Removing digital certificates from the end-user's experience of grid environments

Bruce Beckles

University of Cambridge Computing Service

Abstract

This paper describes a lightweight interim solution to the problem of authentication in grid environments that removes the burden of digital certificates from the end-user's experience of the grid environment. In doing so it increases the usability of the grid environment (and so arguably its security). It is intended to be deployed in existing grid environments, interfacing smoothly with their existing authentication and authorisation mechanisms, rather than replacing the existing authentication mechanisms. A number of other features are provided that may also be of interest to those involved in deploying grid environments.

1. Introduction

The security issues in distributed computing environments such as computational grids are necessarily more complex than those in the simpler network environments with which most computer users and system administrators are familiar. Therefore it is even more important that complexity, always the enemy of security, should be minimized as much as possible, and this is particularly true in the user domain. If the way in which the user interacts with the security mechanisms in their environment is too onerous, then the user will either use these mechanisms improperly, or attempt to circumvent them, thus making the environment less secure [1].

A number of examinations of security in grid environments have discussed the problems with the way in which digital certificates are often used in these environments ([2], [3], [4]). Some of the issues that have repeatedly arisen in these examinations are to do with the usability of the digital certificates and the related infrastructure for handling them. Some examinations (e.g. [4]) have strongly critiqued the usability of at least some of the current grid environments as regards their use of digital certificates.

This paper describes a lightweight solution to the problem of authentication that attempts to address some of these usability concerns by providing an alternative authentication mechanism that interfaces smoothly with the existing authentication and authorisation mechanisms in grid environments. It must be borne in mind that the solution described here is only designed to provide a "good enough" level of security for appropriate environments, and should not be deployed in environments requiring commercial or military grade security.

2. Statement of Objectives

The are two principal objectives for the solution described here, as follows:

- To remove digital certificates from the experience of the end-user in their interaction with a computational grid (whilst preserving a level of security consonant with the use of digital certificates) – see Figure 1.
- As far as possible, to remove manual interactions with digital certificates from the system administrators' domain and replace these interactions with scalable, automatic processes (whilst maintaining an acceptable level of security).

It should be noted that this is only intended to be an interim solution until a more robust, integrated solution to the problem of authentication in grid environments – that has the above features – is developed and in common use.

3. Trust Relationships

3.1 Analysis of Trust Relationships in traditional Grid Environments

Analyses of the trust relationships in a 'traditional' grid environment normally focus on the relationships involving the Certificate Authority, Registration Authority, the end-user or the remote resource. (In this context 'traditional' grid environment refers to a grid environment based on the Globus Toolkit 2 [5].) Clearly all these relationships are very important, but there is another set of trust relationships that are also extremely important,

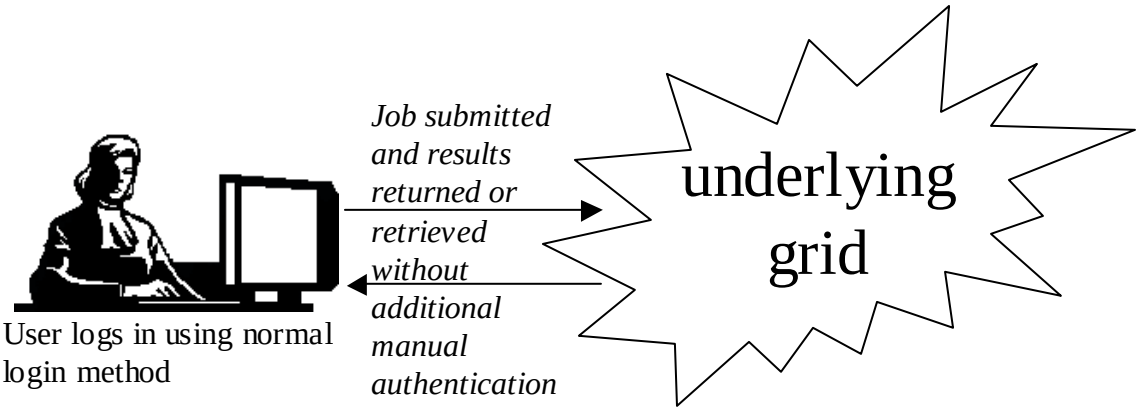

Figure 1: User experience of the grid environment using this solution

namely those involving the resource *from* which the user authenticates and/or submits their job (hereafter referred to as the *submit host*).

It may first be helpful to briefly review the process of user authentication and job submission in these grid environments:

- The user either creates a proxy certificate from their digital certificate, or retrieves a proxy certificate from an online credential repository.
- The proxy certificate is stored on the submit host (normally in a well-known location with a predictable name).
- The user and the remote resource to which they are authenticating/submitting a job mutually authenticate each other using the user's proxy certificate and the remote resource's server certificate.

Regardless of whether or not authentication to the remote resource is successful, the user's proxy certificate will be stored on the submit host. Even if the user immediately destroys this copy of their proxy certificate after authentication to the remote resource/job submission, it is clear that the user is implicitly trusting the submit host for at least as long as a copy of their proxy certificate resides on the local host.

Now consider the process of user authentication from the view of the remote resource:

- The remote resource receives a request from a host elsewhere on the network (the submit host).
- Mutual authentication occurs using the remote resource's server certificate and a proxy certificate presented by the submit host.

The remote resource thus implicitly assumes that the host that has contacted it and presented it with a proxy certificate is entitled to use that proxy certificate – this is a trust relationship between the remote resource and the submit host.

There is another implicit trust relationship between the remote resource and the submit host in most academic grid environments (and also in many grid environments where all resources are owned by the same organisation and are located on a LAN or WAN deemed to be reasonably secure from outside attacks). This relationship becomes apparent when considering what happens after a resource in the grid environment has been misused.

Normally, logs created on the remote resource during the authentication process will enable that resource's administrator to determine what certificate was presented to the resource, and this can then be used to determine the identity of the certificate owner. The logs should also reveal the identity of the submit host.

How can the remote resource's administrator decide whether it was the certificate owner or an unauthorised individual who had somehow obtained the certificate who misused their resource? At a minimum they will need to contact the administrator of the submit host and ask them to confirm who was using the submit host at the time it authenticated against the remote host. The remote resource's administrator therefore has to trust the administrator and integrity of the submit host, or else accept that they can only link misuse of their resource to a proxy certificate and not an individual.

Thus we see that in these grid environments there are important trust relationships between the users and the submit hosts they use, and between the remote resources (and their administrators) and the users' submit host(s). It is worth noting here that authentication in these grid environments does not attempt to verify the identity or integrity of the submit host.

3.2 Additional Trust Relationships in Solution described here

The solution described in this paper is one in which the trust relationship between remote resource and submit host is made explicit. As well as the ordinary trust relationships operative in a traditional environment, this solution requires a more demanding trust relationship between remote resource and submit host, which can be most succinctly put as follows:

"A remote resource trust the submit host to be 'well managed', to the extent that when the submit host asserts that a job is being submitted by a particular user, the remote resource accepts that this is true (unless the submit host or the user has been compromised) and processes the job accordingly."

The reason for this becomes apparent when the workings of this solution are examined in more detail in Section 4.

4. Outline of Solution in Operation

There are two envisaged modes of operation for this solution. In both digital certificates are used for authentication. In one mode of operation these certificates are generated by the submit host(s) (and other relevant machines in the grid environment), whilst in the other the certificates are issued by a Certificate Authority (CA).

4.1 Using certificates generated by submit hosts

- The user interactively logs in to the submit host using whatever authentication method they would normally use for this purpose (e.g. username/password). Depending on the security concerns of the grid administrator(s) there may be only a single submit host, or there may be several, or all hosts participating in the grid environment may be submit hosts, etc.
- The user uses the supplied job submission procedure on their submit host to submit a job. The submit host then generates a digital certificate with a (very) limited lifetime, signed by itself, asserting that the user is whoever it believes that user to be based on their login credentials (e.g. username). This certificate may be cached locally for some specified period of time (no greater than the lifetime of the certificate).
- The remote resource on which the job is to be executed is identified (hereafter referred to as the *execute host*), either because it was specified by the user, or by whatever process would normally be used to determine where the job should be executed (e.g. job allocation according to average load, etc.).
- All machine-machine communication first involves mutual authentication using the digital certificates of the machines in question. These certificates may have been generated by the machines in question or issued to them by a CA or some other entity, as desired.
- The submit host then passes the job, along with the public certificate it has generated, to one of the following, depending on how the grid environment has been set up:
    - The execute host, or
    - A machine elsewhere in the grid environment 'closer' to the execute host, which then passes it to another machine in the grid environment which is 'closer' to the execute host and so on until it eventually reaches the execute host, or
    - A machine elsewhere in the grid environment which has been designated as a 'router' for job submission requests, which then passes it on to another such 'router' and so on, until it arrives at the 'closest' 'router' to the execute host, which then passes it directly to the execute host.

  These latter two options are intended to facilitate features such as transparently moving jobs between resources (e.g. if the intended execute host is overloaded), multi-staged jobs, etc. Some grid environments may not have these features, in which case jobs would be sent directly from submit host to execute host.

  It is also hoped that these latter two options could be built upon to facilitate job handling in 'peer-to-peer' grid architectures where nodes may join or leave the grid environment relatively frequently.
- Each time the job is passed to an intermediate machine, that machine verifies the certificate that is included with the job, and if verification succeeds, and it trusts all previous machines in the chain, it then (optionally) digitally countersigns the certificate, and sends the job and the (optionally newly countersigned) certificate on to the next machine in the chain. Should verification fail, the machine notifies its system administrator, and the system administrator of any machines in the chain whose signature it has not been able to

verify. If the consequence of the failed verification is that the job is aborted, the submit host and the user who submitted the job are also notified.

- When the job reaches the execute host, it validates the attached certificate: if the certificate is valid, and it trusts the submit host and all intermediate machines, then it accepts the job, subject to its authorisation policy regarding the user who submitted the job. Should verification fail, the machine notifies its system administrator, and the system administrator of any machines in the chain whose signature it has not been able to verify, the submit host and the user who submitted the job.

  Depending on the underlying grid infrastructure, it may be necessary to 'unpack' the potentially multiply countersigned certificate so that the job's credentials are once more a 'simple' user certificate, signed by a single entity, before presentation to the underlying grid infrastructure.

If the system administrator of the execute host is ever in any doubt as to who actually submitted the job, etc., then, based on the certificate presented with the job, they should be able to trace the job's path back through the grid environment to the submit host. The system administrator of the submit host should be able to verify the identity of the user who submitted the job by looking at their system logs, etc.

4.2 Using certificates issued by a CA

This mode of operation is similar to that described in Section 4.1, with the following differences:

- At the time when the user wishes to submit a job, they either present the submit host with their certificate from the CA, or a proxy certificate generated from their certificate, or their certificate is retrieved on their behalf by the submit host from a secure central location. (Further details of this 'secure central location' are given in Section 5.) It is important that the Certification Practices Statement (CPS) of the CA is consulted to ensure that the user's certificates are allowed to be stored and retrieved in such a fashion.
- The submit host digitally countersigns the user's certificate before transmitting the user's job.
- Note that, as mentioned in Section 4.1, the submit host, any intermediate machines and the execute host can all use certificates issued by the CA if desired. It is presumed, although not mandated, that in this mode of operation grid administrators would wish to make use of certificates issued by their chosen CA for these machines.

4.3 Flexible security policies

It is intended that grid administrators and resource owners have the flexibility to have more or less constraining relationships with other machines and users in the grid environment. So, for instance, it is possible for intermediate machines to choose not to countersign jobs, or to countersign jobs regardless of whether they can verify the attached certificate. Also, resources can choose to refuse jobs from particular submit hosts, or which have passed through a particular intermediate machine (e.g. one known to be compromised).

Thus, security policies of the following forms are all supported (note that these are examples and not an exhaustive list):

- Refuse jobs from a particular submit host, or which have passed through a particular intermediate machine on their route to the execute host.
- For an intermediate machine, if the job is not intended for that machine, countersign it regardless of its certificate validity and send it to the next machine on its route.
- For an intermediate machine, never countersign jobs, simply send them to the next machine on their route.
- As long as the submit host's identity can be verified, accept jobs from it regardless of which intermediate machines jobs have passed through.

5. Certificate Distribution and Management

5.1 PKIBoot

PKIBoot [6] is a "plug-and-play certificate mechanism", which behaves like "DHCP for obtaining certificates", created by Peter Gutmann and implemented in his open source cryptographic toolkit, cryptlib [7]. This mechanism allows certificates to be securely retrieved from a secure central location (hereafter referred to as the *PKIBoot server*), which can also issue certificates if desired. (Details of the operation of PKIBoot are given in [6].) The certificates that can be securely retrieved in this manner include:

- The user's certificate, including the certificate's private key.

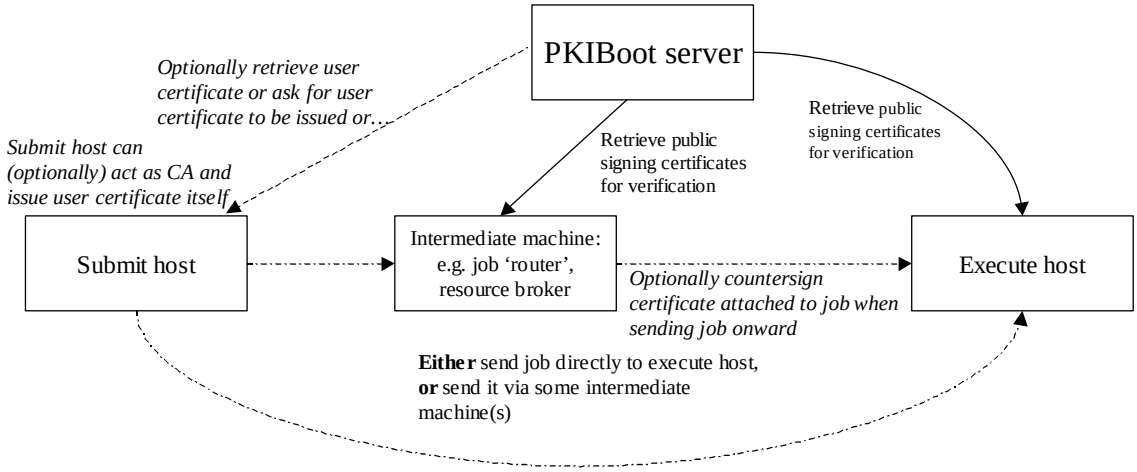

Figure 2: Simple example implementation of solution

- The public signing certificate of a CA, for verifying certificates issued by that CA.
- The public signing certificate of an arbitrary entity for the purpose of verifying that entity's identity or for validating objects digitally signed by that entity.

Thus a machine that is "PKIBoot-enabled" will not need to have already obtained the public signing certificates of a CA (or any entity that may digitally sign objects that are presented to it) in order to verify a certificate issued by that CA (or the digital signature of an object presented to it) – it can always query the PKIBoot server for the relevant public certificate. Further, machines can retrieve their own signing certificate and private key from the PKIBoot server if required, or the PKIBoot server can issue the machine with its own certificate, although these features should clearly be used with caution.

In a PKIBoot environment, users do not need to manually transport their certificates to the machine where they wish to use them as they can be retrieved from the PKIBoot server when required. Depending on exactly what the user needs to do with their certificate it may be the case that their certificate (or proxy certificate) need not be stored locally on the machine they are currently using.

5.2 Use of PKIBoot

In this solution, PKIBoot servers are deployed at convenient locations throughout the grid environment – depending on the size of the grid environment, one PKIBoot server, or one PKIBoot server per geographical site, may be sufficient. Machines will communicate with their 'nearest' PKIBoot server to retrieve the public signing certificate of a CA or other entity as necessary. The PKIBoot servers will periodically use the PKIBoot mechanism to retrieve public signing certificates from each other as necessary.

Resource administrators will merely have to configure their resource to use PKIBoot with the appropriate server and supply the public signing certificate of their resource (which can be either generated by themselves or issued to their resource by a CA) to the PKIBoot server – this is so that others can verify objects digitally signed, or certificates issued, by their resource. Alternatively, the PKIBoot server can issue resources with their own signing certificates, i.e. the PKIBoot server acts as a CA for resources in the grid environment, if desired.

If users are using certificates issued by a CA then they can store these certificates on the PKIBoot server that serves their site. As the PKIBoot mechanism also allows the PKIBoot server to issue users with certificates, this facility could be utilised to make the use of digital certificates in the grid environment entirely transparent (i.e. invisible) to the user, provided the grid environment is one in which the security implications of this are acceptable.

See Figure 2 for a simple implementation of this solution.

6. Relationship with Authentication and Authorisation Mechanisms of traditional Grid Environments

This solution is intended to interface smoothly with the authentication and authorisation mechanisms of traditional grid environments. Thus it will not attempt to actually execute any user jobs directly but will instead merely pass the job, in a suitable form, to the job submission interface of the underlying grid environment. As a user certificate or proxy certificate will be supplied with the job, it will be able to

authenticate to resources in the grid environment in the expected manner. The relevant resource will then be able to use the grid environment's authorisation mechanisms to decide, based on the certificate presented, whether to accept or reject the job.

The way in which this solution will need to interact with the underlying grid environment will depend on the mode of operation of this solution. If used as described in Section 4.1, the authentication and authorisation mechanisms of the underlying grid environment will need to be configured to accept user certificates issued by the submit host(s). An optional mechanism to automatically make available the public signing certificate of the submit host (obtained using PKIBoot) to the authentication mechanism of the grid environment on the execute hosts will be provided to assist resource administrators in this.

If used as described in Section 4.2, then the authorisation and authentication mechanisms will have to be configured to accept user certificates issued by the chosen CA, which would need to be done in any case. (Also note that in this mode of operation the use of credential repositories such as MyProxy [8] are supported.)

If user certificates are to be stored on the PKIBoot server, the CPS of the CA will need to be consulted to ensure that storing and retrieving certificates in this manner is permitted. The UK e-Science Certificate Authority [9] is a CA in common use in the UK e-Science community and the author of this paper has inspected its CPS [10] and believes that it does permit this – however the administrators of this CA should be consulted for a definitive answer.

7. Noteworthy Features

There are a number of "features" of this solution, particularly as pertaining to security, that are worth highlighting:

- The trust relationship between submit host and execute host is made explicit, and, when the mode of operation described in Section 4.1 is used, is a more demanding relationship than in traditional grid environments (as described in Section 3.2).
- Unlike in traditional grid environments, the identity of the submit host is verified as well as that of the remote resource and the user submitting the job.
  Independently verifying the identity of the submit host, as well as the user, increases the security of the environment and the degree to which the user can be held accountable for jobs submitted using their certificate for authentication. (Note that if the mode of operation described in Section 4.1 is used then, from the standpoint of the remote resource, verifying the identity of the user and the identity of the submit host are *not* independent.)
- The usability, for the end user, of the grid environment, when using this solution, is improved as their explicit interaction with digital certificates is minimised (or entirely absent, depending on the way in which this solution is deployed). This arguably makes the environment more secure [11].
- The administrative burden on resource owners and grid administrators concerning the distribution of digital certificates is reduced, and depending on the mode of operation employed, may allow them to automatically configure resources for participation in the grid environment.
- Support is provided for professionally run CAs (depending on the restrictions imposed by their CPS) and for online credential repositories such as MyProxy, if desired.
- Support is provided for transparently redirecting the submitted job from one execute host to another, which may facilitate features such as load balancing and may provide support for 'peer-to-peer' grid architectures.

8. Principal Areas of Development

It should be noted that no additional cryptographic functionality will need to be developed for this solution as the needed functionality is available in the cryptlib [7] and OpenSSL [12] toolkits. Thus the principal areas where development is required are:

- Implementing a suitable PKIBoot-based architecture,
- Developing and implementing an interface between the security layer of traditional grid environments [13] and this solution (this would also lay the foundation for interfaces with the security layers of other grid environments),
- Developing and implementing the mechanisms for the transparent movement of jobs between machines, with optional countersigning of the certificate presented with the job, and
- Developing and implementing an interface between this solution and the submission mechanisms of traditional grid environments (this would also lay the foundation for

interfaces with the job submission mechanisms of other grid environments).

9. Progress to Date

As this is a security solution, and one whose principal aim is to increase the usability of the environments in which it is to be deployed, it is essential that security and usability concerns be addressed from the earliest design stages all the way through to completion. Thus the following iterative process has been adopted in the design phase:

- Consultations with HCI practitioners and researchers with a view to improving usability and to address usability concerns
- Assessment by security experts
- Usability testing of current design using low fidelity techniques such as paper prototyping [14]

At each stage of this process the design is modified according to the feedback received and the process is iterated until a stable design is produced. Only at this point will the development phase commence. At the time of writing, the design has been inspected by HCI researchers and security experts and is largely stable. There are still a number of minor issues to be addressed, and once these have been resolved, the iterative process described above will continue with usability testing of the design.

10. Acknowledgements

The author would like to thank Professor Jon Crowcroft of the University of Cambridge for his advice, encouragement and support; Sacha Brostoff of UCL for many useful discussions concerning usability and other HCI-related issues; and the University of Cambridge Computing Service.

11. References

[1] Adams, A. and Sasse, M.A. Users are not the enemy: Why users compromise security mechanisms and how to take remedial measures (1999). Communications of the ACM, 42 (12), 40-46.

[2] Broadfoot, P. J. and Martin, A. P. A critical survey of grid security requirements and technologies. Technical Report PRG-RR-03-15, Oxford University Computing Laboratory, 2003, http://web.comlab.ox.ac.uk/oucl/work/philippa.hopcroft/Papers/PRG-RR-03-15.pdf

[3] Lock, R., and Sommerville, I. Grid Security and its use of X.509 Certificates. DIRC internal Conference submission 2002. Lancaster DIRC, Lancaster University, 2002, http://www.comp.lancs.ac.uk/computing/research/cseg/projects/dirc/papers/gridpaper.pdf

[4] Beckles, B., Brostoff, S., and Ballard, B. A first attempt: initial steps toward determining scientific users' requirements and appropriate security paradigms for computational grids (2004). Proceedings of the Workshop on Requirements Capture for Collaboration in e-Science, Edinburgh, 14-15 January 2004, 17-43.

[5] The Globus Toolkit:
http://www-unix.globus.org/toolkit/
Globus Toolkit 2.2:
http://www.globus.org/gt2.2/
Globus Toolkit 2.4:
http://www.globus.org/gt2.4/

[6] Gutmann, P. Plug-and-Play PKI: A PKI Your Mother Can Use. Paper presented at the 12th USENIX Security Symposium, Washington, 2003, http://www.cs.auckland.ac.nz/~pgut001/pubs/usenix03.pdf

[7] The cryptlib Encryption Toolkit:
http://www.cs.auckland.ac.nz/~pgut001/cryptlib/

[8] The MyProxy Online Credential Repository:
http://grid.ncsa.uiuc.edu/myproxy/

[9] The UK e-Science Certificate Authority:
http://ca.grid-support.ac.uk/

[10] UK e-Science Grid CA – Certificate Policy and Certification Practices Statement:
http://www.grid-support.ac.uk/ca/cps/index.htm

[11] Zurko, M.E. and Simon, R.T., User-centered security. Paper presented at the SIGSAC New Security Paradigms Workshop '96, Lake Arrowhead, CA, 1996.

[12] The OpenSSL toolkit:
http://www.openssl.org/

[13] The Grid Security Infrastructure (GSI) v2.0: http://www-unix.globus.org/security/v2.0/

[14] Snyder, C. Paper prototyping: The fast and easy way to design and refine user interfaces. Morgan Kaufmann Publishers, London, 2003.