

End-to-end security in the Grid

What is end-to-end security and why we need it?

Today's Grids are composed of hundreds of computing sites and most users use some sort of Workload Management System (WMS) to manage their jobs.

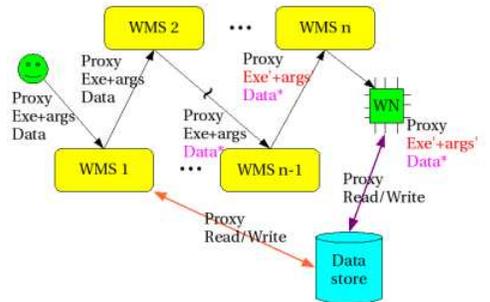
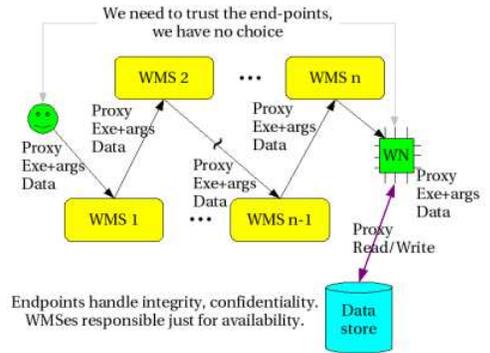
While these WMSes may be trustworthy, they are not managed either by the users nor by the sites. Since no service level agreements are needed by either the users neither the sites, we have a trust problem.

Another potential problem is that such WMSes handle a lot of user jobs, obtaining full delegation of privileges. If any such WMS gets compromised, the potential damage can be very big.

So we want to get to a situation where we do fully trust the end points, i.e. the user and the worker node running our code (because we have to), but we want to get as much trust as possible to be removed from the WMSes. A WMS should be trusted just to not lose our job, but nothing else.

There are three classes of abuse that can happen:

1. A WMS can modify the user executable and/or the associated arguments.
2. A WMS can modify the data associated with a job.
3. A WMS can access a data store in the user's name (this includes the results returned).



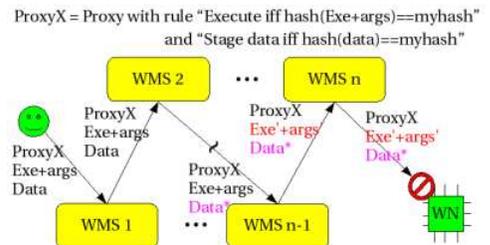
How can we implement end-to-end security?

One way to implement end-to-end security is by restricting the proxy that is being handled to the WMS (or any job handling process/service for that matter). By doing this, the WMS cannot do any damage in the user's name.

To limit the damage a proxy can do, we would need two classes of restrictions:

- a. Calculate the signature of the executable, arguments and data, and embed these signatures into the proxy. Only the new proxy is delegated from the user to any WMS.

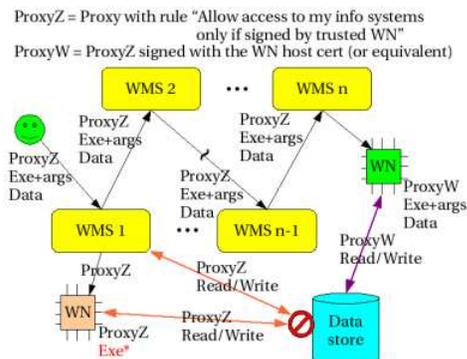
Any legitimate worker node (WN) can check the executable, arguments and data against the signatures in the proxy, and refuse to run if they do not match.



- b. Add a policy that prevents the new proxy to be used to access user data stores unless it is cross signed by a legitimate, trusted worker node.

Any legitimate worker node that gets the user job, creates a new proxy by signing it with its own host cert/proxy, unlocking it, before starting the actual execution of the job. The user job can now use the new proxy to access the data store.

On the other hand, no WMS (or any malicious WN) can access the data store.



Please also have a look at the end-to-end presentation presented at the Fall'07 MWSG (<http://indico.cern.ch/conferenceDisplay.py?confId=20203>).

How do we identify the trusted worker nodes?

You have probably noticed that the above measures rely on distinguishing a trusted worker node from any other node on the network.

This problem still need to be understood, and it is still in the early stages of R&D.

However, as a very preliminary overview, I can see two possible ways to do this:

- i. VOMS
- ii. file

[Igor Sfiligoi \(FNAL\)](#)

[Ian Alderman \(U of Madison\)](#)

Last modified: Fri Jan 18th