Grid Security

Mike Jones

An overview of methods used to create a secure grid.

NESC 17 October 2003
TOC

- Grid Security
- PKI Overview
  - PGP and all that
  - Signing and Encryption
  - Trust
  - x509 certificates
- GSI
- UK eScience CA
- Globus Security
  - Host Configuration
  - User Setup
- CRLs
- Authorisation
- VO Management
- Firewalls
- Security in services
- Best Practices
Grid Security

- Making sure there are no back doors
  - Trustable users
  - Trustable Software
    - Not Spyware
    - Not Broken (or at least someone creates updates)

- Cryptography...
Grid Security

• Authentication, Authorisation and Accounting (AAA)

• Generic Security Services (GSS)
  – Public Key Infrastructure (PKI)
    • x509 Certificates and pkcs12 etc.
  – Kerberos – Shared Secrets (Needham-Schroder)
  – TSL, SSLeay, OpenSSL

• Grid Security Infrastructure (GSI)
  – Delegation, Third Party, Single Sign On
  – Proxy Certificates
PKI Overview

- Symmetric (eg DES) vs Asymmetric (eg RSA)
- Shared Key vs Public and Private Key

- Public Key: [e and N]
  - where N=pq product of two large primes
    - \((p-1)(q-1)\) is almost prime
    - and e (almost prime too)

- To encrypt/decrypt with Public Key:  \(c = (m^e) \mod N\)
PKI Overview

- Symmetric (eg DES) vs Assymetric (eg RSA)
- Shared Key vs Public and Private Key

- Public Key: [e and N]
  - where N=pq product of two large primes
  - \((p-1)(q-1)\) is almost prime
  - and e (almost prime too)

- Private Key: [d and N]
  - where \(e \times d = 1 \mod (p-1)(q-1)\)

- To encrypt/decrypt with Public Key: \(c = (m^e) \mod (N)\)
- To decrypt/encrypt with Private Key: \(m = (c^d) \mod (N)\)
PGP/GPG

Pretty Good Privacy / Gnu Privacy Guard

- Zimmermann: security in data transfer for public
  - Encrypt data using a symmetric key
  - Lookup RSA Public Key of recipient
  - Encrypt the symmetric key with RSA
  - Send the encrypted key & encrypted message

- Included: Key gen. with “entropy gathering device”, Data encryption, Data signing, and easy UI
We've discussed encryption/decryption

To Sign - encryption message with private key.

But... everyone should be able to read it so create a Hash and encrypt that instead.

Hash is a one way digest of the message by a specific algorithm (eg SHA1 or MD5)

Encrypt the hash and include it in the message.

Verify by making the hash and decrypting the signature
Trust

- We rely on ourselves to get true public keys
- Chain of trust rules
  - A public key may be digitally signed by many people
  - some of whom you may trust.
- CA method (Certificate Authority)
  - CA has a “root certificate” and a document called CP/CPS
    http://www.grid-support.ac.uk/ca/cps
  - You choose to trust on the basis of CP/CPS.
  - CA signs your certificate (your public key).
  - Large scale CAs are difficult and costly (~£220 per cert)
**GSI modifications**

- **Proxy Certificate**
- **Features:**
  - Issuer (Me! pretending to be a CA)
  - Short Lived
  - Contains Unencrypted RSA Key
  - Special Subject
  - Includes the issuer (Me!)
- **Requires Subject-issuer constraint**
Proxy Certificates

- `grid-proxy-init [-cert cert.pem -key key.pem]`

- `$X509_USER_PROXY = /tmp/x509up_u`id -u`
  - Contains certificate chain from but excluding CA
  - Contains unencrypted key
  - Has a short lifetime
  - Is read only to owner

- Limited Proxies on remote resources
  - `$X509_USER_PROXY = $HOME/.globus/.gass_cache/local/md5/cb/ab/8e/5031401cebf3a4b1da92857230/md5/95/5c/21/a3713a03e529757cc677fdb079/data`

- `grid-proxy-destroy`
The UK eScience Certificate Authority

- Read CPS
- Get CA cert
- Get CRL
- Request a certificate
- CertDB
- Export Certs

Gets you an x509 cert
Getting Certificates

- Create a private and public key pair
- Send public key to CA
- Identify yourself to the CA (as specified in CPS)
- CA signs your public key.
- CA sends you a digital certificate which contains your public key and the CA's digital signature
- Can be done two ways:
  - in your browser Netscape/IE certificate request
  - on the command line: grid-cert-request (pkcs10)
Globus Security

- Authentication using GSI extension to GSS
  - Third Party data transfer
  - Single Sign-on
  - Delegation

- Authorisation using grid-mapfile
  - Access to user level unix on remote machines
  - you get: terminal, a home directory, a group, a quota, access to basic IP (firewall allowing), access to queues (depending on jobmanager and advertising), access to CPU-sets, etc.
  - systems are notoriously heterogeneous (outside HEP they are very heterogeneous)
  - For stability could use Distributed Computing tools already available: http(s), afs?

- Accounting still at Research Group Level
  - Currently records: Who, When and Which Job-manager
  - Need much more: for Cost and Legal (80-90% malevolent attacks come from within!)
AA steps on a grid

- Host must identify itself to the grid (client or service)
- Users must recognise the host
- Users must identify themselves to the host
- The host must recognise the user's identity
- In a GT2 based grid the host allows the grid-user to become a local user
Host Security Setup

• To Identify the host
  – /etc/grid-security/hostcert.pem
    • openssl pkcs12 -in hostcert.p12 -nokey -clcerts -out hostcert.pem
  – /etc/grid-security/hostkey.pem
    • openssl pkcs12 -in hostcert.p12 -nodes -nocerts -out hostkey.pem
    • hostkey.pem must be readable to root only
Host Setup Cont.

• To identify the incoming grid-user

  NB – We're actually not identifying the grid user. We're identifying that the incoming request has access to the grid-user's private-key/proxy-private-key

  - `/etc/grid-security/certificates/<hash>.0`
  - `/etc/grid-security/certificates/<hash>.r0`
  - `/etc/grid-security/certificates/<hash>.signing_policy`
  - `/etc/grid-security/certificates/<hash>.crl_url (borrowed from EDG)`
CRL Practical

- Example of how to manage CRLS
- Download
  http://www.man.ac.uk/~zzcgumj/crls/crls.tar.gz
- Unpack in /etc/grid-security/certificates
- `crl.sh` does this:
  - read *.crl_url
  - Download CRLs
  - Check type (convert to pem) and Verify CRL
- `Makefile_crl` does this:
  - create symbolic link to <hash>.r0
Authorisation

• Globus uses grid-mapfile
  – Basic 1-to-1 mapping of 'certificate' to UID
  – EDG has a temporary fix:- pool accounts and LDAP VO's

• Alternatives:-
  Akenti engine - Attribute certificate (+ ...) -> Capability Certificate
  CAS – Assertion in proxy from CAS Server + Community Authz
  VOM – Web portal for management + ..... 
  VOMS – cf CAS, signed VOMS assertions added to proxy by client
  PERMIS – Role Based Access Control via X509 Attribute certificates
Firewalls

- Firewalls prevent attacks by blocking access:
  - on specified ports/port-ranges
  - from/to certain servers
- $GLOBUS_TCP_PORT_RANGE
- Globus needs a number of ports open for control and a large hole in your firewall for data streams.
GSIssh Security

- On port 2222 or 22
  - Can be configured with/without password authz on any port system or ephemeral
- Host certificate (uses /etc/host[cert|key].pem)
- Server authenticates itself to client
- Client can (must in L2G) authenticate itself with a Grid certificate
- Same configuration files as openssh except GSS is Globus not Kerberos
- Safer than ssh in that there is trust – proxy trade-off
GridFTP Security

- (x)inetd listens port 2811
- Data transfered not encrypted
- Accepts Limited proxies
- Based on wuftpdp
- Notes on Third Party Transfer
  - GSI authentication to both servers
  - One server told to listen on certain port for connection from other
  - Other server told to connect on that port
- Can be setup with chroot
Globus Gatekeeper Security

- (x)inetd listens on port 2119
- GSI authentication
- grid-map authorisation
- Allows access to command shell
GASS

• GASS server via gatekeeper
• GSI Authentication using limited proxies
• uses https for authentication however, is not compatible with normal https clients.
• Is run on demand under non-privileged account on random port.
MyProxy

- port 7512
- Allows full proxies to be created on a trusted server
- Proxies can be created by authentication with limited proxies to MyProxy server.
Security of other services

- GSIklog/GSSklog client and daemon
  - GSI to AFS server; AFS grid-mapfile; token via SSL; Accepts Limited Proxies
- Kx509
  - Kerberos Security wrapper to CA service
- GridCVS
  - Globus GSS gserver
- Gridsite
  - x509 certificates in web browsers with GACL
- Slashgrid (/grid)
  - DN based file system using coda
Best Practices

- Use firewall but allow ports for traffic
- Keep private keys private
  - Do not send them over the net (sniffed traffic can be cracked)
  - Do not store them on network file systems (ACLs are often forgotten)
  - Do not leave user certs in No DES format (buys time if private key is stolen)
- Always keep on top of advisories
- Keep map-files, CRLS and accounts up-to-date
Security Issues in GT2

- You're giving access to your machine. Once in, a user can do anything a normal user might be able to do!
- Proxies on grid machines are dangerous! Users must trust remote machines to be safe...
- ...Remote machines must trust other grid machines in the same grid.
- Private key maintenance
  - Unix file systems, Windows Certificate stores, Netscape CertDB KeyDB, AFS, Passphrase...
More GT2 Security Issues

- You are essentially using a unix-type environment on a machine somewhere on a grid
- root on that machine can intercept any data on that machine
- data in transit is not encrypted unless one uses scp...
- Log files on remote machines are readable by default
Useful GSI and Openssl Commands

- grid-cert-info -file cert.pem
- grid-proxy-info
- openssl x509 -in cert.pem -noout XXXX
  - where XXXX = -text | -subject | -hash | -modulus | ...
- openssl pkcs12 -in certkey.p12
  - P12 contains cert key and chain to CA
- openssl crl -CApath /etc/grid-security/certificates
  -inform PEM -verify -in crl.pem
- openssl rsa -in key.pem -modulus
1. CAS request, authenticated with User credential
2. CAS reply, including restricted proxy cred:
   Community subject name
   Policy restrictions
3. Resource request, authenticated with CAS proxy
   Community subject name
   Policy restrictions
4. Resource reply

CAS Server
What rights does the community grant to this user?

Resource Server
Is this request authorized for the community?
Do the proxy restrictions authorize this request?

CAS-maintained community policy database
Local policy information
Kerberos (1)

- Ticket Granting Ticket
  - [Session key, Time Stamp, Lifetime & ID] encrypted with password

- Session key for comms.
Kerberos (2)

- **Ticket Granting Ticket**
  - [Session key, Time Stamp, Lifetime & ID] encrypted with password

- **Service Request**
  - Authenticator: encrypted [ID, IP, Time Stamp and (short) Time to Live]
  - Service Ticket: [Service Key, ID, IP, Time Stamp & Lifetime] encrypted with service password
AFS

- Kerberos steps 1 & 2 (AFS server is Kerberos Server)
- Directory Structures everything in /afs/realm/...
- Access Control
  - Based on AFS ACL
  - Groups or users can be given rlidwka
  - These apply to DIRECTORIES
  - New directories inherit parent directories' ACLs
  - File rights take unix owner's rights ie -rwxr-xr-x
  - Watch out: $HOME/mail, $HOME/.globus, etc.
PKI and Kerberos

gsiklog and/or gssklog

Client (user or user's grid job)

- needs AFS client software
- needs GSI Proxy certificate
- gsklog executable
  - (client may have to supply)
- used in same way as klog
  - except no password is required
- User gets afs ticket
  - lifetime dictated by minimum of proxy lifetime, server default and client request

Server (AFS eg afs1.hep.man.ac.uk)

- daømon on AFS server
  - as afs system:administrator
- Server listens port 750
- SSL connection using GSI
- afsgrid-mapfile
  - If success service key and service ticket are sent through socket
Kerberos and PKI

- Reverse the process?
  - kx509
- Need kerberos TGT
- Follow kinit route.
- Remember this?
- Service -> KCA
Kerberos and PKI

- Kerberos Server absorbs another service: KCA
- Client uses TGT
  - gets server ticket
- Use Server Ticket
  - send public key
  - get it signed
- Receive x509 Cert.
  - like Proxy but not GSI