wire
  - Relying party sends user a challenge string
  - User encrypts the challenge string with the private key
  - Relying party uses the public key to decrypt the encrypted challenge. If it comes out unchanged, the user knows the private key.

Authentication: Validating the CA

- Authentication is valid only if it's based on a credential issued by an authority that you trust.
- For certificate-based authentication, this means that you must decide which CAs you trust and acquire copies of their certificates.
- In validating a certificate chain, one of the CAs that appears in the chain must be the one of your trusted CAs.

Revoking an X.509 Certificate

- Certificate Revocation Lists (CRLs):
  - The CA issues CRLs, listing certificates that they've granted and later revoked.
  - CRL format is standardized and can be parsed by software.
  - Relying parties use their discretion in deciding how often to check CRLs.
- Online Certificate Status Protocol (OCSP):
  - The CA runs an OCSP server, which relying parties can query for certificate status.

Sommario

- Il Problema della Sicurezza nelle Grid
- Background: Public Key Encryption and Certificate-Based Authentication
- **Grid Security Infrastructure**
- Autorizzazione

Grid Security Infrastructure

- Fornisce
  - Autenticazione (one-way o mutua)
  - Integrità dei messaggi e riservatezza
  - Delegation
- Estende lo standard di certificazione X.509 per includere i *proxy certificates* per delegation e singolo accesso (sign-on)
- Due modi operativi: **Transport-level** e **Message-level** security.
- GSI è stato sviluppato come parte del Globus Toolkit.

GSI con Transport-Level Security

- Implementazione originale di GSI
- Usa SSL/TLS, esteso per single-sign-on e delegation
- Assume un protocollo di trasporto connection-based (es., TCP).
- Usa certificati X.509 per autenticazione e per stabilire session keys.

GSI con Message-Level Security

- Nuova implementazione di GSI
- Usa WS-Security, XML-Signature e i procolli associati
- Fornisce sia
  - una session-based security (che assume un protocollo di trasporto connection-based e usa session keys) e
  - una per-message security (che non richiede protocollo di trasporto connection-based).

Certificati Proxy X.509

- Prodotti da entità certificata (o un altro certificato proxy), non da una Autorità di Certificazione
  - Ha (effettivamente) lo stesso nome del suo produttore
  - Ha una chiave pubblica/privata differente al suo produttore.
- Permette al possessore (del certificato e della chiave privata associata) di impersonare il produttore, con alcune restrizioni:
  - Usualmente un tempo di validità più breve
  - proxy flag limitati
  - restrizioni alle autorizzazioni
- Usata per singolo sign-on e delegation

Autenticazione con Certificati Proxy X.509

- Simile alla autenticazione usando certificati X.509 regolari, eccetto il "certificate path" più lungo:
  1. CA_1: un "trust anchor", già noto e sicuro certif. CA

     …

  N+1. EEC: un certif. end-entity firmato da CA_N

  N+2. PC1: un certif. proxy firmato da EEC

  N+3. PC2: un certif. proxy firmato da PC1

     …
- Estensione alla sintassi X.509; non riconosciuta da software non-GSI (attualmente un Internet Draft in IETF).

Singolo Sign On

- Un utente può voler fare molte operazioni che richiedono l'autenticazione durante una giornata. Tradizionalmente, questo richiederebbe:
  - Inserire la password a chiave privata molte volte, o
  - Tenere la chiave privata decriptata sul disco, o
  - Fare tutte le operazioni che richiedono autenticazione da una istanza di un programma, o
  - Usare un hardware di autenticazione specializzato.

Usare i Certificati Proxy X.509 per Singolo Sign On

- L'utente:
  - Crea un certificato proxy di breve periodo
  - Mantiene il certificato e la chiave privata decriptata sulla memoria locale
  - Lo usa al posto di un certificato permanente
  - Al termine distrugge il proxy o lo lascia scadere.
- Vi è il rischio che la chiave privata proxy possa essere compromessa, ma il potenziale danno è limitato dal breve periodo di vita del proxy.

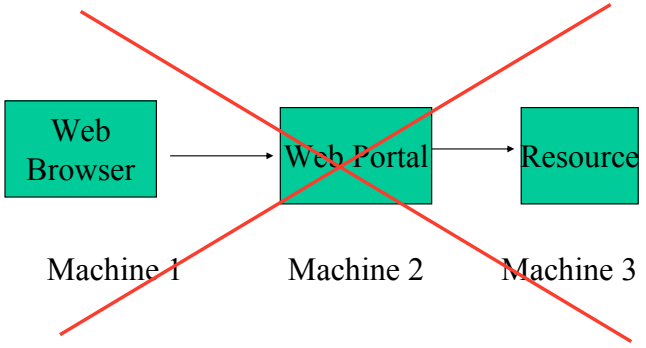
Usare i Certificati Proxy X.509 per Delegation

- Assumiamo che un processo utente sull' host **A** vuole delegare un processo server sull' host **B**, che deve accedere risorse sull' host **C**.
  1. Il processo server genera una coppia di chiavi e invia una richiesta (con la chiave pubblica) al processo utente.
  2. Il processo utente usa il suo certificato proxy locale ($\mathbf{PC_A}$) per firmarne uno nuovo ($\mathbf{PC_B}$) in risposta alla richiesta del server
  3. Il processo server sull'host **B** quindi usa $\mathbf{PC_B}$ (e la chiave privata generata al passo 1) per autenticarsi sull' host **C**.
- Nessuna chiave privata viene inviata sulla rete.

Esempio di Delegation

Macchina 1 Macchina 2 Macchina 3

Esempio di non-Delegation



Altri Usi dei Certificati Proxy

- **Restricted Proxy Certificate**: include una politica che limita gli usi del certificato
- **"Independent" Proxy Certificate**: non include diritti (i diritti devono essere delegati esplicitamente).

Sommario

- Il Problema della Sicurezza nelle Grid
- Background: Public Key Encryption and Certificate-Based Authentication
- Grid Security Infrastructure
- **Autorizzazione**

Obiettivi delle Autorizzazioni in Griglie

- Compatibile con le politiche di sicurezza esistenti nei siti.
- Compatibile con le politiche di sicurezza delle VO.
- Facile da capire e verificare.
- Facile da amministrare.
- Compatibile con i meccanismi di sicurezza dei siti.

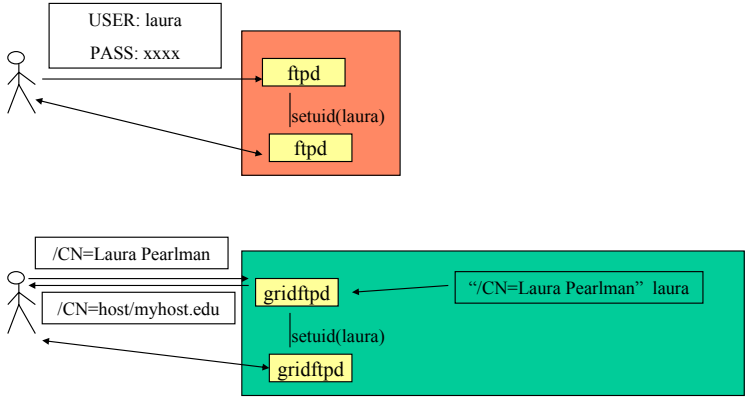
Metodi di Autorizzazione nelle Grid

- Metodi di Autorizzazione "Classici" :
  - Mapping d'Identità
  - Autorizzazioni "Self" e "service"
  - Autorizzazione Diretta
- Altri Metodi di Autorizzazione :
  - Community Authorization Service
  - Akenti
  - PERMIS

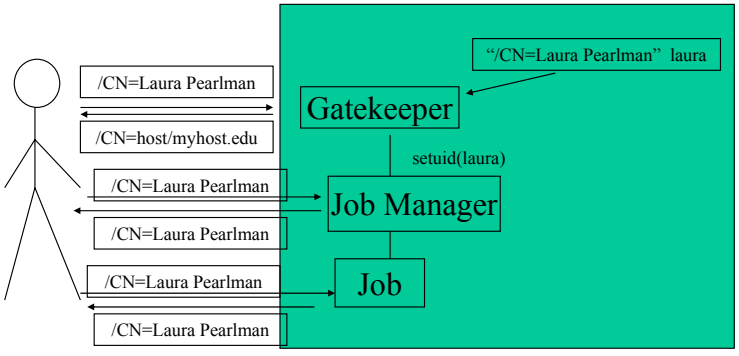
Autorizzazioni "Classiche" nelle Grid

- **Mapping d'Identità** : associa una "grid identity" (cioè, il Distinguished Name da un certificato utente X.509) ad una identità locale, che permette al sistema operativo di fare gli opportuni controlli.
- **"Self" authorization**: permette l'accesso se l'identità remota è uguale alla identità corrente.
- **"Service" authorization**: usa un algoritmo basato su *host name* e *service name* per determinare se l'identità remota è una di quelle accettabili.

Esempio di Autorizzazione : FTP e GridFTP



Authorization (and delegation) Example: GRAM



Caratteristiche del Mapping d'Identità

- Facile da capire e implementare per servizi che hanno un modello di autorizzazione esistente basato sull'identità locale.
- L'amministratore del sito mantiene il controllo locale.
- Richiede un accesso "root" nel sito per gestire gli utenti, e in qualche caso per garantire i permessi.
- I permessi che possono essere garantiti dipendono dalle implementazione dei siti.

Autorizzazione Diretta

- Usa un meccanismo di controllo degli accessi basato sui subject name di un servizio preesistente.
- Funziona per servizi che ne hanno già uno…
- … ma per questi è molto semplice.
- E' dipendente dall'implementazione:
  - L'insieme dei permessi che sono offerti dipendono dall'implementazione
  - Puo' richiedere l'intervento degli amministratori dei siti per garantire e revocare permessi.

Community Authorization Service (CAS)

- I siti eseguono il controllo degli accessi a "grana grossa" garantendo diritti di accesso a gruppi di risorse per le Virtual Organizations (VO)
- Le VO usano le CAS per eseguire il controllo degli accessi a "grana fine", garantendo diritti di accesso di singoli utenti a singole risorse.
- I server di risorse rafforzano sia la politica dei siti sia la politica delle VO.

Gestione delle Politiche CAS

- I siti mantengono le politiche locali usando metodi esistenti (es., gridmap files e unix account).
- Le politiche di comunità sono mantenute usando il CAS server e il protocollo CAS.
- I siti non devono gestire politiche per le gli utenti e i gruppi di comunità.

Community Authorization Service (CAS)

- Gli amministratori di VO usano l'interfaccia amministrativa di CAS per definite le politiche delle VO.
- Gli utenti contattano il CAS server per ottenere "asserzioni firmate" sulle operazioni possibili (basate della politica della VO)
- Gli utenti presentano le asserzioni firmate ai server di risorse durante l'autenticazione.

Signed Authorization Assertions

Subject: /O=Grid/CN=Laura

Valid: 3/25/03 11:00 – 3/26/03 11:00

AuthorizationAssertion (non-critical extension):

Target Subject: /O=Grid/CN=Laura

Valid: 3/25/03 13:00 –15:00

These actions are allowed:

Read gridftp://myhost/mydir/*

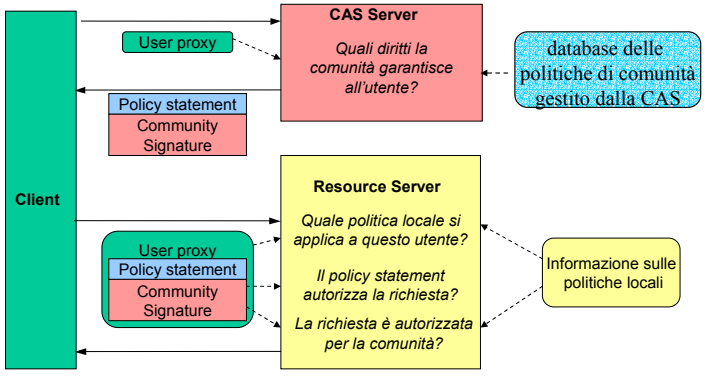
Signature (of assertion, by the VO CAS server)

Signature (of all above, by the user)

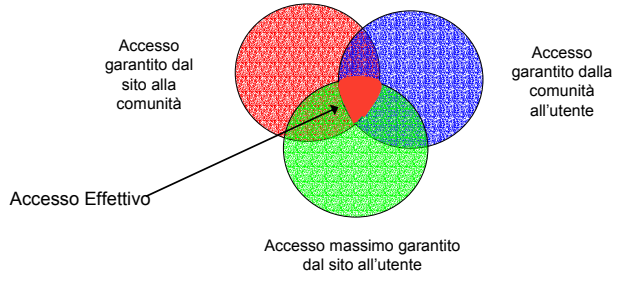
L'asserzione di autorizzazione è firmata dal CAS server della VO. Esso delega un sottoinsieme dei diritti della VO ad un utente per un dato tempo.

Essa è valida se usata con le credenziali di autenticazione dell'utente.

Una Richiesta CAS Tipica



Politiche Effettive CAS



Akenti

- Sistema di autorizzazione sviluppato al LBNL
- Progettato per permettere a più soggetti di fornire informazione di controllo di accesso.
- Usa policy files, use-condition certificates, e attribute certificates.
- I proprietari delle risorse creano policy files che indicano dove cercare e chi è autorizzato a creare, use-condition certificates e attribute certificates.

Akenti Use-Condition Certificates

- Certificati firmati che definiscono condizioni di uso di una risorsa. Es., "tutti quelli con attributo *X* possono leggere il file *A*").
- Possono essere indicati come "critici" se indicano le condizioni che si DEVONO soddisfare per l'accesso alle risorse.
- Tutti i certificati vanno controllati per decidere l'accesso alle risorse.

Akenti Attribute Certificates

- Certificati firmati che associano attributi ad identità
- Gli attributi non restringono i diritti di accesso (es., non vi sono attributi che dicano "nessuno con attributo X può leggere un file A").
- Non è necessario avere tutti gli attributi per decidere un accesso ad una risorsa.

Role-Based Access Control (RBAC)

- I Ruoli sono insiemi di permessi su oggetti.
- Gli utenti sono associati ai ruoli.
- Le definizioni dei ruoli (quali ruoli possono decidere accessi a risorse) sono tenute separate dall'informazione sull'assegnazione dei ruoli (quali utenti in quali ruoli).

PERMIS

- Sistema di autorizzazione basato sui ruoli sviluppato all'University of Salford
- Usa *policy certificates* (quali ruoli possono operare su quali risorse?) e *role certificates* (chi è in quel ruolo?).
- Attualmente, le API PERMIS prelevano le appropriate *policy* e *role certificates* dai server LDAP per prendere decisioni di autorizzazioni.

Altri lavori sulle Autorizzazioni

- Standardizzazione nel GGF OGSA-Authorization group:
  - Servizio di Autorizzazione (policy decision point)
  - Linguaggio di politiche
  - Attribute assertion language

the globus project
www.globus.org

Ulteriori Informazioni

- Globus Security: http://www.globus.org/security
- Akenti: http://www-itg.lbl.gov/Akenti/homepage.html
- PERMIS: www.permis.org
- Global Grid Forum security area: https://forge.gridforum.org/projects/sec