SSH & GSI-X.509

Happily Living Together in Harmony

Frank Siebenlist

franks@mcs.anl.gov

MWSG@LBNL - Dec 6, 2007

GSI-X509-PKIX

- “Somewhat” difficult to setup
- “Somewhat” difficult to manage
- “Somewhat” incomprehensible trust model
- “Somewhat” difficult to explain why you need it when you already have X

- …but all our Grid middleware relies on X.509
  - …but not on PKIX
  - …just on SSL & X509Certs (& proxies)

SSH-PKI

- Secure…
  - uses same RSA pk-crypto as X509&SSL
- Relatively easy to setup
- Relatively easy to understand trust model
  - You “are” your key
- Works well with “limited” number of hosts and users
  - No registration/certification authorities
    - Actually there are… but they are the normal account admins
- User is often her own admin for authZ
- Sysadmins have few issues with an SSH-PKI
  - They do have issues with PKIX admin

authorized_keys vs gridmap

- Both simple access control list
- authorized_key maps key to account
  - By being in the account already

- gridmap maps user-name/DN to account
  - Single map per host
    (at least for gram/gridftp, but we can have gridmaps per resource also)

- ? See any similarities ?

known_hosts vs GSI Host AuthZ

- known_hosts
  - binds the host “name(s)” to a key
    - “name(s)” are ip-names & ip-addresses
    - Asserted by the client by managing the entries
      (either maintained dynamically or distributed by sysadmin)
- GSI Host AuthZ
  - Host’s X509-Cert validation
    - Cert’s CA in “trusted file/directory”
    - Or self-signed cert in “trusted file/directory”
  - Host-Cert’s AltSubject name equals host’s ip-name
- ? See any similarities ?

How to leverage those similarities? (1)

- GSI requires SSL requires X.509
  => GSI requires X.509
  - The SSL-protocol requires X.509 Certs

- Thus we cannot use the SSH protocols themselves for GSI

- …but can we use the SSH authN and authZ fabric (id_rsa,known_hosts&authorized_keys)?

How to leverage those similarities? (2)

- SSH: You are your key
  - Your public key is your “name”
- GSI: Use self-signed X.509 Certificates where the name/DN is the public key
  - Subject=DN=\PK=123a3b… (naming convention) (urn:base64:sha1: or ssh’ readable key format)
- Reuse the SSH credentials in ~/.ssh/id_rsa
  - Use that key to generate a self-signed X.509 cert
  - Copy private key to default key file used by SSL

How to leverage those similarities? (3)

- GSI Client&Server AuthN through SSL:
  - Present self-signed cert to peer
  - Validation succeeds if self-signed cert in trusted file/directory
  - Cert’s (Alt)Subject name should “equal” cert’s public key (“public-key-DN”)
- GSI Client AuthZ:
  - If an account’s authorized_keys’ file has entry for public key then gridmap file should have mapping entry for that account and the equivalent public-key-DN
- GSI Host AuthZ:
  - If the requested host’s name (ip-name/address) equals an entry in the client’s known_hosts file AND that same entry’s public key matches the host-cert’s name (“public-key-DN”) then host is OK

How to leverage those similarities? (4)

- X.509 Validation of Self-Signed Keys
  - “Somehow” self-signed user and host certs should be added to trusted file/directory on clients and servers
    - After user&host self-signed cert are created, sysadmin could collect them and install them in the right place and make them available to users to install locally.
      (would work with any default SSL install)

or

  - “Enhance” path-validation routine to **always** accept self-signed certs where subject name equals public key
    (there is no real risk associated with this policy…over time ssl dev-community could be convinced to add it to their code)

How to leverage those similarities? (5)

- GSI Client AuthZ:
  - Replicate each account’s authorized_keys’ entries in gridmap file by hand (or by cron job)

  or

  - “authorized_keys-PDP” could check account’s authorized_keys files in real-time

- GSI Host AuthZ:
  - Enhance host authZ code to optionally check whether host’s public-key-DN and ip-name/address has entry in user’s known_hosts file (requires coding), or
  - Leverage OGSA-Sec-BP’s EPR annotation to include the self-signed Host-cert (wouldn’t use known_hosts…), or
  - Get some real host authZ in place with real pluggable PDPs and such, or just a “host-gridmap-file” (=known_hosts ;-)), or
  - Just use “normal” X509-PKI host-certs

Conclusion

- We can leverage an existing SSH-PKI&authZ deployment for our GSI-PKI&authZ “almost” out of the box!
  - Well defined recipe: we can generate self-signed certs by hand from the SSH creds, move certs in the right SSL-trusted places, modify gridmap based on authorized_keys, use EPRs annotated with host certs (or add known_hosts check)
  - Or enhance path-validation, write authorized_keys PDP and known_hosts check/PDP…
- The result would truly make GSI deployment easier for small scale usage…
  - And all would work as now (even delegation/proxies with GridFTP)