WLCG Traceability and Isolation WG: recommendations

Document status

This document has now been agreed upon within the WLCG Traceability and Isolation Working Group.

Aim

This document provides recommendations toward the development of a new traceability and isolation paradigm and the implementation of a new model taking advantage of new technologies and VO frameworks while keeping full trustworthy traceability and isolation of users actions.

Reasons behind traceability and isolation

Understanding the cause of security incidents is critical to contain them without having to stop the whole infrastructure. Identifying their root causes is essential to prevent them from reoccurring. For this reason, WLCG/EGI policies, in particular the Security Traceability and Logging Policy [EGI-2934] states that:

> The minimum level of traceability for use of the IT Infrastructure is to be able to identify the source of all actions (executables, file transfers, pilot jobs, portal jobs, virtual machine management, image management, etc.) and the individual who initiated them.

These policies were further refined in the case of Pilot Jobs, as part of the Policy on e-Infrastructure Multi-User Pilot Jobs [EGI-2933]:

> 5. The pilot job must use the approved system utility to map the application and data files to the actual owner of the workload and interface to local Resource Centre authorization, audit and accounting services. The owner of the user job is liable for all actions of that user job.

Fundamentally, such traceability relies on isolation of the end-user’s payload: if anything can invade another user’s environment or obtain their credentials, that entity will be able to impersonate the given user, rendering any traceability illusory. This was clearly expressed in the Policy on e-Infrastructure Multi-User Pilot Jobs [EGI-2933]:

> 8. The pilot job framework must isolate user jobs from one another, including any local data files created during execution and any inter-process communication.

Isolation mechanisms

In the past decade, the only complete mechanism for isolating processes on Linux systems was based on user accounts: unless two processes were running under two different local accounts, they would be able to manipulate each other. However, user switching in Linux is a privileged mechanism, leading to multiple issues. One tool, gLExec, was made to employ user switching to isolate user payloads from pilot jobs but was not adopted by most VOs.

In the last few years, a new isolation mechanism has become prominent in Linux: namespaces & cgroups, more generally called “containers” [Containers]. While the initial implementation required privileges, new development, after this feature proved its use, is pushing it towards a non-privileged implementation. Most distributions now support it unprivileged, including Red Hat Enterprise Linux (and derivatives) since version 7.6.

Using containers for isolation brings another advantage: the uniformity of the execution environment. As containers can bring their own execution environment, the environment in which they are executed does not matter, as long as the container image is compatible with the kernel of the underlying host. As a result, there would be almost no need to install complex dependencies in the host environment: these could be put into the containers themselves.

There are currently still at least two known major limitations to most container implementations:

- Some useful features, such as “overlay” support, have yet to mature, are not considered safe enough for unprivileged access, and thus are not available within unprivileged containers on most distributions. Development exists upstream (e.g. fuse-base overlay & unprivileged fuse), but nothing can be used in production yet. Fortunately, adequate workarounds exist (e.g. “underlay”).
- Most container implementations restrict permissions that can be obtained within the container, including the possibility to create a new container. This non-recursivity of creating containers limits the use of containers for resource providers that internally already use containers to offer a safe and known environment for VOs.

Recommendations for isolation

Regardless of the technology used for isolating pilot jobs (e.g. user switching, containers, virtual machines, …) or where they are executed (e.g. grid worker nodes, clouds, …), user payloads **MUST** be isolated from any other user’s payloads:

- Files: user payloads MUST NOT be able to steal or manipulate local files of other users’ payloads, or files from their caller (e.g. Pilot) if those contain credentials or other means to affect other users’ payloads. Proper care SHOULD also be given to files containing sensitive data (e.g. personal data).

- Processes: user payloads MUST NOT be able to manipulate (attach to, inject into, etc) processes of other users’ payloads, or their caller (e.g. Pilot) if those contain credentials or other means to affect other users’ payloads. Proper care SHOULD also be given to processes processing sensitive data (e.g. personal data).

Scenarios compatible with these recommendations

While not exhaustive, the following scenarios detail the spirit of this recommendation into technical details, to ensure a common understanding.

Using containers to isolate user payloads from VO pilots

The isolation scenario initially imagined when designing this document is a one to one replacement of the gLExec isolation layer by a container-based one: the VO pilot itself keeps VO credentials allowing it to communicate with its VO framework while the user payload is isolated from it using containers (or any technology with similar capabilities):

- Mount namespace: VO pilots MUST isolate files of other user payloads and their own files from the user payload by building a custom mount namespace which only exposes parts of the file system used by the current user payload and nothing else.
- Process ID (pid) namespace: VO pilots MUST isolate processes of other user payloads and their own processes by creating a new pid namespace dedicated to a single user payload.
- Interprocess Communication (ipc) namespace: When possible, VO pilots SHOULD create a new ipc namespace dedicated to a single user payload in order to isolate communications of other user payloads and their own internal ones.

In addition to this isolation, it MUST NOT be possible to use the credentials passed to the user payload to interact with the VO framework to obtain and start new user payloads.

Using VO pilots without VO framework capabilities

When the VO pilot doesn’t have the capacity to create a container, an alternative solution is to fully dedicate the job to the user, removing from it any credentials that could be used to interact directly with the VO framework to obtain and start new user payloads, at least by the time the first user payload is started.

However in such cases, an isolation layer MUST exist within the local job framework, isolating jobs from each other:

- Pilot jobs running on the same host MUST NOT run under the same user ID, unless there is further isolation (e.g. containers).
- If multiple user payloads are started within the same pilot job, in parallel or consecutively, they have to be from the same user.
- Shared file systems, local or remote, have to be carefully protected to avoid interactions between user payloads. In particular (but not exhaustively):
    - The home folder of the user payload MUST NOT be simultaneously writeable by payloads of other users and must be cleaned of any pre-existing files.
    - Shared writable folders MUST NOT contain executable files or files used in the execution (e.g. a library).
    - Environmental paths (PATH, PYTHONPATH, LD_LIBRARY_PATH, etc) must not contain shared writable folders.
    - Shared readable files MUST NOT contain personal credentials or other sensitive data (e.g. personal data).

Technical recommendations for isolation using containers

In order to achieve a certain unity of software and solution within VOs, drive migration efforts for sites and open the possibility of non privileged solutions, the following recommendations are suggested for isolation using containers:

- On any distribution where unprivileged containers are not fully supported, these features **SHOULD**, when privileged containers are available, be provided using a single tool* shared between all VOs providing a restricted access to privileged containers.
- On any distribution where unprivileged containers are fully supported and enabled, these features **SHOULD NOT** be provided using any SUID binary that provides more capabilities than are covered by unprivileged containers. Instead these features **SHOULD** be provided by any unprivileged container wrapper.
- When using any tool providing restricted access to privileged containers, advanced features which are not available via unprivileged containers or any other unprivileged system **SHOULD NOT** be used.

* As of 2018, the tool selected by all by all VOs for this purpose, on RHEL/SL/SLC/CentOS 6, is “Singularity”.

Maintaining full traceability

The policies mentioned in the introduction put most of the responsibility and burden of traceability directly on the sites. While sites initially were able to provide such capabilities, with the rise of pilot jobs and virtualization, this has become increasingly difficult if not impossible: VO payloads (started from Pilot jobs, Virtual Machines, Containers, …) cannot be inspected by sites, which have less and less control or access to them and can only monitor any externally visible activity.

However, during the past decade, VOs have, for their own operational reasons, built their own logging and audit systems for their pilot factories and central services, capable of providing extensive information about end-user jobs and payloads. A basic traceability challenge run in 2017 showed that the VOs were all able to provide enough information to cover a significant part of the full traceability responsibility originally assigned to the sites. The VOs can thus be made responsible for that part instead of the sites.

This means VOs need to be active participants in the incident response process and thus maintain appropriate capabilities and expertise to investigate security incidents.

Split traceability recommendations

- Sites **MUST** be able to trace activity on their systems (worker node, hypervisor, ...), to a job issued by a VO. In particular:
    - Given a time and IP address, identify the corresponding VOs and job IDs.
    - Given a time, IP address and specific activity, uniquely identify the corresponding VO and job ID.
    - Given a time period and VO (potentially with job IDs), identify the corresponding IP addresses.
- VOs **MUST** be able to trace the activity within their pilot framework. In particular:
    - Given a time and IP address, identify the corresponding users and payloads.
    - Given a time, IP address and job ID, uniquely identify the corresponding user payload.
    - Given a time and user, identify all pilot jobs that executed payloads submitted by that user.

- The retention policies of these logs **MUST** be in line with the WLCG policies.

- WLCG and EGI policies, in particular the Policy on e-Infrastructure Multi-User Pilot Jobs [EGI-2933], **SHOULD** be revised to account for such split traceability.

- Challenges verifying the capabilities of sites and VOs defined here **SHOULD** be regularly conducted.

Incident response and emergency suspension

- VOs **MUST** take appropriate actions, in particular the suspension of problematic payloads and users, within the time allocated by the corresponding policies, upon notification from the site or the infrastructure CSIRTs.
    - In case of delays or failure to accomplish such actions, sites **MUST NOT** be penalized if they suspend the whole VO until the appropriate actions are taken.
    - If a site offers a grid-wide standardized service to validate user payloads against users suspended for security reasons, VOs **SHOULD** make reasonable efforts to consume this information with their VO pilots and have mechanisms to prevent the execution of matching users.

- In collaboration with the relevant infrastructure CSIRT, VOs **SHOULD** connect the central emergency suspension mechanisms to their pilot factories, to ensure timely suspension at any time.

Traceability for storage accesses

With the replacement of gLExec by containers, where the isolation mechanism does not depend on the user itself, the pilot does not necessarily have to be equipped with user tokens (x509 certificate) for starting its payload. However, access to storage should still be controlled by the VOs and properly audited. VOs are encouraged to explore non-x509 mechanisms, including in particular mechanisms supporting restricted delegations.

While this Working Group had started discussing these issues and potential solutions, this work has now been taken over by the ongoing WLCG AuthZ WG, which is exploring solutions for this particular problem at a more general level.

References

[EGI-2933] https://documents.egi.eu/public/ShowDocument?docid=2933

[EGI-2934] https://documents.egi.eu/public/ShowDocument?docid=2934

[Containers] https://en.wikipedia.org/wiki/Container_(virtualization)