Grid Security in Practice

John Watt

http://csperkins.org/teaching/2004-2005/gc5

UNIVERSITY
of
GLASGOW

Overview

- Current options for security implementations
- Grid Security Infrastructure (GSI)
  - Certificates, authentication and proxies
  - GT3 Implementation of GSI
- Authentication
- Authorisation
  - PERMIS
  - Other Authorisation mechanisms
- *** Programming Exercise Update ***

Security Ingredients

- Authentication
  - checking you are who you say you are
- Authorisation
  - allowing you to do only what you're allowed to do
- Audit/accounting
  - checking what you're doing and when
- Confidentiality
  - making the data you use secure
- Privacy
  - making sure the data is subsequently used securely
- Integrity
  - making sure your data isn't corrupted
- Fabric management
  - ensures minimal impact from other applications' security faults
- Trust
  - how much we expect you to do what we asked with what we gave you

Options?

- Kerberos
  - Authenticates users through a secure transaction with a centrally maintained key server
  - Designates trustworthy key servers in other organisations
    - Mechanism for inter-organisational authentication
- OK, but
  - Sites need to negotiate many cross-realm authentication agreements
    - Surrenders too much control of the local security policy.
    - Requires organisation to be completely ‘Kerberos’
      - Inter-site AND intra-site communication

Options?

- Secure Shell (SSH)
  - Widely used, PKI based, simple technology
  - Protects user credentials with link encryption
  - Very easily deployed
- Easy, but
  - Users need to manage lots of different passwords/public keys for inter-site work
  - No authorisation control
    - Without invasion of privacy
  - Only supports simple tasks (remote shell or file transfer)
    - Doesn't support collaborative environments (or Web browsers!!!)

Grid Security Infrastructure

- The implementation of secure functionality in Globus Toolkit…
  - Provides secure communication between grid elements
  - Preserves site control for access policies and local security
  - Supports “single sign-on” and delegation of credentials for applications that require multiple resources
- All elements of the Globus Toolkit are built on top of this basic infrastructure

Certificates

- A central concept in GSI is that of the digital certificate
  - Unique to every user (or resource) on the grid
  - Signed by a certification authority (CA)
  - Encoded in the standard X509 format
    - Compatible with web browsers (format established by the Internet Engineering Task Force)
  - The UK e-Science Certificate Authority issues certificates to the e-Science community
  - But one can define your own CA using the globus toolkit package simpleCA

GSI certificates

- GSI certificates includes four primary pieces of information
  - A “subject”
    - Identifies the person or object the certificate represents
  - A public key belonging to the subject
  - The identity of a third party (the Certificate Authority) that has signed the certificate to certify the public key and subject name belong to the subject
    - To trust the certificate you MUST trust the CA
  - The digital signature of the CA
    - To ensure the information on the certificate hasn’t been changed

Digital Signatures

- Ensures integrity of the certificate
  - To digitally sign a certificate (or any information)
    - Compute a mathematical hash of the info using an algorithm known to the intended recipient (this doesn't have to be secret!)
    - Using your private key, encrypt this hash and attach it to the end of the info (recipient needs your public key [in cert anyway!])
    - Recipient will compute the hash of the info using the algorithm you agreed on
    - Recipient then decrypts the encrypted hash you attached to the end of the info
    - If the hash computed by the recipient and the decrypted hash encrypted by you are the same this verifies that:
      - YOU signed the information
      - The information hasn't changed since you signed it

Mutual Authentication

- Two parties have certificates and they both trust the CAs that signed them
  - They can now prove to each other that they are who they say they are
  - In GSI, this process is called mutual authentication
    - GSI uses SSL (now known as TLS) for its mutual authentication protocol
    - Once mutual authentication is performed, GSI steps aside so communication can occur without encryption/decryption overheads
    - GSI can be used to establish a shared key for encryption if confidential communication is desired
      - e-Health and e-Commerce projects particularly desire this
    - GSI Integrity may also be turned on
      - data is readable but unchangeable by an intruder (less overheads than encrypting messages all the time

Mutual Authentication process

- Person A establishes connection with person B
  - A gives B their certificate
    - Tells B who A claims to be, what his public key is, and which CA is used to sign the certificate
  - B checks the digital signature of A's certificate
    - Checks CA actually signed the certificate and it hasn't been tampered with (remember B must trust A's CA)
  - B now checks that A is definitely A
    - B generates a random message and sends it to A
      - A encrypts the message with his private key and returns it to B
      - B decrypts the message with A's public key
      - If the message is the same as the original, A is definitely A
- The same operation from B to A completes mutual authentication – both parties know who they are talking to

Delegation and single sign-on

- Essential to easy access of multiple resources
  - GSI extends the SSL protocol to reduce the number of times you have to enter your passphrase
  - Done through the creation of *proxy certificates*
    - A proxy consists of a new *short-lifetime* certificate (with a new public key within it) and a new private key
    - The certificate is signed by the OWNER not the CA
    - The new private key must be kept secure – but since it is not valid for long it is ok to store it locally unencrypted – the file permissions should be enough
    - This proxy certificate and private key may be used for mutual authentication without the need to enter a password
      - The process differs slightly but establishes a chain of trust from the CA, through the owner, to the created proxy

GSI Authentication

- Log-on to the grid
  - grid-proxy-init
    - Enters your password to decrypt your private key, and creates a short-lived proxy certificate in the /tmp directory
      - Name is typically x509up_username
  - grid-proxy-destroy
    - Removes this proxy certificate
  - grid-cert-info
    - Returns the information contained in your certificate
    - Need this information for the grid-mapfile --- next

GSI Authorisation

- Achieved through use of a *grid-mapfile*
  - List mapping a user's grid identity to a local identity
  - Grid identity is the distinguished name from the user's X509 certificate
  - Local identity is associated with a local unix account
  - Stored in **`/etc/grid-security`**
  - Owned by root
- Every user needs to be present in this list to use the grid.
  - Administration nightmare when the grid scales up
    - Tackled by Role Based Access Control - coming up in 10 minutes…

A “typical” grid-mapfile

```
“/C=UK/O=eScience/OU=Glasgow/L=Compserv/CN=steve kee” skee
“/C=UK/O=eScience/OU=Edinburgh/L=NeSC/CN=stewart mills” smills
“/C=UK/O=eScience/OU=Glasgow/L=Compserv/CN=iain mcbride” imcbride
“/C=UK/O=eScience/OU=Aberdeen/L=GeSC/CN=nikki salter” nsalter
“/C=UK/O=eScience/OU=Newcastle/L=NEReSC/CN=nicola wightman” nwightman
“/C=UK/O=eScience/OU=London/L=LeSC/CN=scott mccaig” smccaig
“/C=UK/O=eScience/OU=Glasgow/L=Compserv/CN=kev mcneil” kmcneil
“/C=UK/O=eScience/OU=Glasgow/L=Compserv/CN=nik martin” nmartin
“/C=UK/O=eScience/OU=Edinburgh/L=NeSC/CN=ann robertson” aroberts
```

- Format:
  - “<Grid DN of user>”<space><local account name>…
    - DN must be in quotes as the surname tends to get chopped off if you miss it out

Grid Security Infrastructure

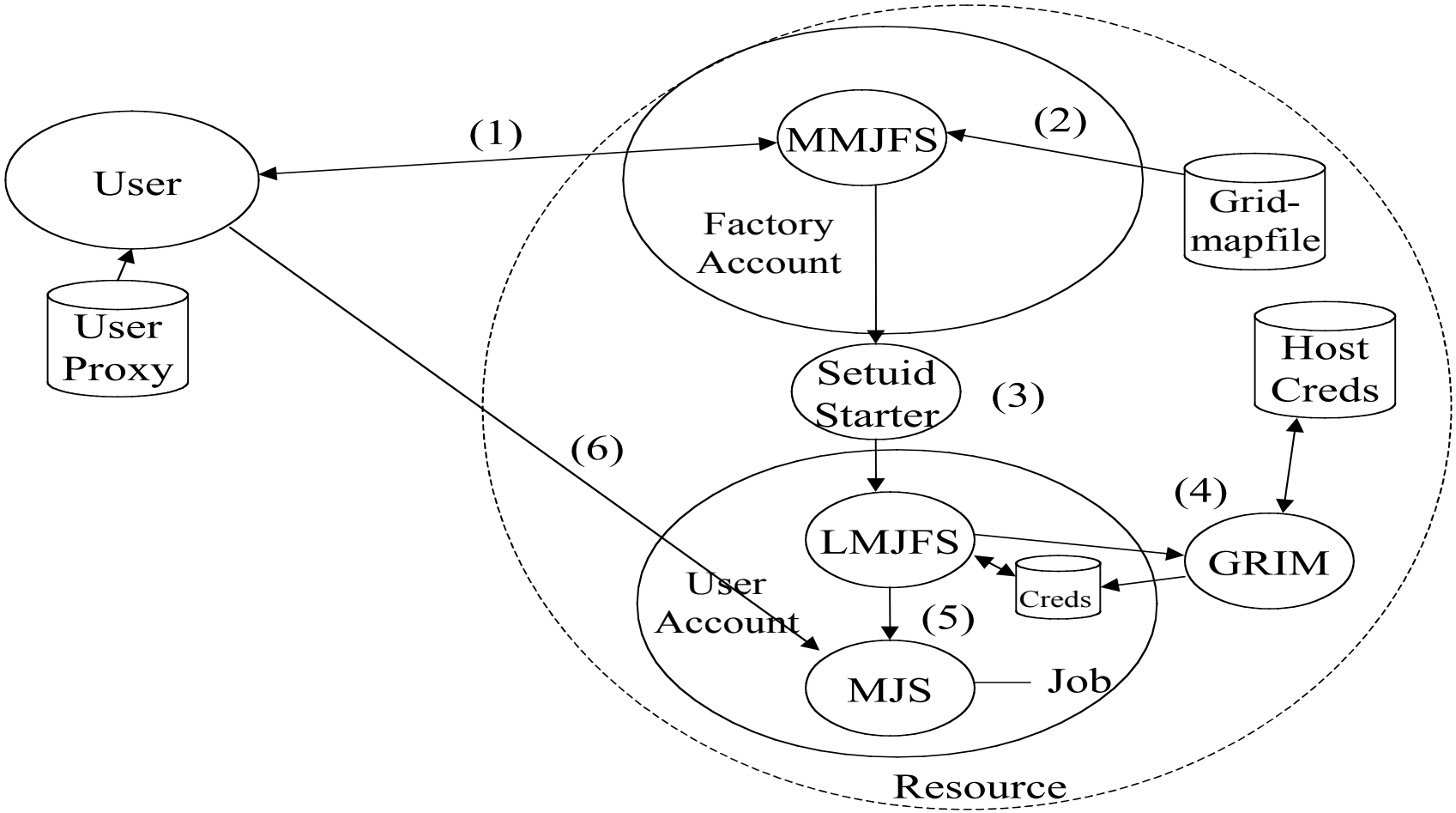
- All GSI APIs adhere to the GSS-API
  - Standard API for security systems produced by the IETF

- Remember (Lecture 4)
  - GT2 performed client-to-service delegation through a third party (the gatekeeper)
  - GT3 has removed this level of indirection through the use of Web Services with more secure results…

GT3 Security Enhancements

- Improved resource security model
  - GT3 services accepting connections from the network run with NO special local privileges, but use two programs to perform the actions that DO require them
    - setuid starter and GRIM… will describe these in a minute
  - Compromised network service can only give a denial of service
    - If service ran as ‘root’ we would be in trouble
    - Attacker may run the setuid programs, but these have very tight constraints over what they are capable of
  - Network services removed from user trust model
    - Instead, they invoke services required by the user using trusted local services the user may authenticate to.
      - Confused? More diagrams…

GT3 Job Submission



Step 1

- Client generates a proxy user certificate
  - Using grid-proxy-init
- Client generates a job request
  - (globusrun equivalent)
- Client signs this request with their proxy credentials and sends to the Master Managed Job Factory Service (MMJFS) on the remote resource.

Step 2

- The MMJFS runs in a non-privileged factory account.
  - e.g. user ‘globus’ in a typical GT3 install
- It verifies the signature on the request and establishes the identity of the user who sent it.
- It then determines the local account in which the job should be run.
  - Currently this is done by using the grid-mapfile and user's grid identity.

Step 3

- The MMJFS invokes the *setuid starter* process to start a Local Managed Job Factory Service (LMJFS) in the user's account
  - Assuming the user does not already have one running in their account

- *setuid starter*
  - a small setuid program running with root privileges that has the SOLE function of starting the LMJFS in the user's account.
    - Security!!

Step 4

- LMJFS calls the Grid Resource Identity Mapper (*GRIM*) to acquire a set of credentials.
- This ‘proxy’ credential has embedded in it
  - the user's Grid identity
  - the local account name and
  - local policy about the user.
    - The latter policy is obtained from the Grid map file entries that apply to that local account.
- *GRIM* is a setuid program that accesses the local host credentials and from them generates the proxy for the LMJFS.

Step 5

- MMJFS then forwards the original user-signed job instantiation request from the user to the LMJFS.
  - LMJFS verifies the signature on the request
    - make sure it has not been tampered with and to make sure it was created by a user that is authorized to run in the local user account.
- Once these checks are successfully completed, the LMJFS instantiates a Managed Job Service (MJS), presents it with the user's request, and returns a reference to the MJS to the user

Step 6

- User connects to the MJS.
- User and MJS then perform mutual authentication
  - the user using their proxy and the MJS using the credentials acquired from GRIM.
- The MJS authorizes the user as a valid user to access the local account it is running in.
  - The user authorizes the MJS as having a credential issued from an appropriate host credential and containing a Grid identity matching it's own, thus verifying the MJS it is talking to is running not only on the right host, but in an appropriate account. The user would then delegate GSI credentials to the MJS for the job to use and start the job running.

Authorisation

- Key to establishment of Virtual Organisations
- Recall from the GGF (Lecture 8):
  - SAML Authz specification defines elements for making assertion and queries regarding authentication and authorisation
  - Includes message exchange between a policy enforcement point (PEP) and a policy decision point (PDP)
    - SAML Authz provides generic PEP approach
- PDP is application specific. Will look at the PERMIS Privilege Management Infrastructure

PERMIS

- Developed an RBAC PMI that uses X509 Attribute Certificates to store users' roles
  - All ACs stored in one (or more) LDAP directories
  - Access control decisions driven by an authorisation policy (created by a Policy Editor)
    - Itself is stored in an X509 Attribute Certificate – guarantees integrity
    - Authorisation policy written in XML according to a DTD published at XML.org
  - Access Control Decision Function (ADF) written in Java – simple API (3 methods + constructor)
  - Privilege Allocator tool signs ACs and stores in LDAP for use by the ADF

PKI vs PMI

- A PMI is to authorisation what a PKI is to authentication – hence similar concepts

Concept	PKI Entity	PMI Entity
Certificate	Public Key Certificate (PKC)	Attribute Certificate (AC)
Certificate Issuer	Certification Authority (CA)	Attribute Authority (AA)
Certificate User	Subject	Holder
Certificate Binding	Subject's name to Public Key	Holder's Name to Privilege Attribute(s)
Revocation	Certificate Revocation List (CRL)	Attribute Certificate Revocation List (ACRL)
Root of trust	Root Certification Authority or Trust Anchor	Source of Authority (SOA)
Subordinate authority	Subordinate Certification Authority	Attribute Authority (AA)

PERMIS Policy Editor

- The local security policy is written using the PERMIS policy editor
- Version 1.3 out now from PERMIS website (with password), or from me!
  - http://www.dcs.gla.ac.uk/~jwatt/software/PolicyEditor.zip
  - RUN ON WINDOWS ONLY!! (machines in DCS)
  - Still a work in progress, may contain a few bugs
  - Will introduce the syntax now, so you are aware of the structure of a typical, simple (!) PERMIS policy
    - And to help identify any errors in the final XML

PERMIS Policy

- Subject Policy
  - Specifies the domain (specified as an LDAP subtree) of users who may be granted roles within the PMI

```
<SubjectPolicy>
   <SubjectDomainSpec ID="MyCompany">
        <Include LDAPDN="o=My Organisation,C=GB"/>
        <Exclude LDAPDN="ou=OtherCompany,o=MyOrganisation,C=GB"/>
   </SubjectDomainSpec>
   <SubjectDomainSpec ID="everyoneElse">
        <Include LDAPDN=""/>
   </SubjectDomainSpec>
</SubjectPolicy>
```

PERMIS Policy

- Source Of Authority (SOA) policy
  - Lists the LDAP DNs of SOAs which are trusted to issue roles to the subjects specified above
  - First name listed is the LDAP DN of the policy creator (required), subsequent names are SOAs which are "cross-certified" by the policy creator
  - This name(s) will become the root issuer name(s) in a signed Attribute Certificate
    - Any trusted AC for this policy must have been signed by one of them

```
<SOAPolicy>
   <SOASpec ID="MyCompanyAdmin" LDAPDN="cn=Admin,o=MyOrganisation,C=GB"/>
</SOAPolicy>
```

PERMIS Policy

- Role Hierarchy Policy
  - Defines the role hierarchies supported by this policy
  - Specified as a "directed graph" of Superior-Subordinate attribute values
  - Each role named using an attribute type, attribute value pair (e.g. permisRole,Slave)

```
<RoleHierarchyPolicy>
   <RoleSpec OID="1.2.345.0.1.321432.1.1.14" Type="permisRole">
        <SupRole Value="Boss">
                <SubRole Value="Slave">
        </SupRole>
        <SupRole Value="Slave">
   </RoleSpec>
</RoleHierarchyPolicy>
```

PERMIS Policy

- Role Assignment Policy
  - Specifies which roles can be given to which subjects by which SOAs
  - Supports delegation and time constraints (not used here)

```
<RoleAssignmentPolicy>
   <RoleAssignment ID="MyCompanyAdminAllocator">
        <SubjectDomain ID="MyCompany"/>
        <RoleList>
               <RoleType="permisRole" Value="Boss"/>
               <RoleType="permisRole" Value="Slave"/>
        </RoleList>
        <Delegate Depth="0"/>
        <SOA ID="MyCompanyAdmin"/>
        <Validity/>
   </RoleAssignment>
</RoleAssignmentPolicy>
```

PERMIS Policy

- Target Policy
  - Specifies the target domains covered by this policy
  - Give the name of your Grid Service you want to protect here

```
<TargetPolicy>
   <TargetDomainSpec ID="MyCompanyGridService">
        <Include URL="http://localhost:8080/ogsa/services/GridServices/
                core/first/MyCompanyGridService"/>
   </TargetDomainSpec>
</TargetPolicy>
```

PERMIS Policy

- Action Policy
  - Defines the actions (operations on targets) supported by this policy
  - Lets say the MyCompany Grid Service operates the doors in the MyCompany building…
  - ‘Name’ is the specific command to perform the action

```
<ActionPolicy>
   <Action Args="MyCompanyGridService" Name="lockMainDoor"/>
   <Action Args="MyCompanyGridService" Name="unlockMainDoor"/>
   <Action Args="MyCompanyGridService" Name="lockOfficeDoor"/>
   <Action Args="MyCompanyGridService" Name="unlockOfficeDoor"/>
   <Action Args="MyCompanyGridService" Name="getMainDoorStatus"/>
   <Action Args="MyCompanyGridService" Name="getOfficeDoorStatus"/>
</ActionPolicy>
```

PERMIS Policy

- Target Access Policy
  - Specifies which roles are needed to access which targets for which actions (+ under what conditions)
  - Operates a Deny All Unless Specifically Granted rule
  - One must possess all roles within a target clause to gain access (may need multiple ACs to access)

```
<TargetAccessPolicy>
   <TargetAccess ID="public">
        <RoleList/>
        <TargetList>
               <Target Actions="getMainDoorStatus,getOfficeDoorStatus">
                      <TargetDomain ID="MyCompanyGridService"/>
               </Target>
        </TargetList>
   </TargetAccess>
   <TargetAccess ID="slaves">
        <RoleList>
               <RoleType="permisRole" Value="slave"/>
        </RoleList>
        <TargetList>
               <Target Actions="unlockOfficeDoor,lockOfficeDoor">  etc…
```

PERMIS Policy

- Final policy is all the above code sandwiched together and then stored in your LDAP server
- A PERMIS DTD has been published, and is included in every XML PERMIS policy
  - Creators were using IBM Xeena and Microsoft IE which didn't support schemas

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X.509_PMI_RBAC_Policy SYSTEM
   http://sec.isi.salford.ac.uk/permis/Policy.dtd>
<X.509_PMI_RBAC_Policy OID="25.0.0.1">
```

***** All previous XML in here *****

```
</ X.509_PMI_RBAC_Policy OID="25.0.0.1">
```

Privilege Allocator

- Allows ACs to be created for each user (and SOA) in the LDAP repository
- The policy just created should be embedded in the certificate and stored in the SOA node.
  - Specify Common Name (CN) of User
    - 'Holder' is the LDAP DN of the Administrator
  - Specify serial number for this certificate
    - Make this the same number as the final number of the OID
      - Will get OID from a member of staff when you are ready…
  - Load your policy into the PA
  - Create and sign the certificate

PERMIS ADF

- Deployed as Grid Services within your container
  - Download permisAuthz.gar and authz-sample.gar
    - From PERMIS website http://sec.isi.salford.ac.uk/permis
  - Deploy into the Globus container
    - Using ant deploy with your ‘globus’ factory account
- Require modifications to your server-config.wsdd file (in $GLOBUS_LOCATION)
  - Find the entries related to your grid service and include the parameters to utilise the PERMIS authz service
- Client must contain calls to GSI methods

Other Authorisation Infrastructures

- Community Authorisation Service
  - Allows resource providers to specify course-grained access control policies in terms of communities as a whole, delegating fine-grained access control policy management to the community itself
  - Resource providers maintain ultimate authority over their resources but are spared day-to-day admin tasks
  - Builds on the Globus Toolkit GSI

  - … and it doesn't work. Simple as that!!
    - Has been made to work with GridFTP only

Other Authorisation Infrastructures

- VOMS
  - Virtual Organisation Membership Service
    - Provides info on the user's relationship with their VO
  - Supports
    - Single login with voms-proxy-init at the beginning of the session
    - Expiry time
    - Backwards compatible with non-VOMS services
  - Logging in to multiple VOs creates "aggregate" proxy certificates that allow access to all the resources
  - Is basically an account database which serves your credentials in a special format (VOMS credential)

And more

- AKENTI, CARDEA, myProxy
- All these are being evaluated as we speak..
  - No right/wrong way to do things yet
    - No real standards defined
  - By working with PERMIS you are contributing to knowledge about how we can implement easily administered, secure authorisation within the Globus Toolkit.

- And finally…

Programming Exercise News

- Change to the Exercise
  - We would like you to do the PERMIS based Grid Service questions (9 and 10) BEFORE the GSI version (7 & 8)
  - This is due to the machines being configured specially for PERMIS right now.
  - You may try it in the order given in the exercise, but you may have trouble when switching between the two implementations
    - So please try the PERMIS work before attempting the GSI stuff